On Minimizing Decoding Complexity for Binary Linear Network Codes

Jian Wang(✉), Kui Xu, Xiaoqin Yang, Lihua Chen, Wei Xie, and Jianhui Xu

Army Engineering University of PLA, Nanjing 210007, China
farfly217@163.com, lgdxxukui@126.com, 15261856573@139.com, clhtyue@163.com, edifier77@163.com, xujianhui900118@163.com

Abstract. The typical method adopted in the decoding of linear network codes is Gaussian Elimination (GE), which enjoys extreme low policy complexity in determining actions, i.e., the XORing operation executed upon the decoding matrix and the coded packets. However, the amount of the total required actions is quite large, which makes the overall decoding complexity high. In this paper, we consider the problem of minimizing the decoding complexity of binary linear network codes. We formulate the decoding problem into a special shortest path problem where the weight of each edge consists of: (1) a const weight due to the execution of the action; (2) a variable weight due to the adopted policy in determining the action. The policy is formulated as an optimization problem that minimizes a particular objective function by enumerating over a certain action set. Since finding the optimal policy is intractable, we optimize the policy in dual directions. At one hand, we guarantee the objective function and the action set are similar to the optimal policy that minimizes const weight summation; at the other hand, we guarantee that the objective function have simple structure and the action set is small, so that the variable weight summation is also small. Simulation results demonstrate that our proposed policy can significantly reduce the decoding complexity compared with existing methods.

Keywords: Binary linear network codes · Decoding complexity

1 Introduction

In an erasure broadcast channel, a sender needs to send N packets reliably to M receivers. In each transmission, each receiver either receives a packet exactly or experiences an erasure. For this problem, linear network codes [1] has been demonstrated as a promising solution, where each coded packet is the linear combination of the N source packets over the finite field $GF(q)$. We refer to the coefficients of the linear combination as the encoding vector. A coded packet is

This work is supported by Jiangsu Province Natural Science Foundation (BK20160079), National Natural Science Foundation of China (No. 61671472, No. 61501511, No. 61771486).

© ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2019
Published by Springer Nature Switzerland AG 2019. All Rights Reserved
M. Jia et al. (Eds.): WiSATS 2019, LNICST 281, pp. 576–586, 2019.
https://doi.org/10.1007/978-3-030-19156-6_54

said to be *innovative* to a receiver if its encoding vector is not in the subspace spanned by the encoding vectors of packets already received. Once a receiver has collected N innovative packets, the N source packets can be recovered through decoding.

Linear network codes can be generated with or without feedback. LT codes [2], Raptor codes [3] and Random Linear Network Codes (RLNC) [4] can be used without feedback. For these codes, coded packets are with certain probabilities to be innovative. With feedback, the design of the coded packets are more targeted [5,6]. It is indicated in [5] that an innovative packet to all receivers can always be found if $q \geq M$. As a result, the *completion time*, measured in terms of the number of packet transmissions, is minimized. For a broadcast system, to obtain a near optimal completion time performance, the above condition can be further relaxed, i.e., it is unnecessary for each coded packet to be innovative to all receivers. We only need to make sure the coded packet is innovative to the receivers with least innovative packets. The simulation results in [6] demonstrate that almost optimal completion time performance can be obtained with $q = 2$.

The excellent performance of linear network codes encourages researchers to consider its practicality. In particular, decoding complexity is an important issue, as mobile devices typically have low computation speed and limited energy. Despite of its excellent completion time performance, it is quite nature to consider adopting $q = 2$, i.e., to generate network codes over the finite field $GF(2)$, due to two obvious benefits. First, the adoption of $GF(2)$ avoids the complex multiplication calculations over the finite field. In traditional RLNC, at most $\frac{N^3}{3} + N^2L$ addition calculations and $\frac{N^3}{3} + N^2L$ multiplication calculations over $GF(q)$ are required with Gaussian Elimination (GE) decoding, where L denotes the length of the packets. Since multiplication calculations are much more complex than addition calculations, $GF(2)$ has incomparable advantage in reducing decoding complexity. Second, $GF(2)$ naturally lead to the sparsity of encoding vectors, which can obviously reduce decoding complexity. In traditional RLNC, each coded packet is the linear combination of all source packets. One important way to reduce the decoding complexity of RLNC is to introduce sparse encoding, i.e., to courage coded packets to contain less source packets [7].

