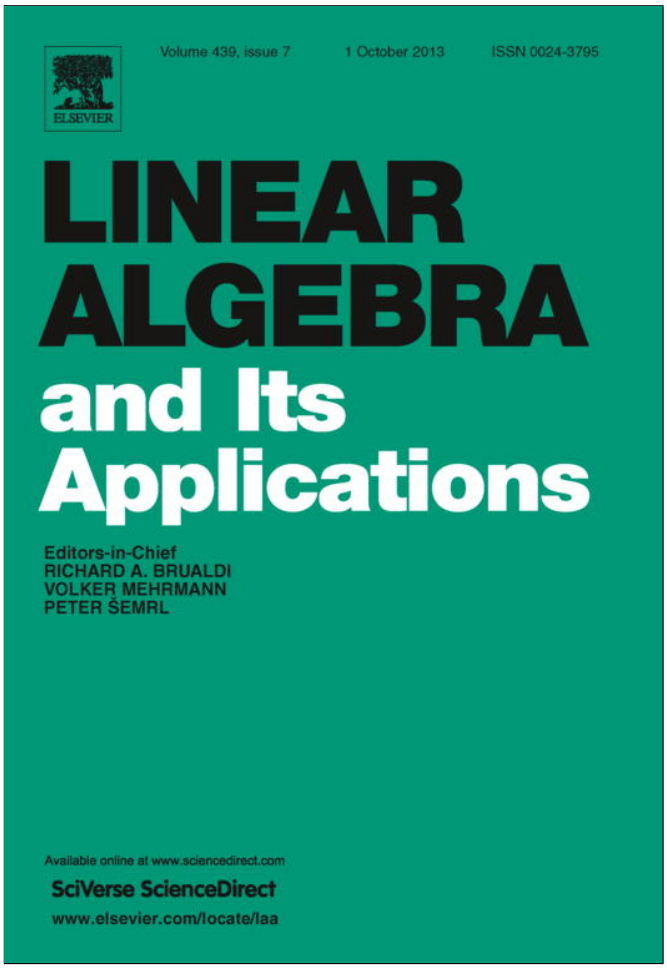

Linear Algebra and its Applications 439 (2013) 1815–1824

Contents lists available at SciVerse ScienceDirect

Linear Algebra and its Applications

www.elsevier.com/locate/laa

Solving sparse linear systems of equations over finite fields using bit-flipping algorithm

Asieh A. Mofrad [a], M.-R. Sadeghi [a,*], D. Panario [b]

[a] Department of Mathematics and Computer Science, Amirkabir University of Technology, Tehran, Iran
[b] School of Mathematics and Statistics, Carleton University, K1S 5B6, Ontario, Canada

ARTICLE INFO

Article history:
Received 22 February 2012
Accepted 20 May 2013
Available online 7 June 2013
Submitted by E. Zerz

Keywords:
Bit-flipping
Sparse linear systems
Finite fields

ABSTRACT

Let $\mathbb{F}_q$ be the finite field with q elements. We give an algorithm for solving sparse linear systems of equations over $\mathbb{F}_q$ when the coefficient matrix of the system has a specific structure, here called *relatively connected*. This algorithm is based on a well-known decoding algorithm for low-density parity-check codes called bit-flipping algorithm. We modify and extend this hard decision decoding algorithm. The complexity of this algorithm is linear in terms of the number of columns n and the number of nonzero coefficients ω of the matrix per iteration. The maximum number of iterations is bounded above by m, the number of equations.

1. Introduction

Let $\mathbb{F}_q$ be the finite field with q elements where q is a prime power. In this paper we consider the classical problem of solving large sparse linear systems of equations of the form

$$A\mathbf{x} = \mathbf{b} \tag{1}$$

where $A = [a_{ij}]_{m\times n}$, $\mathbf{b} = [b_i]_{m\times 1}$ and $\mathbf{x} = [x_i]_{n\times 1}$ with a_{ij}, b_i, x_i in some algebraic structure for $1 \leqslant i \leqslant m$, $1 \leqslant j \leqslant n$. Our interest in this paper is when the elements belong to the finite field $\mathbb{F}_q$.

In general, classical Gaussian elimination method can be applied. For square matrices of dimension $n \times n$, Gaussian elimination takes time $O(n^3)$ and space $O(n^2)$ that makes it impractical for large and sparse systems. Over finite fields several algorithms have been proposed such as Wiedemann method [13] and its variants [4], the conjugate gradient and Lanczos algorithms (see [3,5,9]) and

* Corresponding author.
E-mail addresses: abolpour_asie@aut.ac.ir (A.A. Mofrad), msadeghi@aut.ac.ir (M.-R. Sadeghi), daniel@math.carleton.ca (D. Panario).

