This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE ICC 2011 proceedings

Completion Delay Minimization for Instantly Decodable Network Coding with Limited Feedback

Sameh Sorour and Shahrokh Valaee
The Edward S. Rogers Sr. Department of Electrical and Computer Engineering
University of Toronto
Toronto, ON, M5S 3G4, Canada
Email:{samehsorour, valaee}@comm.utoronto.ca

***Abstract*— In this paper, we consider the problem of minimizing the broadcast completion delay for instantly decodable network coding with limited feedback. We first extend the stochastic shortest path formulation of the full feedback scenario in [1] to the limited feedback scenario. We then show that the resulting formulation is more complicated to solve than the original one but has its same properties and structure. Based on this result, we design four variants of the algorithms employed in [1] with four different approaches to deal with un-acknowledged transmissions. We finally compare these four algorithms through extensive simulations and show that the algorithm that temporarily avoids all un-acknowledged transmissions in subsequent coding decisions and transmissions can result in a tolerable degradation compared to the full feedback performance while using a much lower feedback frequency.**

***Index Terms*—Wireless Broadcast; Instantly Decodable Network Coding; Limited Feedback.**

I. Introduction

Network coding (NC) has shown great abilities to substantially improve transmission efficiency, throughput and delay over broadcast erasure channels. These merits can greatly impact the performance of emerging and proliferating streaming and real-time applications that require quick and reliable data transmission over lossy channels, such as satellite imaging, roadside to vehicle safety and streaming communications and internet TV. In [1]–[4], an important subclass of NC, namely the instantly decodable network coding (IDNC), was considered due to its numerous desirable properties. First, IDNC provides instant packet recovery upon appropriate packet reception, a property that general linear and random NC lack and which is very desirable to reduce decoding delay in the aforementioned applications. Moreover, IDNC encoding can be implemented using binary XOR, which eliminates the complicated operations over large Galois fields and the coefficient reporting overhead, required by linear NC. This XOR encoding also simplifies the decoding process at the receivers as each receiver can simply cancel out the packets it already knows. This eliminates the need for matrix inversion at the receivers, which is a computational bottleneck in linear and random NC [2]. Finally, no buffers are needed at the receivers to store coded packets for future decoding possibilities. These simple decoding and no-buffer properties allow the design of simple and cost-efficient receivers.

Two main types of IDNC can be distinguished in the literature, namely the sender driven and receiver driven IDNC. The former type, studied in [2], [3], forces the sender to transmit coded packets including at most one missing packet from each of the receivers. The latter type, we employed in [4], relaxes the instant decodability constraint at the sender, but forces the receivers to discard all packet combinations including more than one of their missing source packets. It is easy to infer that the receiver driven IDNC allows more efficient packet combinations at the sender, while preserving all the aforementioned benefits of IDNC. In this paper, we consider this type and refer to it as IDNC for short.

One major drawback of IDNC is that it is not a rate-optimal approach and thus may result in high completion delay. In [1], we considered the problem of minimizing the completion delay in IDNC and formulated it as a stochastic shortest path (SSP) problem. Although this formulated SSP turned out to be impossible to solve in real-time, we employed its properties and structure to design simple maximum weight clique search algorithms, to efficiently reduce the completion delay in IDNC. The designed algorithms were shown to almost achieve the optimal completion delay in wireless broadcast.

One assumption in the algorithms proposed in [1] is that the sender obtains full feedback from all the receivers after each transmission. This assumption is practically infeasible for many network settings, in which feedback cannot be received before several packet transmissions. One important example is time division duplex based cellular networks, in which several packets are sent in each downlink frame before receiving a feedback in the uplink frame. This practical constraint raises the following question: *How can we minimize the completion delay for IDNC in such limited feedback scenario?*

To answer this question, we first extend the SSP formulation of the full feedback scenario in [1] to the limited feedback scenario, by extending the time period of one step to the number of transmissions between two feedbacks. We then show that the resulting SSP formulation is more complicated to solve than the original one but has the same structure and properties. Consequently, we propose to solve the problem for the limited feedback scenario using a similar approach to that employed for the full feedback scenario, after making elimination and update decisions for the attempted but un-acknowledged vertices in the IDNC graph [1]. We then propose four different approaches to make these elimination

978-1-61284-233-2/11/$26.00 ©2011 IEEE

and update decisions for these vertices and compare their performances through extensive simulations.

