Pliable Index Coding

Siddhartha Brahma
EPFL, Lausanne
siddhartha.brahma@epfl.ch

Christina Fragouli
EPFL, Lausanne
christina.fragouli@epfl.ch

***Abstract*—We propose a new formulation of the index coding problem, where instead of demanding a specific message, clients are "pliable" and are happy to receive any one message they do not have. We prove that with this relaxation, although some instances of this problem become simple, in general the problem remains NP-hard. However, we develop heuristic algorithms that in our (extensive) experimental evaluation required only a logarithmic (in the number of messages) number of broadcast transmissions; this is an exponential improvement over traditional index coding requirements.**

I. Introduction

In the well-known Index Coding with side-information problem (ICOD), a server holds m messages, and can broadcast over a noiseless channel to a set of receivers or clients. Each client has as side information some subset of the m messages, and requests from the server a specific message she does not have. The objective is to devise an optimal coding strategy that minimizes the number of broadcast transmissions the server makes to satisfy the demands of all clients [4]. This problem has been proven to be NP-hard; infact, it is even hard to find constant-factor approximations [2], [14], [12].

But what if the clients are "pliable", and are happy to receive any one message they do not already have? There are several applications that motivate this relaxation: for example, the clients might be doing an internet search, have collected some information and are interested in receiving with low delay additional information on the subject they are searching; they do not care which specific piece of information they do receive, as long as it is something they do not already have. We term this formulation pliable index coding (PICOD). In this paper, we are interested in *linear* coding solutions for this problem.

Our first question was, whether removing the specific requirements we can find an optimal linear code in polynomial time. We hoped that this might be true, since we now have a significantly larger number of choices in what to transmit to each receiver. This turns out not to be the case; we prove that the problem remains NP-hard.

Our second question was, how many transmissions do we need in this case, i.e., what is the effect on the length of the code. In traditional index coding, Haviv *et.al.* [11] show that the minimum length linear index codes for random instances (i.e. random side information sets) of the problem is almost surely $\Omega(\sqrt{m})$, where m is the number of messages in the server. Pliable network coding can only do better, and much better in some cases. For example, assume that there are m clients and they do not have any side information; if each one of them requests a different message, with index coding we need m transmissions, while if the clients are pliable, we can simply send any one message. Although we expected that pliable coding to do better, we were still surprised to find, through extensive experimental results, that even with simple heuristic algorithms the number of transmissions grew as $\log m$ as a function of m.

The heuristic algorithms we develop also form a contribution of this paper. They are based on a bipartite graph representation of the problem, and take advantage of cases where we have "easy" solutions of the coding problem, as was the case we described above with no side information.

The paper is organized as follows. In Section III we precisely define the PICOD problem. In Section IV, we prove that finding the optimal code for PICOD is NP-Hard. In Section V we develop efficient polynomial time approximation algorithms. In Section VI we show through simulations on random instances of the problem that the algorithms work very well in practice and the required number of broadcasts is significantly less than that required in ICOD for similar settings. The paper concludes with a discussion of open problems and possible future directions of research.

II. Related Work

Over the past few years, there has been a significant amount of work on the theory of ICOD, especially for linear codes. The problem was introduced by Birk *et.al.* [4] in the context of an application in satellite communication networks. Bar-Yossef *et.al.* [2] presented the first theoretical analysis of the problem. They showed that the optimal length for a *linear* index code is given by a certain graph functional called the *minrk*. They conjectured this to be true even for non-linear codes, which was subsequently disproved by Lubetzky *et.al.* [13]. Several new graph parameters were introduced in [1] showing the strict separation of optimal solutions for different field sizes.

