Instantly Decodable Network Codes for Real-Time Applications

Anh Le*, Arash S. Tehrani+, Alexandros G. Dimakis†, Athina Markopoulou*

* {*anh.le, athina*}*@uci.edu* – University of California, Irvine
+ *arash.sabertehrani@usc.edu* – University of Southern California
† *dimakis@austin.utexas.edu* – University of Texas, Austin

***Abstract*—We consider the scenario of broadcasting for real-time applications and loss recovery via instantly decodable network coding. Past work focused on minimizing the completion delay, which is not the right objective for real-time applications that have strict deadlines. In this work, we are interested in finding a code that is instantly decodable by the maximum number of users. First, we prove that this problem is NP-Hard in the general case. Then we consider the practical probabilistic scenario, where users have i.i.d. loss probability, and the number of packets is linear or polynomial in the number of users. In this case, we provide a polynomial-time (in the number of users) algorithm that finds the optimal coded packet. Simulation results show that the proposed coding scheme significantly outperforms an optimal repetition code and a COPE-like greedy scheme.**

I. Introduction

Broadcasting data to multiple users is widely used in several wireless applications, ranging from satellite communications to WiFi networks. Wireless transmissions are subject to packet losses due to channel impairments, such as wireless fading and interference. Previous work has shown that coding can improve transmission efficiency, throughput, and delay over broadcast erasure channels [1]–[6]. Intuitively, the diversity of lost packets across different users creates coding opportunities that can improve various performance metrics.

In this work, we are interested in packet recovery for real-time applications, such as, fast-paced multi-player games and live video streaming. Real-time applications have two distinct characteristics: (i) they have strict and urgent deadlines, *i.e.*, a packet is outdated after a short amount of time, and (ii) they can tolerate some loses, *e.g.*, a game client can restore the game state by re-syncing periodically [7]. Despite this fault tolerance, these applications can suffer significantly from packet losses, that often lead to jittery game animation and low quality video playback. Hence, it is highly desirable to recover packet losses with very low delay and within a very narrow coding window. Motivated by the above observations, we focus on coding schemes for loss recovery that allows instantaneous decoding, *i.e.*, zero delay. These coding schemes are also known as Instantly Decodable Network Codes (IDNC).

Previous work on IDNC [4]–[6], [8]–[10] focused on minimizing the completion delay, *i.e.*, the time it takes to recover all the losses at all users. We formulate a different problem that is more relevant to real-time applications, called *Real-Time IDNC*: Consider a source that broadcasts a set of packets, $\mathcal{X}$, to a set of users. Each user u wants all packets in $\mathcal{X}$ and already knows a subset of them, $\mathcal{H}_u \subset \mathcal{X}$, for example through previous transmissions. The goal is to choose one (potentially coded) packet to broadcast from the source, so as to maximize the number of users who can immediately recover one lost packet. This problem is highly relevant in practice, yet - to the best of our knowledge - only solved in heuristic ways so far, *e.g.*, see [4], [11]. Our main contributions are the following:

- We show that Real-Time IDNC is NP-hard. To do so, we first map Real-Time IDNC to the Maximum Clique problem in an IDNC graph (to be precisely defined in Section III). We then show that the Maximum Clique problem is equivalent to an Integer Quadratic Program (IQP) formulation. Finally, we provide a reduction from a well-known NP-Hard problem (the Exact Cover by 3-Sets) to this IQP problem.
- We analyze random instances of the problem, where each packet is successfully received by each user randomly and independently with the same probability. This problem, referred to as Random Real-Time IDNC, corresponds to a Maximum Clique problem on an appropriately created random IDNC graph. Surprisingly, we show that when the number of packets is linear or polynomial in the number of users, the Maximum Clique problem can be solved with high probability on this particular family of random graphs, by a polynomial-time (in the number of users) algorithm, which we refer to as the *Max Clique* algorithm.

We implement and compare our coding scheme, Max Clique, against two baselines: an optimal repetition code and a COPE-like greedy scheme proposed in [4]. Simulations show that our scheme significantly outperforms these state-of-the-art schemes over a range of scenarios, for the loss probability varying from .01 to .99. For example, for 20 users and 20 packets, our scheme improves by a factor of 1.3 on average, and performs up to 1.6 times better than the COPE-like scheme and up to 3.8 times better than the optimal repetition code.

The remainder of this paper is organized as follows. Section II discusses related work. Section III formulates the problem. Section IV describes the maximum clique and integer program formulations as well as the proof of NP-completeness. Section V analyzes the probabilistic version (Random Real-Time IDNC) and provides the polynomial-time algorithm to find a maximum clique w.h.p. Section VI evaluates and compares our coding scheme with existing schemes. Section VII concludes the paper.

This work has been partially supported by the following grants: NSF CAREER Award 0747110, AFOSR MURI Award FA9550-09-0643, AFOSR Award FA9550-10-1-0310, NSF Award 1055099, NSF Award 1218235, and research gifts by Intel and Cisco. This work was conducted when A. Dimakis was with USC and A. Le was visiting USC. A. Le and A. Markopoulou are also affiliated with CPCC and CalIT2 at UC Irvine.

II. Related Work

Katti *et al.* [11] proposed COPE, an opportunistic inter-session network coding scheme for wireless networks. Encoded packets are chosen so that they are immediately decodable at the next hop. The algorithm considers combining packets in a FIFO way (as stored in the transmitting queue) and greedily maximizes the number of receivers that can decode in the next time slot. Keller *et al.* [4] investigate algorithms that minimize decoding delay, including two algorithms that allow for instantaneous decoding: a COPE-like opportunistic algorithm and a simple repetition algorithm. In Section VI, we compare our proposed algorithm to these two algorithms.