0024-3795/$ – see front matter
http://dx.doi.org/10.1016/j.laa.2013.05.016

also the structured Gaussian elimination method [9]. These algorithms can be used for linear systems modulo any integer g by applying the Chinese remainder theorem and Hensel lifting method [6]. These methods are probabilistic in nature and are designed for square matrices A. If the coefficient matrix of the system is an $m \times n$ matrix with $m \neq n$, the system has to be converted to a new system with a square coefficient matrix, so then the new system can be solved. In this paper we extend Gallager's bit-flipping (BF) algorithm [8] for solving sparse linear systems over finite field $\mathbb{F}_q$ when the coefficient matrix has a specific form that we call *relatively connected*. The bit-flipping is a low-complexity algorithm which was introduced for decoding LDPC (low-density parity-check) codes.

LDPC codes were first introduced by Gallager [8] in early 1960s and rediscovered by MacKay [11] in 1996. An LDPC code is a code such that its parity-check matrix H is sparse. In communications, when a vector $\mathbf{x}$ is sent through a channel, it is affected by noise (induced by the channel). Hence, a vector $\mathbf{y}$, which may be different from $\mathbf{x}$, is received. The decoding problem is, given a received vector $\mathbf{y}$, to find a good estimation for $\mathbf{x}$. The key points here are that $\mathbf{x}$ has to satisfy all parity-checks, that is, $H\mathbf{x}^{\mathbf{T}} = \mathbf{0}$ over $\mathbb{F}_q$ (in practice we are mostly interested in the case $q = 2$). When using LDPC codes an important issue is that the parity-check matrix H is sparse. When solving a sparse linear system, however, in contrast to the communications case, we do not have $\mathbf{y}$ as an input vector to the algorithm and so we start the algorithm with a random vector.

There are several LDPC algorithms based on *soft* and *hard* decisions [10]. In soft decision algorithms, we have input vectors that contain channel information in their components. This information is used when decisions are taken while executing the algorithm. Hard decision algorithms, in contrast, do not contain channel information in their components. Hence, since when solving systems of equations we do not deal with channels, we choose an effective hard decision algorithm like bit-flipping.

In [1], the bit-flipping algorithm is used for solving sparse linear systems of equations modulo a prime number p under the condition that the coefficient matrix has column degree at most 2. In the present paper we extend the algorithm to solve more general systems by introducing the *relatively connected* constraint. As a consequence, we have extended the work from matrices having column degree at most 2 to matrices having any column degree under the relatively connected constraint.

The structure of the paper is as follows. In Section 2, the general form of the bit-flipping decoding algorithm is explained. In Section 3, the extended bit-flipping algorithm is introduced. This algorithm can solve linear systems of equations over $\mathbb{F}_q$ that satisfy the relatively connected condition. This concept as well as examples are given in this section. The extended bit-flipping algorithm for systems over $\mathbb{F}_2$ is given in Section 4. This case is of practical importance, while at the same time easier to explain than the general case over $\mathbb{F}_q$ treated in Section 5. The cost of the algorithm is provided in Section 6. Finally, in Section 7, further directions of research are given.

2. The bit-flipping algorithm

For decoding binary LDPC codes we are given a received vector $\mathbf{y}$ and need to find a vector $\mathbf{x}$ such that $H\mathbf{x}^{\mathbf{T}} = \mathbf{0}$ over $\mathbb{F}_2$, or equivalently, $H\mathbf{x}^{\mathbf{T}} \equiv \mathbf{0} \pmod 2$. For this purpose, in the bit-flipping algorithm [8], the decoder first computes all the parity-checks; then, given a fixed parameter β called the *threshold*, the decoder changes any bit in the received vector $\mathbf{y}$ that is contained in more than β unsatisfied parity-check equations. The flipped bits are then used in the next iteration of the decoding process. The decoding algorithm stops when either all of the parity-checks are satisfied or a predefined maximum number of iterations is reached.

In contrast with choosing β as a fix number, adaptive threshold is suggested in [7]: if decoding fails for a given value of β, then the value of β can be reduced to allow further decoding iterations. Bit-flipping decoding with an adaptive threshold technique is given in detail in [2]. We use adaptive threshold technique but in a slightly different manner than in [2].

A simple bit-flipping decoding algorithm is given in Algorithm 1.

Algorithm 1 *(Bit-flipping algorithm).*
Input: parity-check equations and input vector $\mathbf{y}$.
Output: a solution vector.

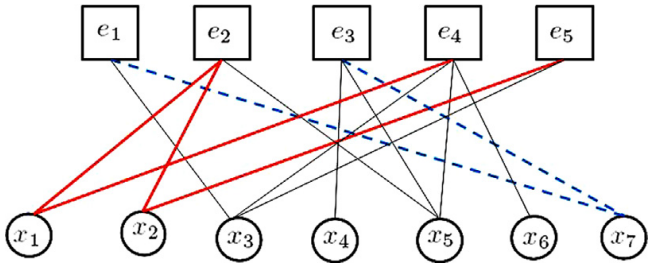

Fig. 1. A relatively connected Tanner graph.