The rest of the paper is organized as follows. In Section II, we introduce the system model and parameters. The IDNC graph is illustrated in Section III. In Section IV, we present the problem formulation using SSP and compare its complexity and structure to the full feedback formulation. Our proposed modified IDNC selection algorithms are introduced in Section VI and their performances are compared in Section VII. Finally, Section VIII concludes the paper.

II. System Model and Parameters

The model consists of a wireless sender that is required to deliver a frame (denoted by $\mathcal{N}$) of N source packets to a set (denoted by $\mathcal{M}$) of M receivers. The sender initially transmits the N packets of the frame uncoded in an *initial transmission phase*. Each sent packet is subject to erasure at receiver i with probability p_i, which is assumed to be fixed during the frame transmission period. After each T_f packet transmissions, each receiver feeds back to the sender a positive/negative acknowledgement (ACK/NAK) for each received/lost packet during these transmissions. We call this period between two feedback arrivals at the sender by the feedback period and denoted by T_f. By the end of the initial transmission phase, two sets of packets are attributed to each receiver i:

- The *Has* set (denoted by $\mathcal{H}_i$) is defined as the set of packets correctly received by receiver i.
- The *Wants* set (denoted by $\mathcal{W}_i$) is defined as the set of packets that are lost by receiver i in the initial transmission phase of the current broadcast frame. In other words, $\mathcal{W}_i = \mathcal{N} \setminus \mathcal{H}_i$.

The sender stores this information in a *state feedback matrix (SFM)* $\mathbf{F} = [f_{ij}],\ \forall\ i \in \mathcal{M}, j \in \mathcal{N}$, such that $f_{ij} = 0$ if $j \in \mathcal{H}_i$ and $f_{ij} = 1$ if $j \in \mathcal{W}_i$.

After the initial transmission phase, a recovery transmission phase starts. In this phase, the sender exploits the SFM to transmit network coded combinations of source packets. Any receiver having only one missing source packet in each combination can decode this packet. The receivers that receive non-instantly decodable packets discard them. After each T_f transmissions, the receivers send ACK/NAK packets that are used by the sender to update the SFM. This process is repeated until all receivers obtain all the packets. We define the *completion delay* of a frame as the number of recovery transmissions required until all receivers obtain all the packets.

III. IDNC Graph

The IDNC graph is a graph that defines the set of all feasible instantly decodable packet combinations. It was first introduced in the context of a heuristic algorithm design solving the index coding problem [5], [6]. The IDNC graph $\mathcal{G}$ is constructed by first generating a vertex v_{ij} in $\mathcal{G}$ for each packet $j \in \mathcal{W}_i,\ \forall\ i \in \mathcal{M}$. Two vertices v_{ij} and v_{kl} in $\mathcal{G}$ are adjacent if one of the following conditions is true:

- $j = l \Rightarrow$ The two vertices are induced by the loss of the same packet j by two different receivers i and k.
- $j \in \mathcal{H}_k$ and $l \in \mathcal{H}_i \Rightarrow$ The requested packet of each vertex is in the Has set of the receiver that induced the other vertex.

Consequently, each edge between two vertices in the graph represents a coding opportunity that is instantly decodable for the two receivers inducing these vertices. Given this graph formulation, we can easily define the set of all feasible packet combinations in IDNC as the set of packet combinations defined by all cliques in $\mathcal{G}$. Consequently, the sender can generate an IDNC packet for a given transmission by XORing all the packets identified by the vertices of a selected clique κ in $\mathcal{G}$. In the rest of the paper, we say that an IDNC packet *targets* a receiver if its selected clique includes a vertex belonging to this receiver.

IV. Problem Formulation using SSP

A. Problem Formulation

The minimum mean completion delay problem for IDNC can be formulated as an SSP problem as follows.

1) State Space $\mathcal{S}$:
States are defined by all possibilities of SFM $\mathbf{F}(s)$ that may occur during the recovery transmission phase. For state s, the matrix represents the content of the Has and Wants sets in s (i.e. $\mathcal{H}_i(s)$ and $\mathcal{W}_i(s)\ \forall\ i \in \mathcal{M}$). The transition from one state to the following occurs after T_f transmissions, when the sender obtains the feedback from the receivers at the end of the feedback period. Note that the cardinality of the state space is $|\mathcal{S}| = O\left(2^{MN}\right)$.