Sadeghi *et al.* [5] improved the opportunistic algorithm in [4] by giving high priority to packets that are needed by a large number of users. The authors also gave an Integer Linear Program formulation to the problem of finding an instantly decodable packet that maximizes the number of beneficiary users. Furthermore, they show that it is NP-hard based on the *Set Packing* problem. We note that their formulation differs from ours since they require that a coded packet must be instantly decodable by *all* users, where there may be some users that do not benefit from the packet. This may lead to a suboptimal solution, *i.e.*, there may be a coded packet that is only instantly decodable *by some but not all* users but is beneficial to a larger number of users. Our formulation ensures that we find this optimal packet.

Sorour *et al.* have an extensive line of work investigating instantly decodable codes [6], [8]–[10], [12], [13], focusing on minimizing the completion delay. They introduced the term Instantly Decodable Network Coding (IDNC) that we adopt in this work. Based on a stochastic shortest path formulation, they proposed a heuristic algorithm to minimize the completion delay [6]. In [8], they introduced the notion of *generalized* IDNC problem, which does not require the transmitted code to be decodable by all users as in the *strict* version studied previously [4], [6], [12]. Real-Time IDNC considers the generalized version. Furthermore, in [8], they relate finding an optimal IDNC code to the maximum clique problem in IDNC graphs and suggest that it is NP-Hard; however, no explicit reduction was provided.

Li *et al.* [14] use IDNC for video streaming and show that, for independent channels and sufficiently large video file, their schemes are asymptotically throughput-optimal subject to hard deadline constraints when there are *no more than three users*. In contrast, we consider an arbitrary number of users and we provide the optimal single transmission.

Also related to Real-Time IDNC are the Index Coding, its variation [15], and Data Exchange problems. In the interest of space, we refer the reader to our technical report [16] for the detailed comparisons. We note that both of these problems, however, are fundamentally different from Real-Time IDNC as their objective is to minimize the number of overall transmissions.

III. Problem Formulation

Let $\mathcal{U} = \{u_1, \cdots, u_n\}$ be the set of n users, and $\mathcal{P} = \{p_1, \cdots, p_m\}$ be the set of m packets. We assume that the original m packets were broadcast by a base station. Due to packet loss, each of n users missed some of the m packets. We denote the set of packets that were successfully received by user i by $\mathcal{H}_i$. Furthermore, let $\mathcal{W}_i$ be the set of packets that user i still wants, *i.e.*, $\mathcal{W}_i = \mathcal{P} \setminus \mathcal{H}_i$. Consistent with [12] and [17], we call $\mathcal{H}$'s and $\mathcal{W}$'s the "Has" and "Want" sets.

After the initial broadcast, the base station tries to recover the losses, $\mathcal{W}$'s, by sending coded packets and exploiting the side information of the already delivered packets, $\mathcal{H}$'s. Let the $n \times m$ matrix $\mathbf{A}$ be the identification matrix for the side information of the users, *i.e.*, entry $a_{ij} = 1$ if user u_i wants packet p_j and 0 otherwise. $\mathbf{A}$ is also called a feedback matrix, as in [6], [8]–[10], [12], [13]. Let us clarify this by an example.

Example 1. Consider a scenario with 3 users and 6 packets. Furthermore, assume that after the initial broadcast, user u_1 successfully received packets p_1 and p_2; user u_2 received p_3 and p_5; and user u_3 received p_3 and p_6. Thus the side information matrix is

$$\mathbf{A} = \begin{pmatrix} 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 \end{pmatrix}.$$

To deliver the packets in the Want sets of the users, we focus on instantly decodable, lightweight coding schemes that operate in GF(2). For a set of packets $\mathcal{M}$, their corresponding coded packet c is $c = \bigoplus_{p_i \in \mathcal{M}} p_i$, where $\bigoplus$ denotes the binary sum.

Definition 1. *A coded packet, $c^{\mathcal{N}}$, is* instantly decodable *with respect to a set of users, $\mathcal{N}$, if and only if*

(i) Every user, $u_i \in \mathcal{N}$, can decode $c^{\mathcal{N}}$ immediately upon reception to recover a packet $p^i \in \mathcal{W}_i$. That is, each user in $\mathcal{N}$ benefits *from $c^{\mathcal{N}}$ by recovering one of the packets from its want set.*

(ii) Every packet in the binary sum of $c^{\mathcal{N}}$ is wanted by at least one user in $\mathcal{N}$.

For example, for the scenario of Example 1, the coded packet $c_1^{\{u_1,u_2,u_3\}} = p_1 \oplus p_3$ is instantly decodable with respect to $\{u_1, u_2, u_3\}$ since u_1 can recover p_3, while u_2 and u_3 can get p_1. Meanwhile, $c_2^{\{u_2,u_3\}} = p_5 \oplus p_6$ is not instantly decodable with respect to u_1. Furthermore, we do not consider $c_3^{\{u_2,u_3\}} = p_3 \oplus p_5 \oplus p_6$ instantly decodable with respect to $\{u_2, u_3\}$ since although $c_3^{\{u_2,u_3\}}$ can be decoded by u_2 and u_3, p_3, which is a component of $c_3^{\{u_2,u_3\}}$, is not wanted by either u_2 or u_3. From here on, we may omit the superscript $\mathcal{N}$ of $c^{\mathcal{N}}$ when there is no ambiguity.

We would like the coded packet to be immediately beneficial to as many users as possible. Thus, our notion of optimality is with respect to the cardinality of the set of beneficiary users $|\mathcal{N}|$.