Building on the work of [12], [10] which investigate the connections between Index Coding and Network Coding and using information theoretic linear programs Blasiak et.al [5] prove some of the best known bounds for the index coding problem. The work of Blasiak *et.al* [5], [6] also shows several separation results between the optimal linear and non-linear index codes. These results can also be used to come up with instances in network coding that have large gaps between linear and non-linear coding rates. The best known

This work was partially supported by the European Research Council grant
NOWIRE ERC-2009-StG-240317

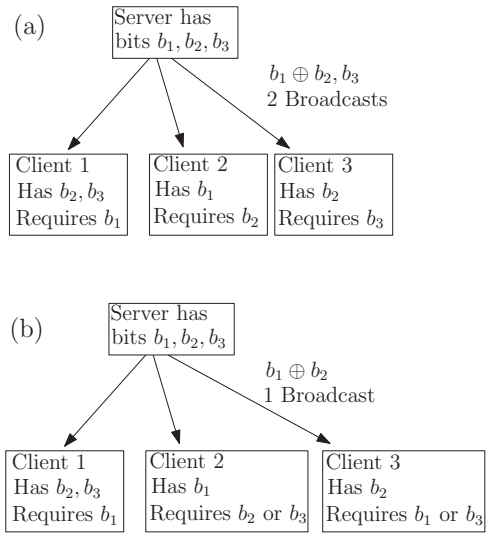

Fig. 1. (a) ICOD needs 2 broadcasts and (b) PICOD needs just one broadcast

approximation algorithms using semidefinite programming are due to Chlamtac *et. al.* [8]. There have been several other papers dealing with variants of the ICOD problem including the complementary index coding problem [7], secure index coding [9] and index coding with outerplanar side information [3].

III. Problem Definition

We will assume that the messages are bits and encodings are linear i.e. encoded bits are binary sum of the individual bits done over the field $GF(2)$. Suppose that the server has m bits of information $b_1, \cdots, b_m$ and there are n clients $c_1, \cdots, c_n$. Each client c_i knows a subset of bits $b_{N[i]}$, where $N[i]$ is a strict subset of $[m]$, as side information. Throughout this paper, $b_{N[i]}$ will mean the set $\{b_j,\, j \in N[i]\}$, $N[i]$ is a strict subset of $[m]$ and $[m] = \{1, 2, \cdots, m\}$. Given m, n and the side information sets $N[i]$, the *linear* Pliable Index Coding with Side Information (**PICOD**) problem is to devise a minimum length linear code $\mathcal{C}$ for $\{0,1\}^m$ which consists of

1) A linear encoding function E mapping $x \in \{0,1\}^m$ to $E(x) \in \{0,1\}^l$, where l is the length of the code.
2) Decoding functions $D_i, \cdots, D_n$ for the n clients such that $D_i(E(x), b_{N[i]}) = b_{k_i}$ for *some* $k_i \in [m] \setminus N[i] = \overline{N[i]}$.

As is standard, it is assumed that the server knows the side information sets. In contrast to standard ICOD where the bit requirements $k_i \in \overline{N[i]}$ are specified precisely, in PICOD it is sufficient for each client c_i to know any one bit that it does not know. To illustrate the difference, consider the scenario shown in Figure 1. In ICOD, at least 2 broadcasts are needed. Client 1 can decode b_1 from $b_1 \oplus b_2$ as it knows b_2. Client 2 can decode b_2 from $b_1 \oplus b_2$ as well and client 3 gets b_3 directly. It is easy to see that one broadcast will not suffice in this case. On the other hand in PICOD, it is sufficient to send just $b_1 \oplus b_2$ as clients 1 and 3 can decode $b_1 = b_2 \oplus (b_1 \oplus b_2)$ and client 3 can decode $b_2 = b_1 \oplus (b_1 \oplus b_2)$ using it. Note that in both cases, coding does help as the number of broadcasts is less than 3.

We also also study a generalized version of PICOD where each client is interested in exactly k bits that it does not know, for some $k \geq 1$. If the number of bits c_i does not know is less than k, then it is sufficient if it can decode $\min\{k, |\overline{N[i]}|\}$ bits. We will call this the k**-PICOD** problem, which reduces to the PICOD problem for $k = 1$.

IV. PICOD is NP-Hard

For given side information sets, the length of the optimal pliable index code cannot be worse than the length of the optimal index code. This is because the index code encodes for a specific set of required bits, which is just one of the many configurations allowed in the pliable case. However, as we show in this section, even for linear codes, PICOD is a NP-Hard problem. This will be accomplished by reducing the MONOTONE-1in3-SAT problem to the PICOD problem.