The decoding of binary linear network codes is equivalent to solving sparse linear equations over $GF(2)$ as well as implementing corresponding packets addition. Matrix-free methods such as Wiedemann's coordinate recurrence algorithm [8] and the Lanczos algorithm [9] have smaller complexity compared with GE based method, but the superiority can only be realized when N is large and L is small. However, in practice, N is dozens while L is several thousands or even ten thousands, that means, $N \ll L$. As a result, GE based methods are more suitable for the decoding of binary linear network codes. As far as we know, only few GE based algorithms [6,10–12] have been designed, and the decoding problem had not been formally addressed. In this paper, we address the problem of minimizing the decoding complexity of binary network codes using GE based method. The contributions can be summarized as follows.

First, we formally formulate the decoding problem of binary linear network codes into a special shortest path problem, where all possible states of the decoding matrix together with the coded packets are vertices, and a two-way directed edge between a pair of vertices exists if there is a certain row-adding action can lead to the transformation between them. In traditional shortest path problems, the weight of each edge is fixed, thus to find the path with shortest path is equivalent to finding a sequence of actions that lead to the path with minimum weight summation. However, in our formulated problem, the weight of a edge is related to the adopted policy. In particular, the weight of a edge consists of two components, i.e., a const weight due to the execution of the action upon the decoding matrix as well as the corresponding coded packets, and a variable weight due to the application of the policy in determining the action. In this case, to solve the problem is in fact to find the optimal policy, so that the weight summation, i.e., the addition of the const weight summation and the variable weight summation, can be minimized.

Second, we formulate the optimality equation in terms of the const weight summation and show that the optimal const weight summation function is intractable, which demonstrates that the optimal policy in terms of the weight summation is also intractable. To obtain a good policy, we optimize the policy in dual directions, at one hand, we grasp the main monotonic characteristic of the optimal const weight summation function to make sure the objective function has similar characteristic, and action set contain important actions; at the other hand, we guarantee that the objective function is with simple structure and the cardinality of the action set is small.

Finally, we conduct simulations of our proposed policy for decoding. Simulation results demonstrate that compared with existing decoding methods, our proposed policy can significantly reduce the overall decoding complexity.

2 Decoding Problem Formulation

Consider a broadcasting scenario where a sender wants to deliver N source packets $\{\mathbf{p}_n\}_{n=1}^{N}$ to multiple receivers. Each source packet contains L bits. A transmitted packet is generated as the linear combination of a subset of all source packets over the finite field $GF(2)$. By convenience, we use an N-dimensional vector called *encoding vector* to present a coded packet, where the n-th source packet is contained in the coded packet if the n-th element of encoding vector equals 1, otherwise the n-th element equals 0. After collecting N innovative coded packets $\{\mathbf{q}_n\}_{n=1}^{N}$, each receiver maintains a full-rank binary $N \times N$ *decoding matrix* $\mathbf{S}$ with n-th row being the encoding vector of $\mathbf{q}_n$. A sequence of row-adding actions upon the decoding matrix over $GF(2)$ together with bitwise adding actions upon the corresponding coded packets over $GF(2)$ are executed until each row (or column) of the decoding matrix contains only one non-zero element, and all source packets are recovered. Since the row-adding actions upon the decoding matrix and the bitwise addition actions upon the coded packets are implemented simultaneously, throughout the paper, we only mention the row-adding actions for simplicity.

We add a subscript to $\mathbf{S}$ to represent the row-adding action procedure. In particular, we assume that T row-adding actions are required, and the initial decoding matrix and the target decoding matrix are denoted as $\mathbf{S}_1$ and $\mathbf{S}_{T+1}$ respectively. For a particular $\mathbf{S}_t, 1 \leq t \leq T$, a row-adding action $u_t = (i_{\mathbf{S}_t}, j_{\mathbf{S}_t})$ can be selected from the action set

$$\mathcal{U} = \{(i,j)|i,j \in [N] = \{1,2,\cdots,N\}, i \neq j\}.$$

After executing u_t, the decoding matrix is transformed from $\mathbf{S}_t$ to $\mathbf{S}_{t+1}$. In particular, we have

$$\mathbf{S}_{t+1}^{(m,n)} = \begin{cases} \mathbf{S}_t^{(i_{\mathbf{S}_t},n)} \oplus \mathbf{S}_t^{(j_{\mathbf{S}_t},n)} & m = j_{\mathbf{S}_t} \\ \mathbf{S}_t^{(m,n)} & m \in [M] \backslash \{j_{\mathbf{S}_t}\} \end{cases}$$

where $\mathbf{S}_t^{(m,n)}$ denotes the element in m-th row and n-th column of $\mathbf{S}_t$, and $\oplus$ denotes the adding operation over $GF(2)$. For the purpose of emphasizing the action u_t applied at $\mathbf{S}_t$, we also use $\mathbf{S}_{u_t}$ interchangeably with $\mathbf{S}_{t+1}$. Figure 1 illustrates the decoding procedure given an initial decoding matrix.