The Real-Time IDNC Problem: Given a side information matrix $\mathbf{A}$, find the optimal instantly decodable packet $c^{\mathcal{N}}$.

IV. Maximum Cliques in IDNC Graphs

Given a side information matrix $\mathbf{A}$, we form an Instantly Decodable Network Coding (IDNC) graph corresponding to $\mathbf{A}$ as in [12]: We create a vertex v_{ij} when user u_i still wants packet p_j. For instance, for matrix A in Example 1, there is a vertex for each entry 1 in the matrix. Given a vertex v_{ij}, we use the term *user index* of v_{ij} to indicate i and *packet index* of v_{ij} to indicate j. There is an edge between two vertices v_{ij} and $v_{k\ell}$ if one of the below conditions hold:

(i) $j = \ell$: In this case, both users u_i and u_k wants the same packet $p = p_j = p_\ell$.

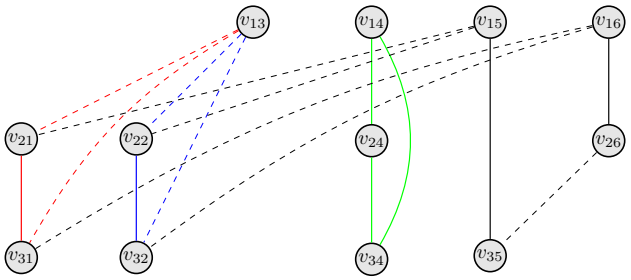


Fig. 1. The Instantly Decodable Network Coding (IDNC) graph of Example 1. Solid edges are edges of type (i) and dashed edges are edges of type (ii). There are three maximum cliques: $\{v_{13}, v_{21}, v_{31}\}$, $\{v_{13}, v_{22}, v_{32}\}$, and $\{v_{14}, v_{24}, v_{34}\}$, all of which are of size 3.

(ii) $p_j \in \mathcal{H}_k$ and $p_\ell \in \mathcal{H}_i$: In this case, user u_k has packet p_j that user u_i still wants, and vice versa.

Denote the IDNC graph corresponding to a matrix $\mathbf{A}$ by $G^{\mathbf{A}} = (\mathcal{V}, \mathcal{E})$. Figure 1 shows the IDNC graph corresponding to the side information matrix given in Example 1.

A. Cliques and Instantly Decodable Packets

Proposition 1. *Finding an optimal instantly decodable code given a side information matrix* $\mathbf{A}$ *is equivalent to finding a maximum clique in the corresponding IDNC graph* $G^{\mathbf{A}}$.

For the interest of space, we refer the reader to our technical report [16] for the proof of this proposition. Intuitively, let us consider the clique involving v_{13}, v_{21} and v_{31} in Example 1. XORing all packets corresponding to vertices of this clique, *i.e.*, $p_1 \oplus p_3$, forms an instantly decodable packet because (i) user 1 must have p_1, and users 2 and 3 must have p_3, otherwise there are no edges (v_{13}, v_{21}) and (v_{13}, v_{31}), and (ii) each component of the coded packet is wanted by the user corresponding to the row of the vertex.

B. NP-Completeness

Finding a maximum clique in a general graph is well known to be NP-Hard. This result, however, is not directly applicable to IDNC graphs as they have special structural properties. In this section, we will show that the problem of finding a maximum clique in an IDNC graph is indeed NP-Hard. We show this by first showing that finding a maximum clique in an IDNC graph is equivalent to finding an optimal solution to an Integer Quadratic Programming (IQP) problem. We then describe a reduction from a well-known NP-Complete problem, Exact Cover by 3-Sets, to the decision version of the IQP problem.

1) Integer Quadratic Programming Formulation: Given a side information matrix $\mathbf{A}$ of size $n \times m$, we formulate the IQP problem as follows. Let $\mathbf{r}$ be a binary $n \times 1$ vector: $r_i \in \{0,1\}, i = 1, \cdots, n$. Similarly, let $\mathbf{c}$ be a binary $m \times 1$ vector: $c_j \in \{0,1\}, j = 1, \cdots, m$. Below is the IQP problem for $\mathbf{A}$:

$$\text{Maximize:} \quad V = \mathbf{r}^T \mathbf{A}\, \mathbf{c} = \sum_{i=1}^{n} \sum_{j=1}^{m} r_i\, c_j\, a_{ij}.$$

$$\text{Subject to:} \quad r_i \sum_{j=1}^{m} c_j\, a_{ij} \leq 1,\ \forall i = 1, \cdots, n\,. \qquad (1)$$

$$r_i, c_j \in \{0,1\}\,. \qquad (2)$$

Proposition 2. *Given a side information matrix* $\mathbf{A}$ *and its IDNC graph* $G^{\mathbf{A}}$*, finding a maximum clique in* $G^{\mathbf{A}}$ *is equivalent to finding an optimal solution to the corresponding IQP.*

For the interest of space, we refer the reader to [16] for the proof of this proposition. Intuitively, let us consider the clique $\{v_{13}, v_{22}, v_{23}\}$ in Example 1. The corresponding solution to the IQP is by choosing $\mathbf{r}$ and $\mathbf{c}$ based on the user and packet indices of the vertices, *i.e.*, $r_1 = r_2 = r_3 = 1$, $c_2 = c_3 = 1$, and $c_1 = c_4 = c_5 = c_6 = 0$. The key observation is that since each row of the matrix has at most 1 vertex in a clique, the first constraint of the IQP is always satisfied.

2) Reduction from Exact Cover by 3-Sets: Given a side information matrix $\mathbf{A}$, the decision version of the IQP problem for $\mathbf{A}$, denoted as *D-IQP*, asks the following question: "Is there a feasible solution whose objective value equals N, for some $N > 0$?"