Given a 3SAT instance ϕ with all literals in non-negated form, the MONOTONE-1in3-SAT problem asks whether there is a satisfying assignment such that exactly one literal is True in each clause of the formula. MONOTONE-1in3-SAT has been shown to be NP-Hard by Schaefer [15]. Suppose ϕ is made up of M literals $\alpha_i, \cdots, \alpha_M$ and N_0 clauses

$$\phi(\alpha_1, \cdots, \alpha_M) = \bigwedge_{i=1}^{N_0} (\alpha_{i,1} \vee \alpha_{i,2} \vee \alpha_{i,3})$$

where clause i is a disjunction of the literals $\alpha_{i,1}, \alpha_{i,2}, \alpha_{i,3}$. The precise reduction is shown in the following lemma.

Lemma 4.1: Given an instance ϕ of MONOTONE-1in3-SAT as defined above, there is an instance I_{ϕ,M,N_0} of linear PICOD such that ϕ has a satisfying assignment if and only if I_{ϕ,M,N_0} has a code of length 1.

Proof: Given the MONOTONE-1in3-SAT instance ϕ, consider an instance I_{ϕ,M,N_0} of PICOD defined as follows

1) There are N_0 clients c_i, $i \in [N_0]$ corresponding to the clauses where c_i corresponds to clause i.
2) There are M bits b_j, $j \in [M]$ corresponding to the literals where bit b_j corresponds to literal α_j.
3) The side information set for c_i consists of all the bits that *do not* correspond to the literals in clause i. That is

$$N[i] = \{j, \text{ literal } \alpha_j \text{ is not in clause } i\}$$

Therefore, $|N[i]| = M - 3$ and $|\overline{N[i]}| = 3$ for all $i \in [N_0]$.

Suppose there is a linear code of length 1 that is a solution to I_{ϕ,M,N_0}. It is of the following form

$$S = b_{j_1} \oplus b_{j_2} \cdots \oplus b_{j_s}$$

for some $j_1, \cdots, j_s \in [M]$. Let $J_s = \{j_1, \cdots, j_s\}$. Since every client c_i must be able to decode at least one bit not in $N[i]$ and there is only one code bit, $\forall i \in [N_0]$, there exists $j_t \in J_s$ such that $j_t \in \overline{N[i]}$. Since b_{j_i} has to be decodable by c_i, there can be at most one such index. Thus, the set J_s has the property that exactly one of its members is present

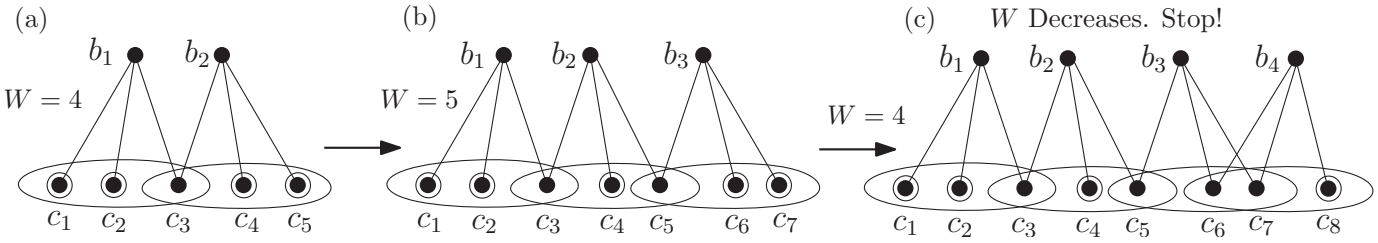

Fig. 2. (a) A subgraph showing the coding scheme (b),(c) Greedy covering of the client vertices by neighboring bit vertices

in each $\overline{N[i]}$. Clearly, if we set the corresponding literals $\{\alpha_{j_k}, 1 \leq k \leq s\}$ to True and others to False, we make sure that all the clauses (which correspond to the clients) are satisfied and exactly one literal in each clause is True, which therefore satisfies ϕ. Thus, a code of length 1 for $I_{\phi,M,N}$ can be used to generate a satisfying assignment for ϕ that has exactly one True literal in each clause. Exactly the same argument can be reproduced backwards to prove the converse, which completes the reduction. Finally, it is easy to see that the reduction can be accomplished in polynomial time. ■

Since MONOTONE-1in3-SAT is NP-Hard, Lemma 4.1 implies that PICOD is NP-Hard. Therefore, in general we cannot hope to get polynomial time algorithms for finding the minimum length code unless P=NP. On the other hand, given the practical nature of the problem, it will be worthwhile to devise efficient approximation algorithms that work well in practice.