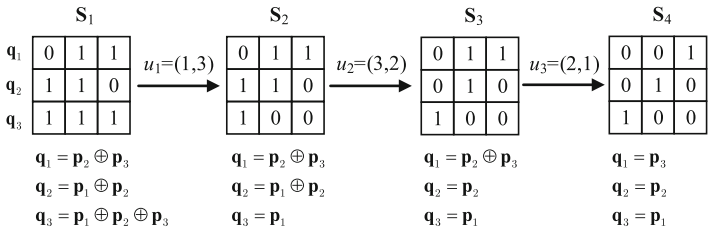

Fig. 1. An illustration of the decoding procedure given an initial decoding matrix $\mathbf{S}_1$.

According to the decoding procedure at each receiver, a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ can be constructed, where $\mathcal{V}$ is the set of the vertices corresponding to all possible states of the decoding matrix, and $\mathcal{E}$ is the set of edges that describes the adjacency relations of different vertices. In particular, the edge from $\mathbf{S}$ to $\mathbf{S}'$ exists if $\exists u \in \mathcal{U}$ so that $\mathbf{S}$ is transformed into $\mathbf{S}'$ after executing u, i.e., $\mathbf{S}' = \mathbf{S}_u$. Given an arbitrary initial vertex $\mathbf{S}_1$, we need to find a path to a target vertex, so that the weight summation of the path is minimized. A vertex is said to be a target vertex if each row (or column) of the corresponding decoding matrix contains only one non-zero element, and the set of all target vertices is denoted as $\mathcal{V}_\mathcal{T}$. The weight summation of a path refers to the summation of weights of edges in this path.

In traditional shortest path problems, each edge has a fixed weight, thus to find the shortest path is in fact to find a best action sequence. However, in this decoding problem, the weight of each edge consists of two components. The first weight component is a const weight due to the execution of the action upon

the decoding matrix and the coded packets. Since the action executed upon the decoding matrix and the coded packets is the binary addition of a pair of N-dimensional vectors and a pair of L bit data sequence respectively, it can simply be expressed as

$$w_c(\mathbf{S}) = N + L \tag{1}$$

and the basic operation is integer addition. The second weight component is a variable weight $w_v(\mathbf{S})$ due to the execution of the policy in determining the action. The definition of a policy is presented as follows.

Definition 1. *A policy $\mu : \mathcal{V}\backslash\mathcal{V}_\mathcal{T} \rightarrow \mathcal{U}$ is defined as a function that maps an arbitrary vertex $\boldsymbol{S} \in \mathcal{V}\backslash\mathcal{V}_\mathcal{T}$ into a particular action $u \in \mathcal{U}$, that is $u = \mu(\boldsymbol{S})$. Note that the shortest path problem is deterministic, when the initial vertex $\boldsymbol{S}_1$ is determined, a policy will lead to a sequence of actions, i.e., $u_1, u_2, \cdots$.*

Since the available actions are discrete and the effectiveness of actions are non-smooth, a policy can only be constructed as the enumeration over a certain action set $\mathcal{Q}(\mathbf{S}) \subseteq \mathcal{U}$ in minimizing a certain objective function $\mathcal{A}(\mathbf{S}, u)$, that is

$$\mu : \arg\min_{u \in \mathcal{Q}(\mathbf{S})} \mathcal{A}(\mathbf{S}, u). \tag{2}$$

The explicit value of $w_v(\mathbf{S})$ depends on the computational complexity of the objective function $\mathcal{A}(\mathbf{S}, u)$ and the cardinality of the action set $\mathcal{Q}(\mathbf{S})$. We denote the computation amount of the objective function $\mathcal{A}(\mathbf{S}, u)$ as $\mathcal{C}(\mathcal{A}(\mathbf{S}, u))$, thus the variable weight can be expressed as

$$w_v(\mathbf{S}) = \sum_{u \in \mathcal{Q}(\mathbf{S})} \mathcal{C}(\mathcal{A}(\mathbf{S}, u)). \tag{3}$$