Proposition 3. *The D-IQP problem is NP-Complete.*

Proof: D-IQP is in NP since given a feasible pair of vectors $\mathbf{r}$ and $\mathbf{c}$, we can compute the objective value in polynomial $O(nm)$ time. In what follow, we show a reduction from the Exact Cover by 3-Sets (X3C) problem to D-IQP. X3C is well-known to be an NP-Complete problem [18] and is defined as follows:

Definition 2. *Given a set* $\mathcal{E}$ *of* $3k$ *elements:* $\mathcal{E} = \{e_1, \cdots, e_{3k}\}$*, and a collection* $\mathcal{F} = \{\mathcal{S}_1, \cdots, \mathcal{S}_\ell\}$ *of subsets* $\mathcal{S}_i \subset \mathcal{E}$ *and* $|\mathcal{S}_i| = 3$*, for* $i \in [1, \ell], \ell > k$*. The X3C problem asks the following question: "Are there* k *sets in* $\mathcal{F}$ *whose union is* $\mathcal{E}$*?"*

The reduction: Given any instance of X3C, we create $3k$ users, $u_1, \cdots, u_{3k}$, and ℓ packets, $p_1, \cdots, p_\ell$. The users correspond to the elements $e_i, i \in [1, 3k]$, and the packets correspond to the sets $S_j, j \in [1, \ell]$. We form the side information matrix $\mathbf{A}^{\text{X3C}}$ corresponding to this X3C instance by setting $a_{ij} = 1$ if $e_i \in \mathcal{S}_j$ and 0 otherwise.

Next, we will show that there is a feasible solution to the D-IQP for $\mathbf{A}^{\text{X3C}}$ whose objective value V equals $3k$ if and only if there are k sets $\mathcal{S}_{j_1}, \cdots, \mathcal{S}_{j_k}$ whose union is $\mathcal{E}$.

($\Rightarrow$) Let $\mathbf{r}$ and $\mathbf{c}$ be the pair of vectors of the feasible solution whose objective value $V = 3k$. First, observe that all r_i, for $i = 1, \cdots, 3k$, must equal 1; otherwise, assume there exists some index $t \in [1, 3k]$ where $r_t = 0$, then $V = \sum_{i=0}^{3k} r_i \sum_{j=0}^{\ell} c_j a_{ij} = \sum_{i=0, i\neq t}^{3k} r_i \sum_{j=0}^{\ell} c_j a_{ij} < 3k$, since each term $r_i \sum_{j=0}^{\ell} c_j a_{ij}$ is at most 1 by constraint (1). This is a contradiction.

Next, we create the corresponding solution to the X3C problem using $\mathbf{c}$. In particular, for $j = 1, \cdots, \ell$, we select S_j if $c_j = 1$. Because $V = 3k$ and $r_i = 1$ for all i, $\sum_{j=1}^{\ell} c_j a_{ij} = 1, \quad$ for $i = 1, \cdots, 3k$. Thus, for a user index $s \in [1, 3k]$, there exists a *unique* packet index $t \in [1, \ell]$, where $c_t\, a_{st} = 1$, which means $c_t = a_{st} = 1$. By construction, we selected set S_t, and also this S_t covers element s as $a_{st} = 1$. Therefore, each element is contained in exactly one set.

($\Leftarrow$) Let $\mathcal{S}_{j_1}, \cdots, \mathcal{S}_{j_k}$ be the solution to the X3C problem. We create the corresponding solution to the D-IQP problem as follows. First, for $\mathbf{r}$, let $r_i = 1$, for all $i = 1, \cdots, 3k$. Then, for $\mathbf{c}$, let $\mathcal{J} = \{j_1, \cdots, j_k\}$, and for $j = 1, \cdots, \ell$, set $c_j = 1$ if $j \in \mathcal{J}$ and 0 otherwise. Since $\mathcal{S}_{j_1}, \cdots, \mathcal{S}_{j_k}$ covers all $3k$ elements and each set has only 3 elements, each element e_s appears in exactly one set $\mathcal{C}_{j_t}$ for some $t \in [1, k]$, and $c_{j_t} = 1$. Thus, for each

element $s \in [1, 3k]$, $\sum_{j=0}^{\ell} c_j a_{sj} = c_{j_t} a_{s\, j_t} = 1 \cdot 1 = 1$ Given the above $\mathbf{r}$ and $\mathbf{s}$, $V = \sum_{i=0}^{3k} r_i \sum_{j=0}^{\ell} c_j a_{ij} = 3k \cdot 1 = 3k$. ■

From Propositions 1, 2, and 3, we have the following main result of this work.

Theorem 4. *Given a side information matrix $\mathbf{A}$ and its IDNC graph $G^{\mathbf{A}}$, finding a maximum clique in $G^{\mathbf{A}}$, and equivalently, an optimal instantly decodable packet, is NP-Hard. Their corresponding decision versions are NP-Complete.*

V. Maximum Cliques in Random IDNC Graphs

In this section, we investigate Random Real-Time IDNC. In particular, we assume that each user, u_i, $i \in [1, n]$, fails to receive a packet, p_j, $j \in [1, m]$, with the same probability $p \in (0, 1)$ independently. For ease of analysis, we assume that m is linear in n: $m = d\, n$, for some constant $d > 0$. (Our results also hold when m is polynomial in n.) A random IDNC graph, denoted as $G^{\mathbf{A}}(p)$, is the graph corresponding to a side information matrix $\mathbf{A}$, whose each entry equals 1 with probability p and 0 with probability $q = 1 - p$ independently. Next, we analyze the size of the maximum clique, *i.e.*, the clique number, of random IDNC graphs.