1. Evaluate the parity-check equations with the input vector **y**. If all of them are satisfied, stop.
2. Find f_i, the number of unsatisfied parity-check equations for each bit i, $1 \leqslant i \leqslant n$.
3. Determine Ω, the set of bits for which f_i is the largest, $1 \leqslant i \leqslant n$. If this largest value is less than β, the algorithm has failed; stop.
4. Flip the bits in the set Ω.
5. Repeat Steps 1 to 4 until all the parity-check equations are satisfied or a predefined maximum number of iterations is reached.

3. Extended bit-flipping algorithm

In this paper we present two versions of the bit-flipping algorithm for solving the system (1). The algorithms are called *extended bit-flipping*. The first one, conceptually easier, is over $\mathbb{F}_2$, while the second one is for any finite field $\mathbb{F}_q$. The algorithms are based on *Tanner graphs* [12]. In the Tanner graph corresponding to a system, a variable x_i is shown as a variable node, and an equation e_k is shown as an equation node. If a variable appears in an equation, there is an edge between their corresponding nodes in the graph. Clearly, a Tanner graph is a bipartite graph in which equation nodes are connected only to variable nodes and reciprocally.

The extended bit-flipping algorithm guarantees finding a solution for systems with a specific property in their Tanner graphs that we call *relatively connected* condition.

Definition 3.1. A Tanner graph is *relatively connected* (RC), if there is a partition on the equation nodes such that the following two properties hold:

1. For any two equation nodes of each equation set in the partition, there exists at least a path on the Tanner graph between them such that all variable nodes in the path have degree 2.
2. For each equation set there exists at least a degree one variable node that is adjacent to an equation node of the set.

We call each of these equation sets a *relative set*. If one relative set has just one equation node, we call it a *single equation node set*.

Example 3.1. Consider the Tanner graph in Fig. 1. It is easy to see that there are just two relative sets: $\{e_1, e_3\}$ and $\{e_2, e_4, e_5\}$. The dashed path satisfies the first RC property for the set $\{e_1, e_3\}$, and the variable node x_4 satisfies the second property for this set. Similarly, the bold path and x_6 satisfy properties 1 and 2, respectively, for the set $\{e_2, e_4, e_5\}$.

There are two important paths for our algorithm that we call *Type* 1 and *Type* 2, respectively. They are depicted in Fig. 2. Type 1 paths start and finish in unsatisfied equation nodes (drawn with crossed squares); a Type 2 path has last variable node with degree 1. In these paths, all intermediate variable nodes have degree 2.

Indeed, the relatively connected condition implies that if we keep all columns (or variable nodes in the Tanner graph) with degrees 1 and 2 together with their connected equation nodes, and remove the rest of columns and rows, the number of rows (or equation nodes) should stay unchanged. Comparing the relatively connected condition with the constraint that column degree be at most 2 in [1],

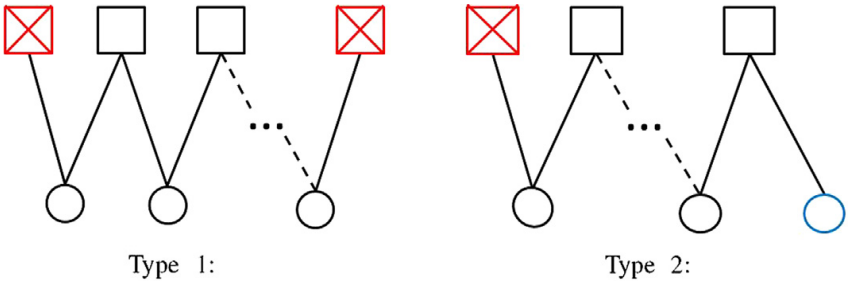

Fig. 2. Two type of paths in a Tanner graph.

it is clear that the new condition in this paper is more general. Therefore, it is more probable that the sparse system (that should be solved) passes the relatively connected constraint than the column degree at most 2 constraint. Moreover, some algorithms need some precomputations before solving the system. For instance, the system may be converted to an equivalent square one. Converting a sparse system to a relatively connected one, following the new version in this paper, is much simpler than in [1]; finding such an equivalent system, is even more simpler when compared to [1], if $n \gg m$.

4. Extended bit-flipping algorithm for linear system of equations over $\mathbb{F}_2$

We start by introducing the notation to be used on the algorithms. For each variable node x_i, $1 \leqslant i \leqslant n$, we denote:

d_{x_i} the degree of variable node x_i;
f_i the number of unsatisfied equation nodes adjacent to variable node x_i;
β_i the reliability ratio f_i/d_{x_i};
β the largest value of β_i's;
Ω the set of variable nodes with reliability ratio equal to β.

If $\mathbf{x}' = [x'_i]_{n\times 1}$ is the output solution of the system (1) and $\mathbf{x} = [x_i]_{n\times 1}$ is an arbitrary vector, a variable node x_i is called a *true node* if $x_i = x'_i$ and it is called a *false node* if $x_i \neq x'_i$. In addition, an adjacent edge to a true node or to a false node is called a *true edge* or a *false edge*, respectively.