We assume that for a particular policy μ, the length of the path is T_μ, thus the decoding problem can be formulated as finding the optimal policy μ^*, i.e., to design $\mathcal{A}(\mathbf{S}, u)$ and $\mathcal{Q}(\mathbf{S})$ so that the weight summation can be minimized, that is,

$$\arg\min_{\mu} V_\mu(\mathbf{S}_1) = \sum_{t=1}^{T_\mu} [w_c(\mathbf{S}_t) + w_v(\mathbf{S}_t)]. \tag{4}$$

3 Policy Optimization

$V_\mu(\mathbf{S}_1)$ can be divided into two parts, i.e., the const weight summation $V_\mu^c(\mathbf{S}_1) = \sum_{t=1}^{T_\mu} w_c(\mathbf{S}_t)$ and the variable weight summation $V_\mu^v(\mathbf{S}_1) = \sum_{t=1}^{T_\mu} w_v(\mathbf{S}_t)$. The const weight summation is the unavoidable cost in solving the shortest path problem. It refers to the costs of executing the actions following a path, which can be regarded as the theoretical lower bound of the weight summation. Since $w_c(\mathbf{S}_t)$ is const, $V_\mu^c(\mathbf{S}_1)$ is linearly proportional to T_μ, thus to minimize $V_\mu^c(\mathbf{S}_1)$ is in fact to minimize T_μ. While the variable weight summation refers to the extra cost that

is produced in the procedure of determining the path, which can be minimized, or even eliminated by adopting simple algorithm such as GE algorithm. To minimize $V_{\mu}^{v}(\mathbf{S}_1)$, we need to minimize the variable weight $w_v(\mathbf{S})$ and the path length T_{μ} simultaneously. Thus, in order to minimize $V_{\mu}(\mathbf{S}_1)$, we should minimize $w_v(\mathbf{S})$ and T_{μ} simultaneously.

Unfortunately, minimizing $w_v(\mathbf{S})$ and minimizing T_{μ} are contradicted with each other. According to (3), the computational complexity of $w_v(\mathbf{S})$ is linearly proportional to the cardinality of $\mathcal{Q}(\mathbf{S})$, i.e., $|\mathcal{Q}(\mathbf{S})|$, and the computational amount of $\mathcal{A}(\mathbf{S}, u)$. Thus to minimize $w_v(\mathbf{S})$, we should make $\mathcal{A}(\mathbf{S}, u)$ as simple as possible and the cardinality of $\mathcal{Q}(\mathbf{S})$ as small as possible. Specially, if $\mathcal{A}(\mathbf{S}, u) = 1$ or $|\mathcal{Q}(\mathbf{S})| = 1$, we have $w_v(\mathbf{S}) = 0$. However, an obvious fact is that the more simple structure $\mathcal{A}(\mathbf{S}, u)$ and $\mathcal{Q}(\mathbf{S})$ have, the more likely that the produced actions are close to random selection, thus the longer the length of the path will be. Due to this reason, we can image that GE algorithm which enjoys zero variable weight will suffer from the a long path.

Tn order to minimize T_{μ}, we should guarantee policy μ has similar property with the optimal policy μ_c^* that can minimize $V_{\mu}^{c}(\mathbf{S}_1)$. Note that the optimality equations in terms of $V_{\mu_c^*}^{c}(\mathbf{S}_t)$ can be constructed as

$$V_{\mu_c^*}^{c}(\mathbf{S}_t) = \min_{u_t \in \mathcal{U}} \left[V_{\mu_c^*}^{c}(\mathbf{S}_{u_t}) + w_c(\mathbf{S}_t) \right], t \in [T_{\mu_c^*}]. \tag{5}$$

Thus policy μ_c^* can be expressed as

$$\mu_c^* : \arg\min_{u \in \mathcal{U}} \ V_{\mu_c^*}^{c}(\mathbf{S}_u). \tag{6}$$