The main results of this section are as follows:

(i) For any $p \in (0, 1)$, the clique number for almost every graph in $G^{\mathbf{A}}(p)$ is linear in n. In particular, it equals $j^* p q^{j^*-1} n$, where $j^* = \text{argmax}\ j p q^{j-1}$, $j^* \in \mathbb{N}$. With high probability, the optimal recovery packet involves combining j^* packets.

(ii) With high probability, the maximum clique can be found in polynomial time, $O(n\, m^{j^*+\delta})$, where δ is a small constant parameter, and we provide an explicit algorithm to find it. Consequently, the optimal recovery packet can be computed in polynomial time in n.

Comparison to Erdős-Rényi Random Graphs: Clique numbers of Erdős-Rényi random graphs with n vertices and $p = 1/2$ are known to be close to $2 \log_2 n$. However, it is widely conjectured that for any constant $\epsilon > 0$, there does not exist a polynomial-time algorithm for finding cliques of size $(1+\epsilon) \log_2 n$ with significant probability [19]. In contrast, for random IDNC graphs with $n \times m$ vertices, we show that when m is linear or polynomial in n, the clique numbers are linear in n and the corresponding cliques can be found in polynomial time in n.

A. Clique Number of Random IDNC Graphs

First, observe that any k 1's that lie in the same column form a clique of size k. Since the expected number of 1's in a single column is np, the expected size of single-column cliques is np. As a result, we expect the maximum clique size to be linear in n.

Fix a set $\mathcal{C}_j$ of j columns. A row r is said to be *good* with respect to $\mathcal{C}_j$ if among the j columns, it has 1 one and $j-1$ zeros. The probability that a row is good w.r.t. $\mathcal{C}_j$ is $f(j) = j\, p\, q^{j-1}$. Let $Z_{\mathcal{C}_j}$ be the number of good rows w.r.t $\mathcal{C}_j$. Then $Z_{\mathcal{C}_j}$ has a binomial distribution: $\text{Bin}(n, f(j))$.

Let $X_{\mathcal{C}_j}$ be the size of the maximum clique that has at least one vertex on every column in $\mathcal{C}_j$, *i.e.*, the clique touches j columns. Observe that if $j = 1$, then $f(1) = p$, and $X_{\mathcal{C}_1} = Z_{\mathcal{C}_1}$, which is the number of 1's in the chosen column. Thus, $X_{\mathcal{C}_1}$ has a Binomial distribution: $\text{Bin}(n, p)$. For $j > 1$, $X_{\mathcal{C}_j} \neq Z_{\mathcal{C}_j}$ since the set of good rows may not have a 1 in every column in $\mathcal{C}_j$. The following lemma states that for a large k, where $k \overset{def}{=} Z_{\mathcal{C}_j} \sim \text{Bin}(n, f(j))$, *i.e.*, given large enough n, $X_{\mathcal{C}_j} = Z_{\mathcal{C}_j}$ with high probability.

Lemma 5. *For a set of constant j columns $\mathcal{C}_j$, there exists a constant $k_j > 0$ such that for all $k \geq k_j$,*

$$Pr[Z_{\mathcal{C}_j} = X_{\mathcal{C}_j} \mid Z_{\mathcal{C}_j} = k] \geq 1 - j \left(\frac{j-1}{j}\right)^k .$$

Proof: For $k \geq j > 0$, let B_k^j denote the number of ways to put k 1's into a matrix of size $k \times j$ such that (i) each row has one 1, and (ii) each column has at least one 1. Note that $B_k^1 = 1$, and we have the following recurrence:

$$B_k^j = j^k - \binom{j}{1} B_k^{j-1} - \binom{j}{2} B_k^{j-2} \cdots - \binom{j}{j-1} B_k^1 . \quad (1)$$

This recurrence states that the number of ways to put k 1's into k rows (each row has one 1) using exactly j columns equals to the number of ways to put k 1's into k rows without any column restriction subtracts the cases where there are $1, 2, \cdots, j-1$ empty columns. It can be shown by induction (details are in [16]) that $B_k^j = \sum_{i=0}^{j-1} (-1)^i \binom{j}{i} (j-i)^k$. Thus, we have that $B_k^j = j^k - j(j-1)^k + \sum_{i=2}^{j-1} (-1)^i \binom{j}{i} (j-i)^k$.

Let k_j be the minimum positive integer value of k such that $\sum_{i=2}^{j-1} (-1)^i \binom{j}{i} (j-i)^k \geq 0$. Then, for all $k \geq k_j$, $\Pr[Z_{\mathcal{C}_j} = X_{\mathcal{C}_j} \mid Z_{\mathcal{C}_j} = k] = \frac{B_k^j}{j^k} \geq \frac{j^k - j(j-1)^k}{j^k}$. ■

The following lemma states that $X_{\mathcal{C}_j}$, the size of the maximum clique that touches all j columns, heavily concentrates around $nf(j)$ for large n.

Lemma 6. *For a set of constant j columns $\mathcal{C}_j$ and any constant $c > 1$, let $\mu = nf(j)$ and $\delta = \sqrt{\frac{3c \ln n}{n f(j)}}$. For a large n such that $\mu - \mu\delta \geq k_j$ (k_j is as in Lemma 5), we have*

$$Pr[\,|\, X_{\mathcal{C}_j} - \mu \,|\, \geq \mu\delta] \leq \frac{2}{n^c} + 2\mu\delta\, j (1 - \frac{1}{j})^{\mu - \mu\delta} .$$

This probability goes to 0 as $n \to \infty$.