V. Approximation Algorithms

It is not hard to come up with instances where PICOD is is quite easy to solve. For example, if there exists a bit such that is not in the side information sets of any client, then it is sufficient to send just that bit. Further, if the side information sets of each client covers all the bits except one, then sending the sum of all the bits solves the problem. Our main idea is to find subsets of bits and clients that have a combination of these properties.

It is simpler to visualize an instance of PICOD using a bipartite graph G with m vertices on one side representing the bits (termed as "bit vertices") and n vertices on the other side representing the clients (termed as "client vertices"). We will identify the vertices by the bits or clients they represent. There is an edge from b_j to c_i if $j \in \overline{N[i]}$. In Figure 2(a) shown above, bit b_1 is not in the side information sets of clients c_1, c_2, c_3 and hence is connected to them in G. The set of client indices connected to b_j is denoted by $N_b[j]$ i.e. the clients $c_{N_b[j]}$ are connected to b_j.

Consider two bits j_1 and j_2 and their neighborhoods $N_b[j_1]$ and $N_b[j_2]$ in G. We distinguish the vertices in $c_{N_b[j_1]} \cup c_{N_b[j_2]}$ into two types depending on the number of bits they are adjacent to. Consider the client vertices that are adjacent to exactly one bit vertex. The set of such vertices for a particular subset B of bit vertices in G is denoted by $W(B)$. In Figure 2(a) for $B = \{b_1, b_2\}$, $W(B) = \{c_1, c_2, c_4, c_5\}$ and are marked by the double circles. Note that if $b_1 \oplus b_2$ is sent to these $|W(B)| = 4$ vertices, they can decode a bit that it does not have. c_1 and c_2 can decode $b_1 = b_2 \oplus (b_1 \oplus b_2)$ as they know b_2. Similarly, c_4 and c_5 can decode b_2 as they know b_1. On the other hand, the set of clients which are adjacent to more than one bit, $\{c_3\}$ in this case, cannot decode either b_1 or b_2.

The same logic can be extended to more than 2 bit vertices. All the $|W(B)|$ vertices in such a configuration will be able to decode a bit that it does not have from the sum of the bits in B. Thus, it is enough to broadcast the sum bit to "satisfy" all the $|W(B)|$ clients. We propose two algorithms based on this idea by greedily maximizing their number. The third algorithm is based on a reduction to ICOD.

A. Algorithm GRCOV1

Using the above intuition, we try to find a set of bit vertices B such that $|W(B)|$ is maximized. Rather than trying to obtain the *maximum* such set, we greedily find a *maximal* such set. Let $B = \{b_{v_1}, \cdots, b_{v_t}\}$ be a set of bit vertices. B is a maximal set if for any vertex $b_{v_{t+1}} \notin B$, $|W(B \cup \{b_{v_{t+1}}\})| < |W(B)|$. To find a maximal set, we start with the null set and keep on adding bit vertices that greedily maximizes $|W(B)|$ in each step and stop when no further additions are possible without decreasing $|W(B)|$. For example Figure 2 represents a possible sequence of operations where $B = \{b_1, b_2, b_3\}$ is a maximal set. When b_3 is added, the cardinality of $W(B)$ increases but further addition of b_4 decreases it, in which case we stop.

Once such a maximal set B_M is obtained, the encoded bit consisting of the sum of all the bits in it is broadcast. The vertices in B_M and $W(B_M)$ are removed and the algorithm is resumed for the remaining graph, until all the client vertices are covered. Thus the code consists of all the encoded bits derived from the maximal sets. We call this GRCOV1 (for greedy cover). A simple implementation of the algorithm has a running time of $O(mn^2)$.

B. Algorithm GRCOV2

GRCOV2 is a slight modification of GRCOV1, where the bit vertices in the maximal set are not removed. The intuition is that the bit vertices obtained in one maximal set may be useful for the remaining client vertices as well. The algorithm is presented in pseudo-code form below. The running time in this case is again $O(mn^2)$.