2) Action Spaces $\mathcal{A}(s)$:
In this problem, actions are taken at the beginning of each feedback period. In any state s, each action a in the action space $\mathcal{A}(s)$ consists of T_f cliques from $\mathcal{G}(s)$, which are employed to generate T_f transmissions to be sent in the next feedback period. Consequently, the action space consists of all T_f clique combinations in the graph $\mathcal{G}(s)$. Note that these cliques may have vertex overlaps as serving a vertex does not guarantee the reception of its packet at its targeted receiver, and thus can be re-attempted. Defining $\mathcal{C}(s)$ as the set of all cliques in $\mathcal{G}(s)$, the cardinality of state s action space $|\mathcal{A}(s)|$ is equal to $\binom{|\mathcal{C}(s)|}{T_f}$.

3) State-Action Transition Probabilities:
To define the state-action transition probability $P_a(s, s')$ for an action $a \in \mathcal{A}(s)$, we define $X_i(s, a)$ as the number of cliques, from the T_f cliques chosen for action a, that include a vertex induced by i. Since each clique can include at most one vertex induced by the same receiver, $X_i(s, a) \leq T_f$ and represents the number of transmissions in action a targeting receiver i. Also let $Y_i(s, s') = |\mathcal{W}_i(s)| - |\mathcal{W}_i(s')|$ and thus $Y_i(s, s') \leq X_i(s, a)$. Consequently, we can express $P_a(s, s')$ as follows:

$$P_a(s, s') = \prod_{i=1}^{M} (1 - p_i)^{Y_i(s,s')}\ p_i^{\left(X_i(s,a) - Y_i(s,s')\right)}. \quad (1)$$

4) State-Action Costs:
Any action taken in the system represents T_f packet transmissions and thus the completion delay is increased by T_f for any taken action. Consequently, in order to minimize the mean completion delay, the cost of all actions in all states should be set to T_f. In other words, $c(s,a) = T_f \ \forall \ a \in \mathcal{A}(s), s \in \mathcal{S}$.

B. SSP Solution Complexity

The optimal policy of an SSP problem can be computed using the famous policy iteration and value iteration algorithms. The complexity of these algorithms are $\Theta\left(|\mathcal{S}|^3 + |\mathcal{S}|^2|\mathcal{A}(s_s)|\right)$ and $\Theta(|\mathcal{S}|^2|\mathcal{A}(s_s))$, respectively, where s_s is the starting state. According to the dimensions of $\mathcal{S}$ and $\mathcal{A}(s)$ described in Section IV-A, we conclude that obtaining the optimal policy is impossible in real-time for typical values of M and N. Also note that the state spaces in both the original and full feedback problems are equal whereas the action space of the limited feedback problem is larger than the original one. This makes the solution of the limited feedback problem more complicated than the original problem.

C. Properties of the SSP Formulation

From the above SSP formulation, we can infer that the problem has the following three properties:

Property 1 (Uniform Cost)**.**
The costs of all actions in all states are all the same except for the absorbing state.

Property 2 (Non-singleton acyclicity)**.**
No state can be revisited once the process moves to a next state, and thus the SSP formulation is acyclic. Indeed, if some packets are received by some receivers, there is no means of going back with these receivers not having them. However, a state can revisit itself (singleton cycles) if none of the targeted receivers by an action receives the IDNC packets.

Property 3 (Non-increasing successor value functions)**.**
Due to the non-singleton acyclicity property, the successor states of a given state are all closer to the absorbing state than itself. Consequently, the expected cost to absorption at a given state is always greater than or equal to the expected costs to absorption at all its successor states.

These three properties are the same properties found in the original full feedback problem, as described in [1]. Consequently, an approach similar to that used to minimize the completion delay in the full feedback problem will also minimize it for the limited feedback problem. In other words, the completion delay in the limited feedback problem will be minimized if the sender gives more priority to targeting the receivers with larger values of $\psi_i \triangleq \frac{|\mathcal{W}_i|}{1-p_i}$, which represents the expected individual completion delay of receiver i. In the full feedback problem, this prioritization is implemented by assigning a weight equal to ψ_i to each vertex v_{ij} in the IDNC graph [1]. For more prioritization control, we can generalize the vertex weights to be ψ_i^n, where n determines the degree of bias given to receivers with higher expected individual completion delays. The set of targeted receivers for each transmission can then be determined by running a maximum weight clique search over this weighed IDNC graph.

The only difficulty to directly apply this same approach for the limited feedback problem is that the sender should perform sequential clique selections for T_f transmissions without knowing the outcome of the attempted and un-acknowledged transmissions within the feedback period. Consequently, to be capable of applying the efficient full feedback algorithm to the limited feedback problem, the sender needs to make decisions about the status of the vertices and the graph, after each transmission, without feedback. In the following sections, we introduce four different approaches to deal with these updates and introduce the modified algorithms accordingly.