For the interest of space, we refer the reader to our technical report [16] for the proof. Intuitively, this result follows from $X_{\mathcal{C}_j} = Z_{\mathcal{C}_j}$ w.h.p. (Lemma 5), and the fact that the Binomial distributed $Z_{\mathcal{C}_j}$, the number of good rows, concentrates heavily around its mean, $nf(j)$. Note that $\mu\delta$ is $\Theta(\sqrt{n \ln n})$; thus, $X_{\mathcal{C}_j}$ is within $\Theta(\sqrt{n \ln n})$ of $nf(j)$ w.h.p.

Next, for a constant j, let X_j be the size of the maximum clique that touches *any* j columns. X_j also heavily concentrates around $nf(j)$. Formally,

Theorem 7. *For a constant j and any constant $c > j$, let $\mu = nf(j)$ and $\delta = \sqrt{\frac{3c \ln n}{n f(j)}}$. For a large n such that $\mu - \mu\delta \geq k_j$ (k_j is as in Lemma 5), we have*

$$Pr[\,|\, X_j - \mu \,|\, \geq \mu\delta] \leq \frac{2d^j}{n^{c-j}} + 2d^j n^j \mu\delta\, j (1 - \frac{1}{j})^{\mu - \mu\delta} .$$

This probability goes to 0 as $n \to \infty$.

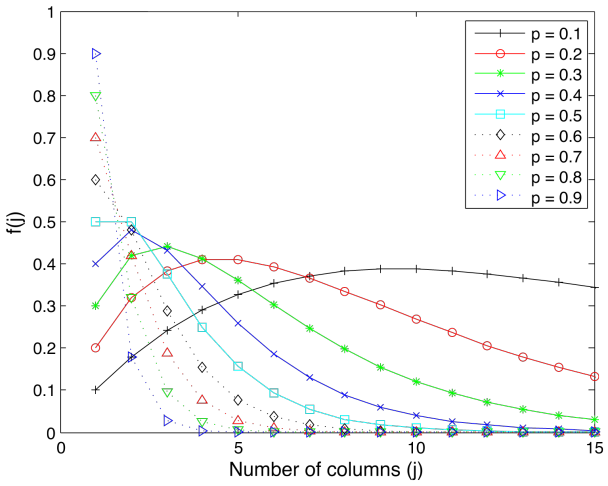


Fig. 2. Plot of $f(j) = jp(1-p)^{j-1}$ for different loss rate p

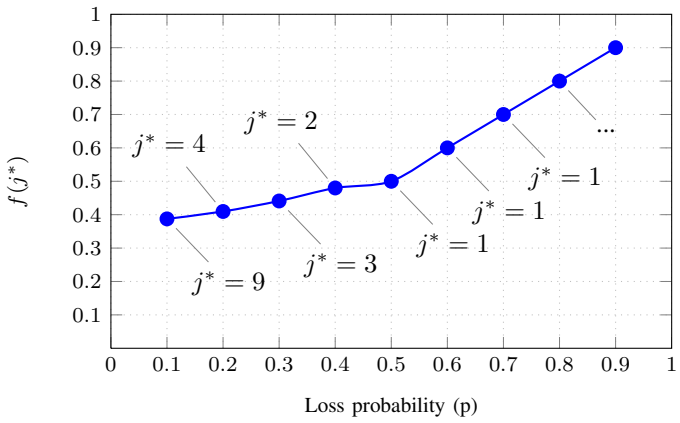


Fig. 3. Values of $f(j^*)$ and its corresponding j^*. The clique number heavily concentrates around $f(j^*) \times n$, and j^* is the number of packets should be coded together.

Proof: The proof is by using the union bound on the result of Lemma 6:

$$\begin{aligned}
&\Pr[\,|\,X_j - \mu\,| \geq \mu\delta] = \Pr[\cup_{\mathcal{C}_j}\,|\,X_{\mathcal{C}_j} - \mu\,| \geq \mu\delta] \\
&\leq \binom{m}{j} \Pr[\,|\,X_{\mathcal{C}_j} - \mu\,| \geq \mu\delta] \\
&\leq m^j \left(\frac{2}{n^c} + 2\mu\delta\, j(1-1/j)^{\mu-\mu\delta} \right) \\
&\leq \frac{2d^j}{n^{c-j}} + 2d^j n^j \mu\delta\, j(1-1/j)^{\mu-\mu\delta}\,.
\end{aligned}$$

■

We note that the above concentration result also holds when the number of packets, m, is polynomial in the number of user, n, *i.e.*, $m = n^d$, for some constant $d > 0$. However, it needs a larger constant c ($c > d\,j$), which means less concentration. Apparently, the results do not hold when m is exponential in n. However, the cases where m is either linear or polynomial in n are sufficient for practical purposes as in real-time applications, such as [7], m is often linear in n.

Now let $j^* = \text{argmax} f(j), j^* \in \mathbb{N}$. There may be a set of consecutive values of $j \in \mathbb{N}$ that maximize $f(j)$, in that case, pick j^* to be the smallest one among them. For a constant p, j^* and $f(j^*)$ are also constant.