C. Algorithm SETCOV

As a comparison, we propose another algorithm that is based on a reduction to the ICOD problem. In an instance of PICOD, it is sufficient that c_i is able to decode any one

Algorithm 2 GRCOV2

Initialize: G is the bipartite graph corresponding to an instance of PICOD with n client vertices and m bit vertices.
Initialize: $UC \leftarrow \{c_1, \cdots, c_n\}$, $num\_bits \leftarrow 0$, $\mathcal{C} = \{\}$.
  while $UC \neq \varnothing$ **do**
    $B \leftarrow \varnothing$.
    while B is not a *maximal set* **do**
      Find bit vertex b_v such that $|W(B \cup \{b_v\})|$ is maximized.
      $B \leftarrow B \cup \{b_v\}$.
    end while
    $UC \leftarrow UC \setminus W(B)$.
    Remove $W(B)$ and all edges connected to it from G.
    $\mathcal{C} \leftarrow \mathcal{C} \cup \{\bigoplus_{t=1}^{|B|} b_{v_t}, b_{v_t} \in B\}$
    $num\_bits \leftarrow num\_bits + 1$.
  end while
  Output $\mathcal{C}, num\_bits$.

bit in $\overline{N[i]}$. We split client c_i into $\left|\overline{N[i]}\right|$ "pseudo-clients" $c_{i,1}, \cdots, c_{i,|\overline{N[i]}|}$ each with a *distinct* bit from $\overline{N[i]}$ as a requirement and with the same *common* side information sets. Therefore, in total we get $\sum_{i=1}^{n} \left|\overline{N[i]}\right|$ pseudo-clients with properly defined side information sets. This is an instance of the ICOD problem and can be solved using one of the algorithms proposed in [4]. We use the simplest one based on greedy clique cover.

Let the set of encoded bits be E. For the greedy clique cover algorithm, each encoded bit allows for decoding in one step. In other words, each client can decode its required bit using just one encoded bit and each encoded bit can satisfy the requirements of a certain number of pseudo-clients. This naturally defines a "covering" relationship where an encoded bit covers a set of pseudo-clients. Also note that each pseudo-client in fact corresponds to an original client, the one from which it was created. Therefore, for each encoded bit $e_t \in E$ we can define the set of original clients that it "covers"

$$C(e_t) = \{c_{t,1}, \cdots, c_{t,s_t}\} \subseteq \{c_1, \cdots, c_n\}$$

In fact, the same client can occur in several of these covering sets. Since we only need a client to be able to decode a single bit that it does not know, it is sufficient to find a collection of $C(e_t)$ that covers all the clients. Further we want to minimize the size of this collection for the optimal encoding. This is precisely an instance of the minimum SET-COVER problem with clients being the elements and the $C(e_t)$ being the sets. In our simulations, we use the standard greedy approximation algorithm to solve it. Thus overall, the algorithm (which we term SETCOV), works by reducing the PICOD instance to an ICOD instance by splitting each client and then solving a set cover problem to minimize the number of encoded bits. The total running time for a non-optimized implementation of the algorithm is $O(m^2n^2)$.

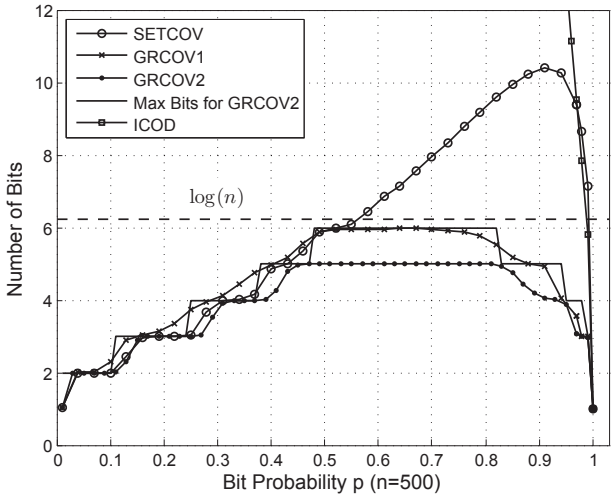

Fig. 3. Average performance of PICOD algorithms

D. Algorithm k-GRCOV2

To solve the more general k-PICOD problem defined in Section III, we propose a natural generalization of the GRCOV2 algorithm presented above. Each client vertex c_i maintains a counter initialized to $\min\{k, |\overline{N[i]}|\}$. The algorithm proceeds as above, except that after finding each maximal set B, we delete the edges connecting B and $W(B)$ in the graph and decrement the counter of the vertices in $W(B)$. Only those client vertices whose counters have become zero are removed from the graph. The running time of this algorithm, which we call k-GRCOV2, is $O(kmn^2)$.