Obviously, to achieve minimum T_{μ}, we should set $\mathcal{A}(\mathbf{S}, u) = V_{\mu_c^*}^{c}(\mathbf{S}_u)$ and $\mathcal{Q}(\mathbf{S}) = \mathcal{U}$. Since there exists a one-to-one correspondence between $\mathbf{S}$ and an N^2-dimensional binary vector, $V_{\mu_c^*}^{c}(\mathbf{S}_u)$ is in fact a pseudo-Boolean function [15], which can be expressed as

$$V_{\mu_c^*}^{c}(\mathbf{S}_u) = \sum_{\mathcal{P} \subseteq \{(m,n) | m \in [N], n \in [N]\}} a_{\mathcal{P}} \prod_{(m,n) \in \mathcal{P}} \mathbf{S}_u^{(m,n)}. \tag{7}$$

Even if we can determine all the coefficients $\alpha_{\mathcal{P}}$, the computation of $V_{\mu_c^*}^{c}(\cdot)$ are with exponential complexity. Consequently, $w_v(\mathbf{S})$ becomes large.

Due to the contradiction demonstrated as above, the optimization over the policy can be implemented by balancing the simplicity and similarity to μ_c^*. At one hand, we should make sure that the expression of $\mathcal{A}(\mathbf{S}, u)$ is simple and the cardinality of $\mathcal{Q}(\mathbf{S})$ is small, thus the variable weight $w_v(\mathbf{S})$ becomes small. At the other hand, the adopted $\mathcal{A}(\mathbf{S}, u)$ should grasp the main characteristics of $V_{\mu_c^*}^{c}(\mathbf{S}_u)$ and $\mathcal{Q}(\mathbf{S})$ should contain the best actions in $\mathcal{U}$ that can minimize $V_{\mu_c^*}^{c}(\mathbf{S}_u)$, so that T_{μ} can be minimized.

3.1 Objective Function Determination

To guarantee the effectiveness of the objective function, $\mathcal{A}(\mathbf{S}, u)$ should reflect the main characteristic of $V_{\mu_c^*}^{c}(\mathbf{S})$, at the same time, it should have simple structure. As presented by (7), $V_{\mu_c^*}^{c}(\mathbf{S}_u)$ is a multilinear function in 0–1 variables, an

effective method of simplifying it is to adopt its linear terms as the objective function, i.e.,

$$\mathcal{A}(\mathbf{S}, u) = \sum_{m\in[N], n\in[N]} a_{(m,n)} \mathbf{S}_u^{(m,n)} \tag{8}$$

where $u = (i_{\mathbf{S}}, j_{\mathbf{S}})$ and

$$\mathbf{S}_u^{(m,n)} = \begin{cases} \mathbf{S}^{(i_{\mathbf{S}},n)} \oplus \mathbf{S}^{(j_{\mathbf{S}},n)} & m = j_{\mathbf{S}} \\ \mathbf{S}^{(m,n)} & m \in [M] \backslash \{j_{\mathbf{S}}\} \end{cases} \tag{9}$$

When we set $a_{(m,n)} = 1$, the objective function represents the number of remaining non-zero elements after executing action u. In this situation, minimizing the objective function is equivalent to maximizing the number of the eliminated non-zero elements. Note that our target is to reduce the number of non-zero element to N as quickly as possible, this objective function is positively related to our target. From the perspective of dynamic programming [14], the objective function with $a_{(m,n)} = 1$ can be regarded as the myopia policy, meaning that it only selects the action that can maximize the number of the eliminated non-zero elements, despite of the risk that successor vertex may not have good actions. The existence of coefficients $a_{m,n}$ is to allocate priorities to different elements, so that we can achieve a tradeoff of between the short-term interest (maximizing the number of the eliminated non-zero elements) and the long-term interest (transforming to the vertices that are more likely to have actions that can eliminate more non-zero elements). The following theorem gives us insight in designing $a_{(m,n)}$.

Theorem 1. *When the number of non-zero elements is fixed, a vertex with a larger variance of feature components is expected to eliminate more non-zero elements with a randomly selected action.*

Proof. We assume two vertices $\mathbf{S}$, $\mathbf{S}'$ that have same number of non-zero elements. Their feature vectors are expressed as $\boldsymbol{\phi}_{\mathbf{S}} = [\boldsymbol{\phi}_{\mathbf{S}}^{(1)}, \cdots, \boldsymbol{\phi}_{\mathbf{S}}^{(n)}, \cdots, \boldsymbol{\phi}_{\mathbf{S}}^{(N)}]$ and $\boldsymbol{\phi}_{\mathbf{S}'} = [\boldsymbol{\phi}_{\mathbf{S}'}^{(1)}, \cdots, \boldsymbol{\phi}_{\mathbf{S}'}^{(n)}, \cdots, \boldsymbol{\phi}_{\mathbf{S}'}^{(N)}]$ respectively, with $\sum_{n=1}^{N} \boldsymbol{\phi}_{\mathbf{S}}^{(n)} = \sum_{n=1}^{N} \boldsymbol{\phi}_{\mathbf{S}'}^{(n)} = K$. Without loss of generality, we assume that the variance of $\boldsymbol{\phi}_{\mathbf{S}}$ is smaller than $\boldsymbol{\phi}_{\mathbf{S}'}$, i.e.,