Algorithm 1 Finding the Maximum Clique

Input: p: loss probability, n: number of users, m: number of packets, $\mathbf{A}$: side information matrix of size $n \times m$.
Output: $\mathcal{I}^*$: vertices of the maximum clique

1: $j^* \leftarrow \min(m, \text{argmax}_{j\in\mathbb{N}} f(j))$
2: $\mathcal{I}^* = \emptyset$
3: **for each** combination of j columns out of m columns, where $j \in [j^* - \delta, j^* + \delta]$
4: $\quad \mathcal{I} = \emptyset$
5: $\quad$ **for** $r = 1 \rightarrow n$ **do**
6: $\quad\quad$ **if** row r has only one 1 at column c **then**
7: $\quad\quad\quad$ Add (r, c) to $\mathcal{I}$
8: $\quad\quad$ **end if**
9: $\quad$ **end for**
10: $\quad$ **if** $|\mathcal{I}| > |\mathcal{I}^*|$ **then**
11: $\quad\quad \mathcal{I}^* = \mathcal{I}$
12: $\quad$ **end if**
13: **end**

Corollary 8. *For a sufficiently large n, with high probability, the maximum clique touches a constant number j^* of columns, where $j^* = argmax f(j)$.*

Intuitively, this follows from the above result that the size of the maximum clique that touches j columns heavily concentrates around $nf(j)$. The proof is provided in [16].

Fig. 2 plots the function $f(j)$ for different values of p. This plot shows that (i) for $p >= 0.5$, $f(j)$ is a decreasing function, and for $p < 0.5$, $f(j)$ initially increases then decreases, and (ii) j^* increases as p decreases, which suggests that *the number of packets should be coded together increases when the loss rate decreases*. Fig. 3 plots the values of $f(j^*)$ and the corresponding values of j^*. An important observation from Fig. 3 is that even when the loss rate is small, the clique size is still high. For instance, when $p = 0.1$, $j^* = 9$ and $f(j^*) \simeq 0.38$, *i.e.*, the optimal coded packet involves coding 9 plain packets together, and this packet will benefit about 38% of the users.

B. Finding a Maximum Clique

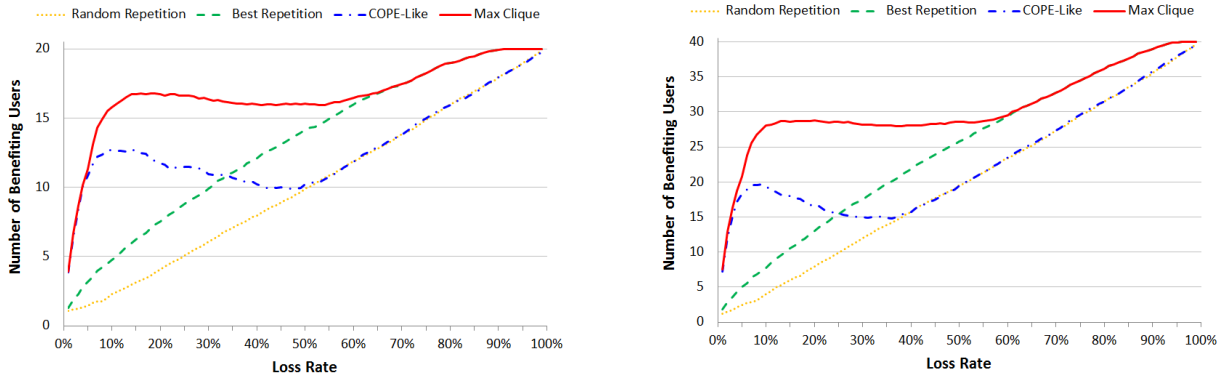
Based on the analysis in the previous section, we propose Algorithm 1 to find a maximum clique in a given random IDNC graph. Algorithm 1 examines all cliques that touch j columns, for all j combinations of m columns, where j is within a small constant δ neighborhood of j^*. In the case j^* is larger than m, j^* is set to equals to m (Line 1), exploiting the fact that for $j < j^*$, $f(j)$ is an increasing function as shown in Fig. 2.

Complexity. In Algorithm 1, the for each loop starting at Line 3 runs at most $2\delta\binom{m}{j^*+\delta}$ times. The for loop starting at Line 5 runs n times. The if condition check at Line 6 examines up to $j^* + \delta$ entries. Thus, the total runtime of Algorithm 1 is at most $2\delta\binom{m}{j^*+\delta} n(j^* + \delta) = O(n\, m^{j^*+\delta})$, *i.e.*, polynomial in n when m is linear or polynomial in n.

Given the vertices of the maximum clique output by Algorithm 1, one can readily compute an optimal instantly decodable packet by XORing the packets whose indices correspond to the packet indices of the output vertices, as indicated in Proposition 1.

VI. Performance Evaluation

In this section, we use simulation to compare the performance of the proposed Max Clique algorithm (Algorithm 1) to an optimal repetition-based algorithm, called Best Repetition, and



(a) $n = 20$ users, $m = 20$ packets

(b) $n = 40$ users, $m = 20$ packets

Fig. 4. Performance of the proposed Max Clique coding scheme in comparison with those of the Best Repetition and COPE-Like coding schemes.

a COPE-like greedy-based algorithm proposed in [4]. These two baseline algorithms operate as follows:

The *Best Repetition* algorithm rebroadcasts the uncoded packet that is wanted by the most number of users. This is inherently the best repetition strategy. The *COPE-Like* algorithm goes through all the packets that are still wanted by at least one user in a random order, and it tries to compute a coded packet that is instantly decodable to *all* users. In particular, it begins by selecting the first packet, $c = p_1$. It then goes through the rest of the packets one by one. At each step j, $j > 1$, it XORs the packet p_j under consideration with c: $c = c \oplus p_j$, if the result is still instantly decodable to all users, and it skips p_j otherwise. For reference, we also include the Random Repetition algorithm, which resends a random packet that is still wanted by at least one user.