V. Vertex and Graph Update Approaches

In this section, we present four vertex elimination approaches that could be applied to the IDNC graph, after each un-acknowledged transmission within the feedback period. These approaches will help turning the problem of jointly selecting T_f cliques into sequential clique selection, using the algorithms proposed in [1], after vertex and/or graph update.

A. Full Vertex Elimination (FVE)

In this approach, all attempted vertices in each transmission within the feedback period are eliminated from the graph. Consequently, the sender will select T_f fully disjoint cliques for the T_f transmissions. Each of the T_f cliques is chosen using the algorithms in [1] after the elimination of all the vertices of the previously selected cliques. At the feedback instants, the sender adjusts its graph according to the received feedback and then re-executes the above process for the following feedback period. In case there are no leftover vertices in the graph while the system is still within the feedback period, the sender retransmits the previously generated cliques, after the last feedback instant, in the remaining transmissions until the arrival of the new feedback.

The advantage of this approach is that it guarantees full innovation for the targeted receivers in each transmission due to the use of disjoint cliques. Retransmissions of the same vertices occurs only when the system is already near completion and have actually served most of the vertices.

B. Full Vertex Elimination with Graph Update (FVE-GU)

This approach is a variant of the previous approach, in which not only all attempted vertices are eliminated but also the sender updates the vertex adjacency, assuming all attempted vertices are correctly received at their intended receivers. With this assumption, the new coding opportunities, arising from the packet reception in the previous transmission, are reflected in the graph. This may result in better and larger clique selections in the subsequent transmissions, which may target more receivers and help the system reach completion faster. However, this approach risks of generating non-instantly decodable packets for the receivers that were falsely assumed to have decoded some packets. Nonetheless, we still consider this approach to study the effect of the above trade-off.

C. Stochastic Vertex Elimination (SVE)

In this approach, attempted vertices are eliminated from the graph probabilistically according to the erasure probabilities of their inducing receivers. When a vertex v_{ij} is attempted in a transmission within the feedback period, the sender keeps this vertex in the graph with probability p_i and eliminates it with probability $1 - p_i$. Similar to FVE, the sender adjusts its graph at each feedback instant and re-executes the above process for the following feedback period. In case there are no leftover vertices in the graph while the system is still within the feedback period, the sender retransmits the previously generated cliques, after the last feedback instant, in the remaining transmissions until the arrival of the new feedback.

This approach better represents the reception probability of packets compared to FVE and thus can re-attempt the non-received vertices in an earlier stage compared to FVE. However, it also falls in the risk of re-attempting received vertices in case the sender decides to keep them in the graph. Again this trade-off may or may not result in a better completion delay.

D. Stochastic Vertex Elimination with Graph Update (SVE-GU)

This approach is a variant of SVE in which the sender updates the graph adjacency assuming the correct reception of the stochastically eliminated vertices in SVE. In this case, both trade-offs of FVE-GU and SVE are attempted to test whether they will result in a better completion delay.

VI. Proposed Algorithm

For the full feedback problem, an efficient IDNC packet selection in state s can be performed by a maximum weight clique selection from the state's IDNC graph $\mathcal{G}(s)$, where the graph vertices are weighted with a power of the expected individual completion delays of their inducing receivers. Given the vertex and graph update approaches, described in Section V, we can apply the same algorithm to generate efficient coded transmissions in limited feedback scenarios. In this case, a maximum weight clique is selected, its vertices are eliminated from the graph according to one of the four approaches described above, then the subsequent maximum weight clique is selected and so on. This will sequentially generate T_f cliques for the transmissions within the feedback periods. In the rest of the paper, we define *virtual states* as states representing the assumed graph status after using a vertex and graph update approach within the feedback period.