$$\sum_{n=1}^{N} \left(\boldsymbol{\phi}_{\mathbf{S}}^{(n)} - \frac{K}{N} \right)^2 < \sum_{n=1}^{N} \left(\boldsymbol{\phi}_{\mathbf{S}'}^{(n)} - \frac{K}{N} \right)^2$$

which can be further simplified as

$$\sum_{n=1}^{N} {\boldsymbol{\phi}_{\mathbf{S}}^{(n)}}^2 < \sum_{n=1}^{N} {\boldsymbol{\phi}_{\mathbf{S}'}^{(n)}}^2. \tag{10}$$

When a random action is selected, the expected eliminated non-zero element in n-th column of $\mathbf{S}$ is

$$\frac{\binom{\phi_{\mathbf{S}}^{(n)}}{2} - \phi_{\mathbf{S}}^{(n)}(N - \phi_{\mathbf{S}}^{(n)})}{\binom{N}{2}} = \frac{3{\phi_{\mathbf{S}}^{(n)}}^2 - (2N+1)\phi_{\mathbf{S}}^{(n)}}{N(N-1)}$$

and the expected number of the eliminated non-zero elements in $\mathbf{S}$ is

$$E_{\mathbf{S}} = \sum_{n=1}^{N} \frac{3{\phi_{\mathbf{S}}^{(n)}}^2 - (2N+1)\phi_{\mathbf{S}}^{(n)}}{N(N-1)} = \sum_{n=1}^{N} \frac{3{\phi_{\mathbf{S}}^{(n)}}^2}{N(N-1)} - \frac{(2N+1)K}{N(N-1)}.$$

Similarly, the expected number of the eliminated non-zero elements in $\mathbf{S}'$ is

$$E_{\mathbf{S}'} = \sum_{n=1}^{N} \frac{3{\phi_{\mathbf{S}'}^{(n)}}^2}{N(N-1)} - \frac{(2N+1)K}{N(N-1)}.$$

According to (10), we conclude that $E_{\mathbf{S}} < E_{\mathbf{S}'}$. The theorem is proved. □

According to Theorem 3, when we allocate higher priorities to the non-zero elements corresponding to small feature components, i.e., set $a_{(m,n)}$ negative to $\phi_{\mathbf{S}}^{(n)}$, we are more likely to transform into a better successor vertex, and a shorter path can be achieved.

Note that only the elements in $j_{\mathbf{S}}$-th row of $\mathbf{S}$ changes after executing u, we can ignore the common expression components in (9), and (9) can be simplified as

$$\begin{aligned} \mathcal{A}(\mathbf{S}, u) &= \sum_{m\in[N], n\in[N]} a_{(m,n)} \mathbf{S}_u^{(m,n)} - \sum_{m\in[N], n\in[N]} a_{(m,n)} \mathbf{S}^{(m,n)} \\ &= \sum_{n\in[N]} a_{(j_{\mathbf{S}},n)} \left[\mathbf{S}_u^{(j_{\mathbf{S}},n)} - \mathbf{S}^{(j_{\mathbf{S}},n)} \right]. \end{aligned} \tag{11}$$

3.2 Action Set Determination

The purpose of designing the action set is to select a small subset of actions from the whole action set $\mathcal{U}$, with the guarantee that the effectiveness of best action selected from the subset is not too far from the best action selected from $\mathcal{U}$ in minimizing $V_{\mu_c^*}^c(\mathbf{S}_u)$. Note that the purpose of executing an action is to eliminate non-zero elements, thus the basic requirement for an action to be effective is that at least one non-zero element is changed to zero. Accordingly, we can naturally construct in total N basic action sets according to the position of the changed non-zero element. In particular, the n-th basic action set $\mathcal{C}_n$ can be constructed as

$$\mathcal{C}_n = \{(i,j) | \mathbf{S}^{(i,n)} = \mathbf{S}^{(j,n)} = 1, i, j \in [N]\} \tag{12}$$

Obviously, any action selected from $\mathcal{C}_n$ can lead to the elimination of a non-zero element in n-th column of $\mathbf{S}$. As a result, the action set $\mathcal{Q}(\mathbf{S})$ should be selected as an subset of $\cup_{n=1}^{N}\mathcal{C}_n$, i.e., $\mathcal{Q}(\mathbf{S}) \subseteq \cup_{n=1}^{N}\mathcal{C}_n$.