VI. Experimental Results

In this section, we present results of extensive simulations on random instances of PICOD and k-PICOD to evaluate the performance of the algorithms presented in the previous section. We have chosen both the number of clients and number of bits to be $m = n = 500$. The independent random instances are generated as a function of the bit probability p - the probability of a client knowing a particular bit. For the purposes of comparison, we generate the instances in a manner such that each client requires a distinct bit in the ICOD problem. For ICOD, we use the simple clique cover algorithm presented in [4]. The behavior of other algorithms for ICOD was similar in our range of interest.

In Figure 3, which shows the average performance of the algorithms over several runs (more than 10,000 for each value of p), we observe that there is a significant difference between the performance of the PICOD algorithms and the ICOD algorithm for the same p. While all the three PICOD algorithms proposed above take less than 11 bits on an average, the ICOD solution hovers in this range only for $p \geq 0.95$ which is the case when the side information sets are very dense. In the remaining range of p values, PICOD is much more efficient in terms of the number of broadcasts. Thus, unless the side information sets are very dense, the number of broadcasts needed for PICOD is much less than ICOD.

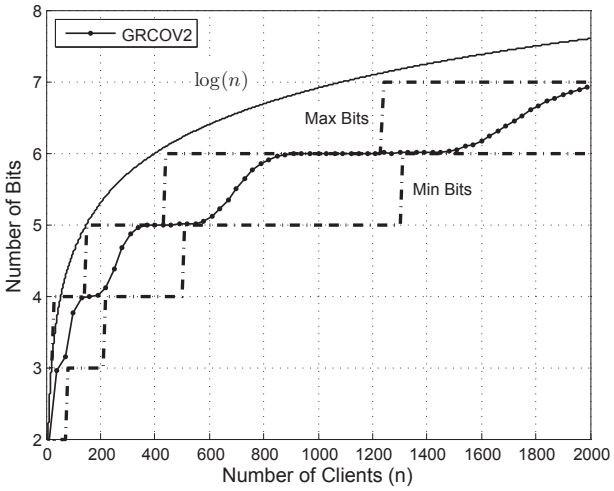

Fig. 4. Average performance of GRCOV2 for varying n

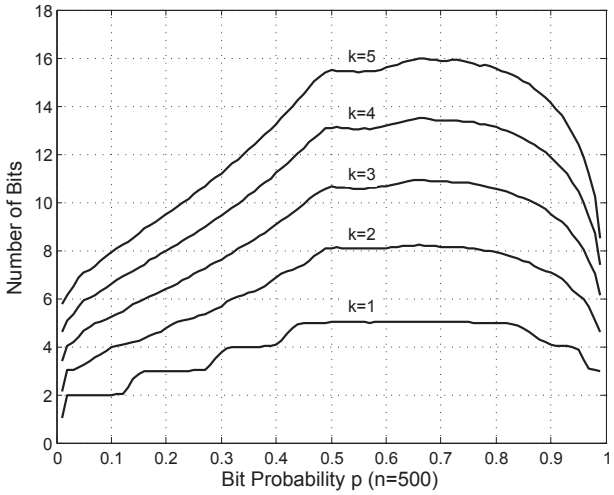

Fig. 5. Average performance of k-GRCOV2 for different values of k

Among the three algorithms presented above, GRCOV2 performs the best in the entire range closely followed by GRCOV1. SETCOV shows an interesting trend, being better than GRCOV2 in the range $p \leq 0.5$ but performing worse after that. This can be partly explained by the fact that for lower values of p the side information sets are sparse which makes the bipartite graph dense and hence each client occurs in several covering sets. This partially compensates for the fact that we are using a suboptimal set cover algorithm. This is not true for denser side information sets.