The maximum weight clique selection algorithm is known to be NP-hard but can be exactly solved in polynomial time for non-large graphs [7]. However, this complexity may still be prohibitive in large networks. Consequently, we can replace, in the above process, the maximum weight clique selection with the simpler maximum weight vertex search algorithm, proposed in [1]. In this search, the weights of the vertices are defined as follows. We first define $a_{ij,kl}(s)$ as the adjacency indicator of vertices v_{ij} and v_{kl} in $\mathcal{G}(s)$ such that:

$$a_{ij,kl}(s) = \begin{cases} 1 & v_{ij} \text{ is adjacent to } v_{kl} \text{ in } \mathcal{G}(s) \\ 0 & \text{otherwise .} \end{cases} \quad (2)$$

We then define the weighted degree $\Delta_{ij}(s)$ of vertex v_{ij} as:

$$\Delta_{ij}(s) = \sum_{\forall v_{kl} \in \mathcal{G}(s)} a_{ij,kl}(s)\, \psi_k^n(s) \; . \quad (3)$$

Thus, a large weighted vertex degree reflects its adjacency to a large number of vertices belonging to receivers with large values of $\psi_i(s)$. We finally define the vertex weight $w_{ij}(s)$ as:

$$w_{ij}(s) = \psi_i^n(s)\, \Delta_{ij}(s) \; . \quad (4)$$

Based on these definitions, the maximum weight vertex search algorithm can be described as follows for the limited feedback problem. In each visited virtual or actual state s, the algorithm computes a maximal clique $\kappa^*(s)$ in $\mathcal{G}(s)$. At first, $\kappa^*(s)$ is an empty set. The algorithm starts by selecting the maximum weight vertex in $\mathcal{G}(s)$ to be the source vertex in $\kappa^*(s)$. For each of the following iterations, the algorithm first recomputes the new vertex weights, within the subgraph adjacent to all previously selected vertices in $\kappa^*(s)$, then adds the new maximum weight vertex to it. The algorithm stops when there is no further vertex adjacent to all vertices in $\kappa^*(s)$. Once the clique is computed, the sender forms and sends an IDNC packet by XORing the source packets identified by the vertices in $\kappa^*(s)$. If the system is within the feedback period, the sender runs the selected vertex and graph update approach to reach a new virtual state. At feedback instants, the sender makes the correct adjustments of the graph, according to the received feedback, to reach a new actual state. In both cases, the process is re-executed until the absorbing state is reached.

VII. Simulation Results

In this section, we compare through extensive simulations the performance of the four vertex and graph update approaches with both the optimal maximum weight clique selection algorithm (denoted by "opt") and the simpler maximum weight vertex search heuristic (denoted by "srh"). We also compare the performance of these approaches to those of the full feedback (FF) IDNC algorithms. In our simulations, we assume that the packet erasure probabilities of different receivers change from frame to frame in the range [0.01,0.3], while maintaining the average erasure probability in the network at 15%.

Figure 1 depicts the comparison of the average completion delays achieved by the different algorithms against T_f, for $M = 60$ and $N = 30$. Figure 2 depicts the comparison of the average completion delays achieved by the different algorithms against M, for $N = 30$ and $T_f = 5$. Finally, Figure 3 depicts the same comparison against N for $M = 60$ and $T_f = 5$.

From all three figures, we can see that the FVE approach achieves the best completion delay performance compared to the other three approaches for both the optimal and search clique selection algorithms. This clearly shows that the differ-