For the purpose of limiting the cardinality of $\mathcal{Q}(\mathbf{S})$, it is quite nature to select one basic action set as $\mathcal{Q}(\mathbf{S})$. We can easily prove that an random action selected from all N basic action set can eliminate a same number of non-zero elements on average. However, according to Theorem 1, the non-zero elements corresponding to smaller feature components play more important roles than those corresponding to larger feature components, the basic action set $\mathcal{C}_{\bar{\pi}^{(1)}}$ can best minimize the objective function, where $\bar{\pi}_{\mathbf{S}}$ is the index sequence of feature components sorted from smallest to largest and $\bar{\pi}_{\mathbf{S}}^{(1)}$ is the first index in $\bar{\pi}_{\mathbf{S}}$ As a result, we set

$$\mathcal{Q}(\mathbf{S}) = \mathcal{C}_{\bar{\pi}_{\mathbf{S}}^{(1)}} \tag{13}$$

4 Simulations

In this section, we demonstrate the effectiveness of our designed policy through simulations compared with existing algorithms:

- *PSD* algorithm proposed in [6], which is equivalent to the GE algorithm in the viewpoint of computational complexity.
- *SVD-NC* algorithm proposed in [12], which uses singular value decomposition to easily obtain the inverse of the decoding matrix and obtain the source packets through multiplying the inverse of the decoding matrix by the coded packets.
- *DC-DBS* algorithm proposed in [10], which is a variant GE algorithm with two improvements, i.e., density check is adopted to make sure when row-adding operation is executed the row with less non-zero elements is added to the row with more non-zero elements, delayed backwards substitution is adopted to make sure non-zero elements are not created in backwards substitution.
- *Sparse-NC* algorithm proposed in [11], where a reorder mechanism is proposed for the decoding matrix, so that density check and delayed backwards substitution is achieved.

We set $a_{(m,n)} = 1, \forall m, n \in [N]$. Although this setting may lead to some performance loss in terms of const weight summation, it can avoid complex calculations, such that variable weight can be maintained in a low level. We assume $L = 10000$, which is typical value for the length of data packets. Figures 1 and 2 depict the comparison of different decoding algorithms versus N in terms of the const weight summation $V_{\mu}^{c}(\mathbf{S}_1)$ and the weight summation $V_{\mu}(\mathbf{S}_1)$ respectively. As seen, in both figures, our proposed algorithm achieves best performance among the five algorithms. The comparison results demonstrate that the variable weight summation has been well limited though policy optimization, and is suitable to be applied in the decoding of network codes. In practical situation, N can not be ignored compared with L, that means, the variable weight summation can be a significant component of the weight summation (Fig. 3).

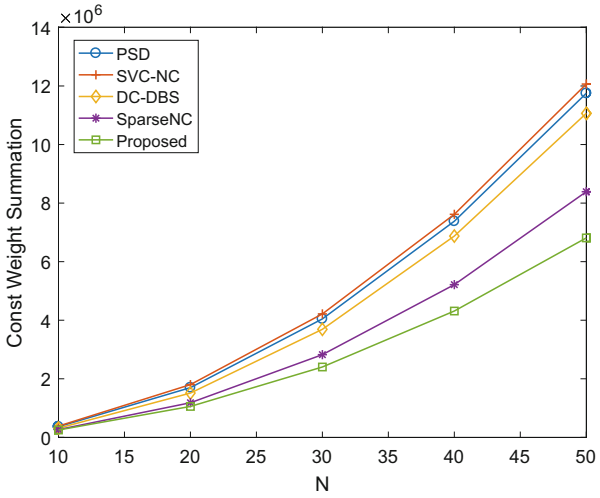

Fig. 2. Comparisons of different algorithms in terms of T_μ versus N. $\rho = 0.5, L = \infty$.