Setting. For each loss rate ranging from 1% to 99% per 1% increment, we randomly generate 100 side information matrices. We then run the algorithms on these matrices. For the Max Clique algorithm, we set δ, the neighborhood around j^*, to 3. Fig. 4 plots the average numbers of beneficiary users as a function of loss rate for the two parameter settings: $\{n = 20, m = 20\}$ and $\{n = 40, m = 20\}$. For clarity, we skip plotting the standard deviations (they are ranging from 0 to 3 for all algorithms).

Results. In Fig. 4, one can see that the proposed Max Clique algorithm consistently and significantly outperforms all other algorithms. In particular, for the case $\{n = 20, m = 20\}$, on average, Max Clique performs 1.3 times better than both the Best Repetition and COPE-Like. For the loss rates between 40% and 50%, Max Clique performs up to 1.6 times better than the COPE-Like algorithm, and for the loss rates between 10% and 15%, Max Clique performs up to 3.8 times better than the Best Repetition algorithm. Similar trend but higher improvement: 1.35 times on average and up to 4.5 times, could be observed for the case $\{n = 40, m = 20\}$.

When the loss rate is larger than a certain threshold (65% in Fig. 4(a)), the performance of Max Clique is similar to that of the Best Repetition, which suggests that Max Clique also tries to select the best uncoded packet. This is because an uncoded packet now benefits many users due to high loss rate. When the loss rate is larger than another threshold (50% in Fig. 4(a)), the performance of COPE-Like is similar to that of Random Repetition, which suggests that packets cannot be coded together while being instantly decodable to all users. This is because when the loss rate is high, given any pair of 2 plain packets, there exists a user who lost both w.h.p.

VII. Conclusion

In this paper, we formulate the Real-Time IDNC problem, which seeks to compute a recovery packet that is immediately beneficial to the maximum number of users. Our analysis shows that Real-Time IDNC is NP-Hard. We then analyze the Random Real-Time IDNC, where each user is assumed to lose every packet with the same probability independently. When the number of packets is linear or polynomial in the number of users, we show that the optimal packet could be computed in polynomial time (in the number of users) with high probability. In the future, we plan to extend this work from a single time slot to a constant number of time slots for recovery.

References

[1] M. Luby, M. Watson, T. Gasiba, T. Stockhammer, W. Xu, "Raptor codes for reliable download delivery in wireless broadcast systems," in *CCNC '06*, pp. 192–197.

[2] J. W. Byers, M. Luby, M. Mitzenmacher, A. Rege, "A digital fountain approach to reliable distribution of bulk data," in *Sigcomm '98*, Vancouver, pp. 56–67.

[3] D. Nguyen, T. Tran, T. Nguyen, and B. Bose, "Wireless Broadcast Using Network Coding, *Tran. Vehicular Tech.*, vol. 58, no. 2, pp. 914–925, Feb. 2009.

[4] L. Keller, E. Drinea, and C. Fragouli, "Online Broadcasting with Network Coding," in *NetCod '08*, Hong Kong, pp. 1–6.

[5] P. Sadeghi, D. Traskov, and R. Koetter, "Adaptive network coding for broadcast channels," in *NetCod '09*, Lausanne, pp. 80–85.

[6] S. Sorour and S. Valaee, "On Minimizing Broadcast Completion Delay for Instantly Decodable Network Coding," in *ICC '10*, Cape Town, pp. 1–5.

[7] A. Le, L. Keller, C. Fragouli, and A. Markopoulou, "MicroPlay: A Networking Framework for Local Multiplayer Games," in *SIGCOMM MobiGames Workshop '12*, Helsinki, pp.155–160.

[8] S. Sorour and S. Valaee, "Minimum Broadcast Decoding Delay for Generalized Instantly Decodable Network Coding," in *Globecom '10*, Miami, pp. 1–5.

[9] S. Sorour and S. Valaee, "Completion Delay Minimization for Instantly Decodable Network Coding with Limited Feedback," in *ICC '11*, Kyoto, pp. 1–5.

[10] S. Sorour and S. Valaee, "Completion delay reduction in lossy feedback scenarios for instantly decodable network coding," in *PIMRC '11*, Toronto, pp. 2025–2029.

[11] S. Katti, H. Rahul, W. Hu; D. Katabi, M. Medard, and J. Crowcroft, "XORs in the Air: Practical Wireless Network Coding," in *ToN*, vol. 16, no. 3, pp. 497–510, Jun. 2008.

[12] S. Sorour and S. Valaee, "Adaptive network coded retransmission scheme for wireless multicast," in *ISIT '09*, Seoul, pp. 2577–2581.

[13] S. Sorour and S. Valaee, "On densifying coding opportunities in instantly decodable network coding graphs," in *ISIT '12*, Cambridge, pp. 2456–2460.

[14] X. Li, C. Wang, and X. Lin, "On The Capacity of Immediately-Decodable Coding Schemes for Wireless Stored-Video Broadcast with Hard Deadline Constraints," in *JSAC*, vol. 29, no. 5, pp. 1094–1105, May 2011.

[15] S. Brahma, C. Fragouli, "Pliable Index Coding," in *ISIT '12*, Boston, pp. 2251–2255.

[16] A. Le, A. S. Tehrani, A. G. Dimakis, A. Markopoulou, "Instantly Decodable Network Codes for Real-Time Applications," Technical Report, Feb. 2013, [online] http://arxiv.org/abs/1303.7197

[17] S. El Rouayheb, M. Chaudhry, and A. Sprintson, "On the Minimum Number of Transmissions in Single-Hop Wireless Coding Networks," in *ITW '07*, pp. 120–125.

[18] M. R. Garey and D. S. Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness," W. H. Freeman & Co. New York, NY, 1979.

[19] A. Juels and M. Peinado, "Hiding Cliques for Cryptographic Security," in *SODA '98*, San Francisco, pp. 678–684.