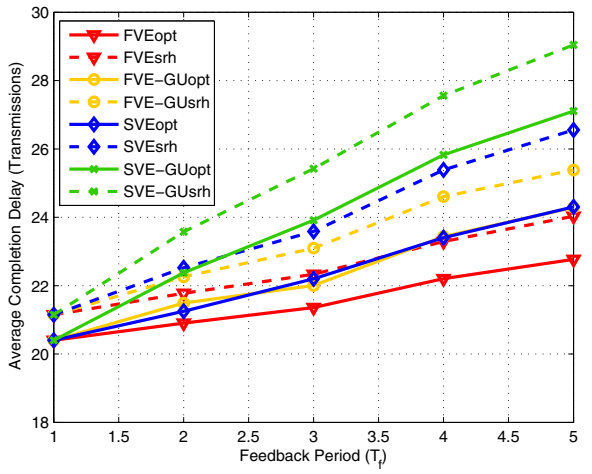

Fig. 1. Comparison of Average Completion Delays against T_f

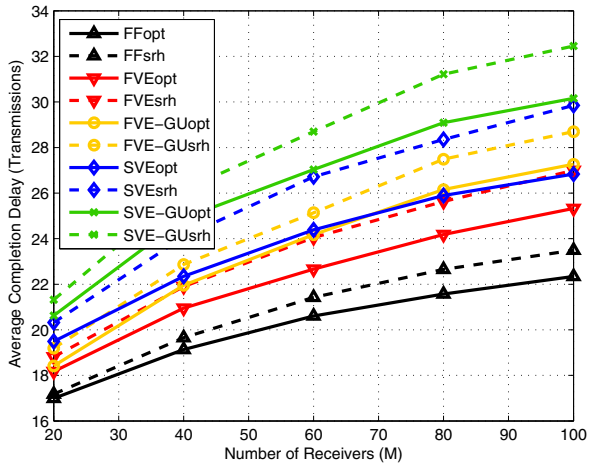

Fig. 2. Comparison of Average Completion Delays against M

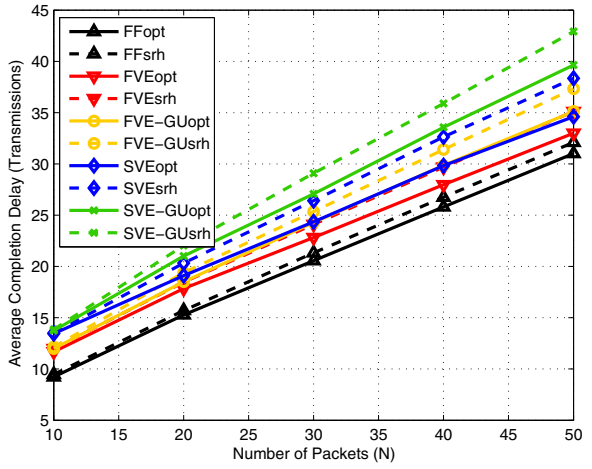

Fig. 3. Comparison of Average Completion Delays against N

ent trade-offs introduced in the other three approaches tend to degrade the performance rather than improving it.

Another important observation is the degradation in average completion delay obtained in the limited feedback scenario compared to the full feedback scenario, which naturally increases with the increase of the feedback period. However, for a relatively large network setting ($M = 100$, $N = 30$) and a considerable feedback period value ($T_f = 5$), this degradation reaches 3 and 3.5 transmissions for the FVEopt and FVEsrh algorithms, respectively, compared to their corresponding full feedback algorithms (depicted in Figure 2). This results in an overall degradation in the frame delivery duration (from the start of the frame transmission until its reception at all receivers) of 6% and 6.5% for FVEopt and FVEsrh, respectively. These values are clearly tolerable given the 80% reduction in the feedback frequency required for the algorithms' operation, a reduction that is of extreme importance in many practical network settings as explained in Section I.

VIII. Conclusion

In this paper, we considered the problem of minimizing the broadcast completion delay for IDNC with limited feedback. We first extended the SSP formulation of the full feedback problem to the limited feedback problem and showed it is more complicated to solve. We then showed that the limited feedback formulation has the same structure as the original formulation. We thus proposed the use of a similar approach to that used for full feedback after making vertex elimination and graph update decisions for the un-acknowledged transmissions. Simulation results showed that full vertex elimination with no graph update achieves the best performance compared to other elimination and update approaches. It also achieves a tolerable degradation compared to the full feedback performance while using a much lower feedback frequency.

References

[1] S. Sorour and S. Valaee, "On minimizing broadcast completion delay for instantly decodable network coding," *IEEE International Conference on Communications (ICC'10)*, May 2010.

[2] P. Sadeghi, D. Traskov, and R. Koetter, "Adaptive network coding for broadcast channels," *Fifth Workshop on Network Coding, Theory and Applications (NetCod'09)*, pp. 1–6, June 2009.

[3] P. Sadeghi, R. Shams, and D. Traskov, "An optimal adaptive network coding scheme for minimizing decoding delay in broadcast erasure channels," *EURASIP Journal of Wireless Communations and Networking*, pp. 1–14, April 2010.

[4] S. Sorour and S. Valaee, "Minimum broadcast decoding delay for generalized instantly decodable network coding," *Accepted for publication in IEEE Global Telecommunications Conference (GLOBECOM'10)*, 2010.

[5] M. Chaudhry and A. Sprintson, "Efficient algorithms for index coding," *IEEE Conference on Computer Communications Workshops (INFOCOM'08)*, pp. 1–4, April 2008.

[6] S. El Rouayheb, M. Chaudhry, and A. Sprintson, "On the minimum number of transmissions in single-hop wireless coding networks," *IEEE Information Theory Workshop (ITW'07)*, pp. 120–125, Sept. 2007.

[7] K. Yamaguchi and S. Masuda, "A new exact algorithm for the maximum weight clique problem," *23rd International Conference on Circuits/Systems, Computers and Communictions (ITC-CSCC'08)*, 2008.