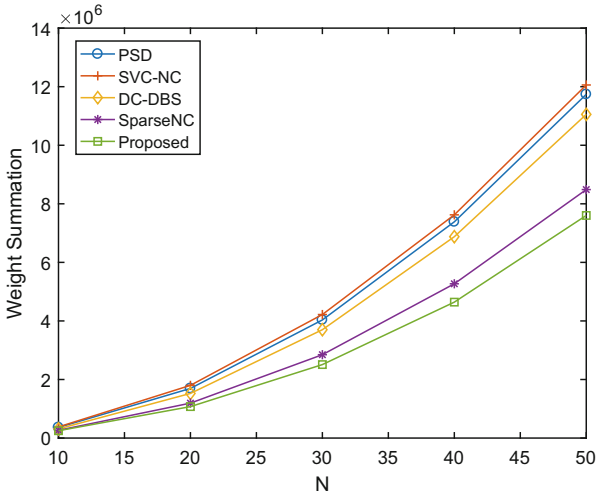

Fig. 3. Comparisons of different algorithms in terms of T_μ versus N. $\rho = 0.5, L = \infty$.

5 Conclusion

In this paper, we focus on minimizing the decoding complexity of binary network codes. Traditional Gaussian elimination method fail to optimize the decoding procedure and suffers from a large decoding complexity. We formulate the decoding problem into a special shortest path problem where the weight of each edge consists of: (1) a const weight due to the execution of the action; (2) a variable weight due to the adopted policy in determining the action. Through

optimization upon the policy, our proposed decoding method significantly outperforms existing methods.

References

1. Li, S.-Y., Yeung, R.R.W., Cai, N.: Linear network coding. IEEE Trans. Inf. Theory **49**(2), 371–381 (2003)
2. Luby, M.: LT codes. In: Proceedings 43rd Annual IEEE Symposium on Foundations of Computer Science, pp. 271–282 (2002)
3. Shokrollahi, A.: Raptor codes. IEEE Trans. Inf. Theory **52**(6), 2551–2567 (2006)
4. Ho, T., et al.: A random linear network coding approach to multicast. IEEE Trans. Inf. Theory **52**(10), 4413–4430 (2006)
5. Sung, C.W., Shum, K.W., Huang, L., Kwan, H.Y.: Linear network coding for erasure broadcast channel with feedback: complexity and algorithms. IEEE Trans. Inf. Theory **62**(5), 2493–2503 (2016)
6. Wang, J., Xu, K., Xu, Y., Zhang, D., Zhu, Y.: Pseudo-systematic decoding of hybrid instantly decodable network code for wireless broadcasting. IEEE Wirel. Commun. Lett. **7**, 840–843 (2018)
7. Tassi, A., Chatzigeorgiou, I., Lucani, D.E.: Analysis and optimization of sparse random linear network coding for reliable multicast services. IEEE Trans. Commun. **64**(1), 285–299 (2016)
8. Kaltofen, E., David Saunders, B.: On Wiedemann's method of solving sparse linear systems. In: Mattson, H.F., Mora, T., Rao, T.R.N. (eds.) AAECC 1991. LNCS, vol. 539, pp. 29–38. Springer, Heidelberg (1991). https://doi.org/10.1007/3-540-54522-0_93
9. Coppersmith, D.: Solving linear equations over GF(2): block Lanczos algorithm. Linear Algebra Appl. **192**, 33–60 (1993)
10. Heide, J., Pedersen, M.V., Fitzek, F.H.P.: Decoding algorithms for random linear network codes. In: Casares-Giner, V., Manzoni, P., Pont, A. (eds.) NETWORKING 2011. LNCS, vol. 6827, pp. 129–136. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23041-7_13
11. Feizi, S., Lucani, D., Sørensen, C., Makhdoumi, A., Médard, M.: Tunable sparse network coding for multicast networks. In: Proceedings of the International Symposium on Network Coding Coding (NetCod 2014), Aalborg, Denmark, pp. 1–6 (2014)
12. Kwon, J., Park, H.: Low complexity algorithms for network coding based on singular value decomposition. In: IEEE 2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN), pp. 641–643 (2016)
13. Sorour, S., Valaee, S.: Completion delay minimization for instantly decodable network codes. IEEE/ACM Trans. Netw. (TON) **23**, 1553–1567 (2015)
14. Bellman, R.: Dynamic programming and lagrange multipliers. Proc. Natl. Acad. Sci. **42**, 767–769 (1956)
15. Hammer, P.L., Rudeanu, S.: Boolean Methods in Operations Research and Related Areas, vol. 7. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-85823-9