Algorithm 2 *(Extended bit-flipping algorithm over $\mathbb{F}_2$).*
Input: the system and a random initial vector $\mathbf{x} \in \{0,1\}^n$.
Output: a solution vector $\mathbf{x}$.

1. Set vector $\mathbf{x}$ on the system. If all the equations are satisfied, then we have a solution; stop.
2. Compute f_i and β_i, for each variable node x_i, $1 \leqslant i \leqslant n$.
3. Compute the threshold β and identify the set Ω.
4. If $\beta > 1/2$ for a variable node $x_i \in \Omega$, flip the value of x_i and remove it from Ω. Repeat the process for any other element of Ω which still has reliability ratio β, until Ω become the empty set. Go to Step 1 with these modified variable node values.
5. If $\beta = 1/2$, for a variable node $x_i \in \Omega$, find a path such as Fig. 2 and flip the value of all variable nodes in the path, then rebuild Ω (the crossed square in Fig. 2 is an adjacent equation node to x_i, however, the adjacent variable node to a crossed square is not necessarily x_i). Repeat the process with the modified variable node values until Ω is the empty set and then go to Step 1 with these new variable node values.

Proposition 4.1. *If the system (1) over a finite field $\mathbb{F}_q$ has at least one solution and the corresponding Tanner graph is relatively connected, there exists a path in Step 5 starting from an unsatisfied equation node e_k (adjacent to a variable node x_i, like in Step 5), and ending with another unsatisfied equation node e_l; or there exists a path starting from e_k and ending with a degree one variable node (Type 1 and Type 2 in Fig. 2, respectively).*

Proof. In Step 5, since $\beta = 1/2$, for each variable node $x_i \in \Omega$ there exists at least an unsatisfied adjacent equation node e_k which belongs to a relative set and is not a single equation node. Because there exists a degree one variable node adjacent to one of the equation nodes of each set, and there is a path between any two equation nodes (by property 1 of RC Tanner graphs), a Type 2 path exists. If there is more than one unsatisfied equation node in the relative set, it is clear from the RC definition that a Type 1 path exists. □

Next we show the correctness of the extended bit-flipping algorithm for linear systems of equations over $\mathbb{F}_2$ satisfying the RC conditions. Alternatively, define *RC systems* as being the ones that satisfy the RC conditions. We observe that systems such that variables appear in at most two equations satisfy the relatively connected condition. The proof of correctness for this particular type of systems with column degree at most 2 over $\mathbb{F}_2$ is shown in [1]. Here we extend that proof to systems satisfying the relatively connected condition.

Proposition 4.2. *For systems like* (1) *over* $\mathbb{F}_2$ *satisfying the relatively connected condition, the case* $\beta < 1/2$ *in the algorithm never occurs.*

Proof. From the relatively connected property, each equation node has at least a degree 1 or a degree 2 variable node. By definition of β an unsatisfied equation node causes $\beta \geqslant 1/2$. The case $\beta = 0$ means that all equations are satisfied and so the algorithm stops in Step 1. □

Theorem 4.1. *If the system* (1) *over* $\mathbb{F}_2$ *has at least one solution and its Tanner graph is relatively connected, the algorithm finds a solution.*

Proof. For case $\beta > 1/2$, by definition of threshold and Step 4, the number of unsatisfied equations is reduced after flipping variables.

In case $\beta = 1/2$, by Proposition 4.1, there is a path (Type 1 or Type 2) on the graph. By flipping the value of the variable nodes in the path, all equation nodes of the path are now satisfied. Thus the number of unsatisfied equation nodes reduces by two units for Type 1 and one unit for Type 2.

We can conclude that during each iteration of the algorithm, the number of unsatisfied equations is reduced by at least one unit. The number of equation nodes is finite so the algorithm stops when the number of unsatisfied equations reduces to zero. □

The algorithm is illustrated in Example 4.1.

Example 4.1. Consider the following system over $\mathbb{F}_2$ with Tanner graph depicted in Fig. 1:

$$\begin{aligned}
x_3 \qquad\qquad\quad &+ x_7 = 0,\\
x_1 + x_2 \qquad + x_5 \quad &= 0,\\
x_4 + x_5 \quad &+ x_7 = 1,\\
x_1 \quad + x_3 \quad + x_5 + x_6 \quad &= 1,\\
x_2 + x_3 \quad &= 0.
\end{aligned}$$

Let us suppose that $\mathbf{x} = (1, 0, 1, 1, 0, 1, 0)$ is the input random vector. The algorithm works as follows:

Step 1. $\mathbf{x}$ does not satisfy equations e_1, e_2 and e_5.
Step 2. We have

$$f_1 = 1, \quad f_2 = 2, \quad f_3 = 2, \quad f_4 = 0, \quad f_5 = 1, \quad f_6 = 0, \quad f_7 = 1,$$

and

$$\beta_1 = \frac{1}{2}, \quad \beta_2 = 1, \quad \beta_3 = \frac{2}{3}, \quad \beta_4 = 0, \quad \beta_5 = \frac{1}{3}, \quad \beta_6 = 0, \quad \beta_7 = \frac{1}{2}.$$

Step 3. $\beta = 1$ and $\Omega = \{x_2\}$.