As shown in Figure 3, the maximum number of bits required by GRCOV2 in our experiments for any PICOD instance is always below $\log(500) \approx 6.22$. As further evidence, in Figure 4, we plot the average number of bits required by GRCOV2 for different values of n (with $m = n$), instances being generated uniformly at random. In each case, the maximum number of bits required for a particular value of n is at most $\log(n)$, for $n \geq 60$. We conjecture that in the asymptotic limit of large n, this is true in general. This would mean an exponential gap between the number of broadcasts needed for the PICOD and ICOD problem for random graph instances. Therefore, in scenarios where PICOD is a more appropriate model, it is better to apply the algorithms presented above.

Results of the performance of the approximation algorithm for the k-PICOD problem are presented in Figure 4 for different values of k. The k-GRCOV2 algorithm performs very well in practice with less than 17 broadcasts being required on an average for each one of the 500 clients to decode 5 additional bits.

VII. Conclusion

The algorithms presented in the paper show that pliable index coding can be solved with significantly fewer number of broadcasts than traditional index coding under similar settings. This makes it more suitable for use in scenarios where the clients are not interested in specific messages. Deriving upper and lower bounds on the number of broadcasts needed in such scenarios as a function of the number of messages and clients and suitably defined parameters on the side information sets is an open problem. The algorithms proposed in this paper suggest that the upper bounds may be logarithmic in nature. Proving approximation ratio bounds on the performance of algorithm GRCOV2 is another open problem.

References

[1] N. Alon, A. Hassidim, E. Lubetzky, U. Stav and A. Weinstein, "Broadcasting with side information", *Proc. of the 49th Annual IEEE Symposium on Foundations of Computer Science*, pp. 823–832, 2008.

[2] Z. Bar-Yossef, Y. Birk, T. S. Jayram and T. Kol, "Index Coding with Side Information", *Proc. of 47th Annual IEEE Symposium on Foundations of Computer Science*, pp. 197-206, 2006.

[3] Y. Berliner and M. Langberg, "Index coding with outerplanar side information", *Proc. of IEEE International Symposium on Information Theory*, pp. 806–810, 2011.

[4] Y. Birk and T. Kol. "Informed-Source Coding-on-Demand (ISCOD) over Broadcast Channels", *Proc. of INFOCOM*, vol. 3, pp. 1257–1264, 1998.

[5] A. Blasiak, R. Kleinberg and E. Lubetzsky, "Index coding via linear programming", Available at http://arxiv.org/abs/1004.1379.

[6] A. Blasiak, R. Kleinberg and E. Lubetzsky, "Lexicographic Products and the Power of Non-Linear Network Coding", *52nd IEEE Symposium on Foundations of Computer Science*, pp. 609–618, 2011.

[7] M. A. R. Chaudhry, Z. Asad, A. Sprintson and M. Langberg, "On the Complementary Index Coding Problem", *Proc. of IEEE International Symposium on Information Theory*, pp. 244–248, 2011.

[8] E. Chlamtac and I. Haviv, "Linear Index Coding via Semidefinite Programming", Available at http://arxiv.org/abs/1107.1958.

[9] S. H. Dau, V. Skachek and Y. M. Chee, "On secure Index Coding with Side Information", *Proc. of IEEE International Symposium on Information Theory*, pp. 983–987, 2011.

[10] S. El Rouayheb, A. Sprintson and C. Georghiades, "On the relation between the Index Coding and the Network Coding problems", *Proc. of the IEEE International Symposium on Information Theory*, pp. 1823–1827, 2008.

[11] I. Haviv and M. Langberg, "On Linear Index Coding for Random Graphs", Available at http://arxiv.org/abs/1107.0390.

[12] M. Langberg and A. Sprintson, "On the hardness of approximating the network coding capacity", *Proc. of the IEEE International Symposium on Information Theory*, pp. 315–319, 2008.

[13] E. Lubetzky and U. Stav, "Non-linear Index Coding Outperforming the Linear Optimum", *Proc. of 48th Annual IEEE Symposium on Foundations of Computer Science*, pp. 161-168, 2007.

[14] R. Peeters,"Orthogonal representations over finite fields and the chromatic number of graphs",*Combinatorica*, vol. 16, no. 3, pp. 417–431, 1996.

[15] T. J. Schaefer. "The complexity of satisfiability problems", *Proc. of the 10th Annual ACM Symposium on Theory of Computing*, pp. 216226, 1978.