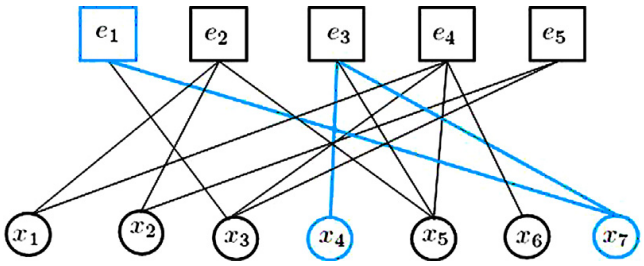

Fig. 3. A path between e_1 and x_4.

Step 4. Flip x_2, set $x_2 = 1$ and exclude x_2 from Ω. Since $\Omega = \emptyset$, go to Step 1.
Step 1. The modified vector $(1, 1, 1, 1, 0, 1, 0)$ does not satisfy the equation e_1.
Step 2.

$$f_1 = 0, \quad f_2 = 0, \quad f_3 = 1, \quad f_4 = 0, \quad f_5 = 0, \quad f_6 = 0, \quad f_7 = 1,$$

and

$$\beta_1 = 0, \quad \beta_2 = 0, \quad \beta_3 = \frac{1}{3}, \quad \beta_4 = 0, \quad \beta_5 = 0, \quad \beta_6 = 0, \quad \beta_7 = \frac{1}{2}.$$

Step 3. $\beta = \frac{1}{2}$ and $\Omega = \{x_7\}$.
Step 5. Equation node e_1 is adjacent to variable node x_7 and there is a path between e_1 and x_4, which is a degree one variable node; see Fig. 3 for the path. We flip the variable node values contained in the path (that is, $x_4 = 0$ and $x_7 = 1$), $\Omega = \emptyset$ and go to Step 1.
Step 1. All equations are satisfied, so the output vector is:

$$x = (1, 1, 1, 0, 0, 1, 1).$$

5. Extended bit-flipping algorithm for linear system of equations over $\mathbb{F}_q$

We use the same notations for β, Ω and d_{x_i} as before. Instead of f_i we define t_i for each variable node x_i as follows. For each adjacent equation node to x_i, first we fix the other variable node values and then solve the equation for x_i. This involves, more precisely, to solve an equation of the form $cx = d$ over $\mathbb{F}_q$. This equation has a unique solution since we work over a finite field.

Next we determine x_i', a value in $\mathbb{F}_q$ that is satisfied by the maximum number of adjacent equation nodes to x_i. If there is more than one such value and the current value of x_i is one of them, we set x_i' to be the current value of x_i; otherwise, we choose one of them at random. We now define t_i to be this maximum number of adjacent equation nodes satisfied by x_i'.

Finally, if the current value of the variable node x_i is satisfied by t_i equations, then we set the reliability ratio of x_i to be $\beta_i = -\frac{t_i}{d_{x_i}}$; otherwise we set $\beta_i = \frac{t_i}{d_{x_i}}$.

Remark 5.1. In contrast with the case $\mathbb{F}_2$ where we set f_i as the number of unsatisfied equations, in $\mathbb{F}_q$ we set t_i as the number of satisfied equations. In both cases we choose the maximum of f_i/d_i (in $\mathbb{F}_2$) and t_i/d_i (in $\mathbb{F}_q$). In $\mathbb{F}_2$ this entails finding the variables with most unsatisfied adjacent equation nodes. Hence flipping them is helpful. In $\mathbb{F}_q$ we need: (1) an element of $\mathbb{F}_q$ (x_i') with maximum number of adjacent equation nodes, and (2) the knowledge whether this number is the current value of x_i or not. If we do not know whether it is the current value of x_i, we may fall into a loop (for example, a variable node with $\beta_i = 1$ causes $\beta = 1$ forever). Hence, we use the negative sign in order to exclude them from the flipping step.

We observe that our threshold β is a real number satisfying $-1 < \beta \leqslant 1$ but in bit-flipping decoding algorithms the threshold is a positive *integer*. The reason for choosing a fraction for the threshold is that unlike many parity-check matrices for decoding algorithms, columns of the coefficient matrix of the linear system do not necessarily have the same column degree. This means that if the number

of unsatisfied parity-checks for each variable is known, then the number of satisfied parity-checks is also known and the decision for flipping can be done. However, in linear systems there may exist some different column degrees. Hence, for a given variable node x_i we define the reliability ratio β_i and set the threshold β as a maximum number of β_i's (not f_i's). This helps us to correctly decide whether to flip the variable value or not.

Algorithm 3 *(Extended bit-flipping algorithm over $\mathbb{F}_q$).*
Input: the system and a random initial vector $\mathbf{x} \in \mathbb{F}_q^n$.
Output: a solution vector $\mathbf{x}$.

1. Set vector $\mathbf{x}$ into the system. If all the equations are satisfied, then we have a solution; stop.
2. Compute t_i and β_i for each variable node x_i, $1 \leqslant i \leqslant n$.
3. Determine the threshold β. If $\beta > 0$ construct the set Ω and go to Step 4. If $\beta < 0$ go to Step 5.
4. For an $x_i \in \Omega$, set $x_i = x_i'$. Exclude x_i from Ω and reconstruct Ω. Repeat this step until $\Omega = \emptyset$ and go to Step 1 with the modified variable values.
5. For a variable node $x_i \in \Omega$ chose an unsatisfied adjacent equation node to x_i, say e_k. Find a degree one variable node (for a Type 2 path) or an unsatisfied equation node (for a Type 1 path). Starting from x_t (the adjacent variable node in the path to e_k), each variable node in this path gets a value such that the previous equation node is satisfied. Repeat the process with the new variable node values until Ω is the empty set and then go to Step 1 with these new variable node values.

Proposition 4.1 also ensures the existence of a Type 1 or a Type 2 path in Step 5 for the above algorithm. The following theorem addresses the correctness of the extended bit-flipping algorithm for linear system of equations over $\mathbb{F}_q$.

Theorem 5.1. *If the system (1) over $\mathbb{F}_q$ has at least one solution and its Tanner graph is relatively connected, the algorithm finds a solution.*

Proof. In Step 4 ($\beta > 0$), the number of unsatisfied equations is reduced after changing the values of the elements in Ω.

In Step 5 ($\beta < 0$), the change of variable node values in the path ensures that the first unsatisfied equation node in the path is now satisfied while the other equation nodes in the path remain satisfied. Hence, the number of unsatisfied equation nodes reduces by at least 1 unit (we observe that if we find a Type 2 path, it is possible that the number of unsatisfied equation is reduced by two).

Therefore, as in the algorithm over $\mathbb{F}_2$, the number of unsatisfied equations reduces to zero and the algorithm finally stops in Step 1. □

Remark 5.2. We observe that β is at least $-1/2$ because each unsatisfied equation node (instead of single nodes) has at least a degree 2 variable node and so its reliability ratio is $-1/2$, and β is the maximum of β_i's. If there is no unsatisfied equation node we have already achieved the solution in Step 1. Also β never gets the value 0 because for each x_i, t_i is at least 1.

The algorithm is illustrated in the following example; we choose the prime field $\mathbb{F}_5$, for simplicity in the calculations.

Example 5.1. Consider the following system over $\mathbb{F}_5$:

$$\begin{aligned}
4x_1 + x_2 \qquad\qquad + 3x_6 \qquad\qquad\qquad &= 3,\\
4x_4 + x_5 \qquad + 4x_8 + 2x_9 + x_{10} &= 4,\\
4x_1 + 3x_4 \qquad\qquad + 2x_9 \qquad &= 1,\\
3x_3 \qquad + 4x_7 \qquad\qquad &= 1,\\
x_2 \qquad + 3x_7 \qquad\qquad &= 1,\\
x_7 + 3x_8 + 4x_9 \qquad &= 4,\\
3x_1 \qquad + 3x_5 \qquad\qquad\qquad &= 4.
\end{aligned}$$

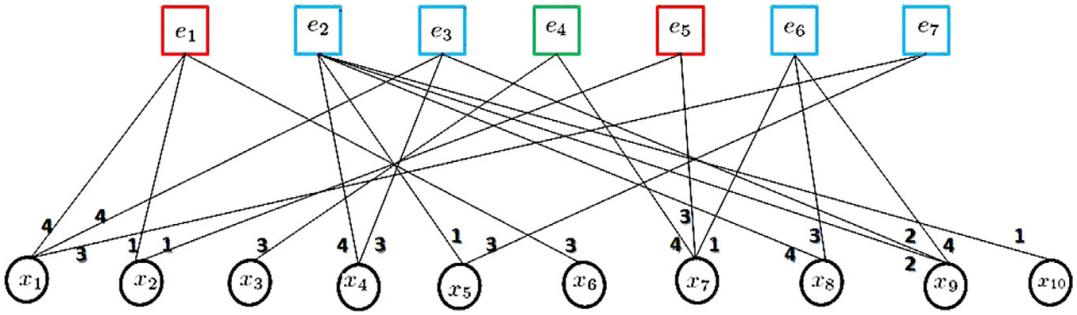

Fig. 4. Tanner graph of the system.

The graph of the system is shown in Fig. 4. We do not include the coefficients (in this case elements in $\mathbb{F}_5$) for simplicity.

There are three relative sets: $\{e_1, e_5\}$, $\{e_2, e_3, e_6, e_7\}$ and $\{e_4\}$ which is a single equation node. The algorithm finds the solution of the system over $\mathbb{F}_5$ as follows:

Let the input vector be $\mathbf{x} = (3, 1, 0, 2, 1, 4, 4, 2, 3, 0)$.

Step 1. The equations e_1, e_2, e_3, e_5, e_6 and e_7 are not satisfied.
Step 2. In order to compute t_i and β_i, we first compute x'_i. We have

$$x'_1 = 1, \quad x'_2 = 4, \quad x'_3 = 0, \quad x'_4 = 1, \quad x'_5 = 2,$$
$$x'_6 = 0, \quad x'_7 = 4, \quad x'_8 = 1, \quad x'_9 = 1, \quad x'_{10} = 1.$$

We get

$$t_1 = 1, \quad t_2 = 2, \quad t_3 = 1, \quad t_4 = 2, \quad t_5 = 1,$$
$$t_6 = 1, \quad t_7 = 1, \quad t_8 = 2, \quad t_9 = 2, \quad t_{10} = 1,$$

and

$$\beta_1 = \frac{1}{3}, \quad \beta_2 = 2, \quad \beta_3 = -1, \quad \beta_4 = 1, \quad \beta_5 = \frac{1}{2},$$
$$\beta_6 = 1, \quad \beta_7 = -\frac{1}{3}, \quad \beta_8 = 1, \quad \beta_9 = \frac{2}{3}, \quad \beta_{10} = 1.$$

Step 3. $\beta = 1$ and $\Omega = \{x_2, x_4, x_6, x_8, x_{10}\}$, then go to Step 4.
Step 4. Set $x_4 = x'_4 = 1$ and remove it from Ω. Since $\beta_2 = 1$, $\beta_6 = 1$, $\beta_8 = -\frac{1}{2}$ and $\beta_{10} = -1$, we have $\Omega = \{x_2, x_6\}$. Set $x_6 = x'_6 = 0$ and remove it from Ω. Since $\beta_2 = -\frac{1}{2}$, $\Omega = \emptyset$ and we go to Step 1.
Step 1. The modified vector does not satisfy the equations e_5, e_6 and e_7.
Step 2. We just compute t_i and β_i's values for variable nodes which are adjacent to at least one unsatisfied equation node. Obviously, the value of β_i's for the other variable nodes equals -1. We obtain

$$x'_1 = 3, \quad x'_2 = 1, \quad x'_5 = 1, \quad x'_7 = 4, \quad x'_8 = 2, \quad x'_9 = 3,$$

and hence we get

$$t_1 = 2, \quad t_2 = 1, \quad t_5 = 1, \quad t_7 = 1, \quad t_8 = 1, \quad t_9 = 2,$$

and

$$\beta_1 = -\frac{2}{3}, \quad \beta_2 = -\frac{1}{2}, \quad \beta_5 = -\frac{1}{2}, \quad \beta_7 = -\frac{1}{3}, \quad \beta_8 = -\frac{1}{2}, \quad \beta_9 = -\frac{2}{3}.$$

Step 3. We have $\beta = -\frac{1}{3}$ so $\Omega = \{x_7\}$ and go to Step 5.

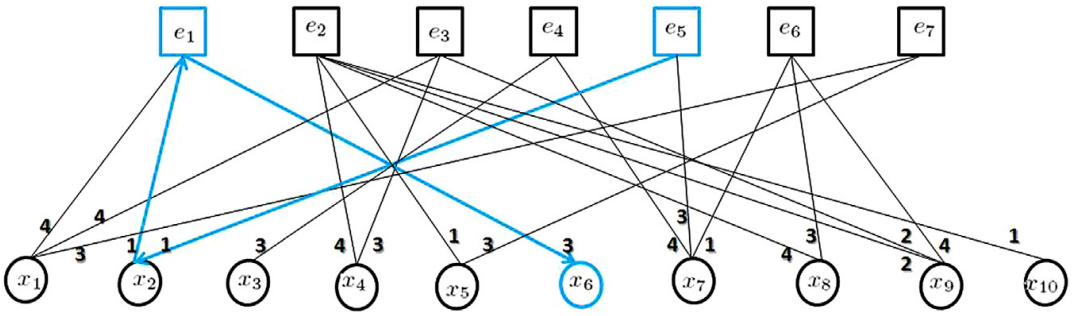

Fig. 5. A path between e_5 and x_6.

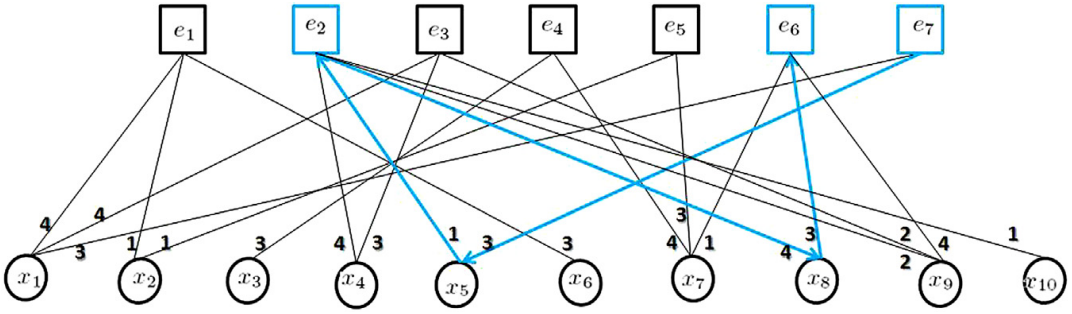

Fig. 6. A path between e_7 and e_6.

Step 5. The equation node e_5 is adjacent to the variable x_7 and there is a path between e_5 and x_6, which is a variable node with degree 1; see Fig. 5.

We set $x_2 = 4$ so e_5 is satisfied; then we set $x_6 = 4$ in order that e_2 is satisfied. Since $\Omega = \emptyset$ we go to Step 1.

Step 1. The modified vector does not satisfy the equations e_6 and e_7.

Step 2. We obtain

$$x'_1 = 3, \qquad x'_5 = 1, \qquad x'_7 = 4, \qquad x'_8 = 2, \qquad x'_9 = 3,$$

and hence we get

$$t_1 = 2, \qquad t_5 = 1, \qquad t_7 = 2, \qquad t_8 = 1, \qquad t_9 = 2,$$

and

$$\beta_1 = -\frac{2}{3}, \qquad \beta_5 = -\frac{1}{2}, \qquad \beta_7 = -\frac{2}{3}, \qquad \beta_8 = -\frac{1}{2}, \qquad \beta_9 = -\frac{2}{3}.$$

Step 3. We have $\beta = -\frac{1}{2}$ so $\Omega = \{x_5, x_8\}$ and go to Step 5.

Step 5. We obtain that e_7 is unsatisfied and is adjacent to x_5. There is a Type 1 path between e_7 and e_6; see Fig. 6.

For satisfying e_7 we set $x_5 = 0$ and for satisfying e_2 we set $x_8 = 1$. We obtain that e_6 is satisfied and $\Omega = \emptyset$, so we go to Step 1.

Step 1. All equations are satisfied, so the output solution is:

$$x = (3, 4, 0, 1, 0, 4, 4, 1, 3, 0).$$

We comment that in Step 2 of the algorithms it is enough to compute f_i and t_i (respectively, for the algorithms over $\mathbb{F}_2$ and $\mathbb{F}_q$) and β_i for variable nodes which are adjacent to at least one unsatisfied equation node. The β_i value for the other variable nodes is equal to 0 in $\mathbb{F}_2$ and to -1 in $\mathbb{F}_q$. Also in Step 3 of the algorithms, we can flip (or change) the value of x_i's with greater d_{x_i}. This might cause a faster reduction in the number of unsatisfied equation nodes.

6. Complexity analysis of the extended bit-flipping algorithms

Proposition 6.1. *The complexity of the extended bit-flipping algorithms is linear in terms of the number of variables and the number of nonzero elements of the coefficient matrix A.*

Proof. We analyze the cost of the algorithms step by step for one iteration. Let d_{x_i}, $1 \leqslant i \leqslant n$, denote the degree of variable node x_i and let ω denote the number of nonzero elements of the coefficient matrix A.

In Step 1 we have at most ω multiplications, $(\omega - m)$ summations and m comparisons.

In Step 2, for $\mathbb{F}_2$, in order to construct the f_i's, we need ω comparisons and at most $(\omega - n)$ summations and in general $(2\omega - n)$ operations. For $\mathbb{F}_q$ in order to construct each t_i, we need to compute d_{x_i} inverses in $\mathbb{F}_q$ and $(d_{x_i} - 1)$ comparisons. Hence, for n variable nodes, we execute $(2\omega - n)$ operations. To construct the β_i's, we compute n divisions.

In Step 3, we have n comparisons.

In Step 4, we have at most n comparisons per iteration.

In the worst-case, we need ω operations in Step 5. Therefore, the number N of operations in each iteration is at most

$$N = 5\omega + 2n.$$

This proves the proposition. □

The number of equations is an upper bound on the number of iterations because in each iteration at least one unsatisfied equation node is corrected. Hence, the total number of operations is bounded above by $m \times (5\omega + 2n)$.

7. Conclusions

Our extended bit-flipping algorithms are fast simple algorithms with good performance for sparse systems with a specific form for the Tanner graph called relatively connected.

Moreover, if the Tanner graph of the system is not relatively connected, the algorithm may be used in a probabilistic way. In this case we try several random input vectors whenever Type 1 or Type 2 paths do not exist for the current vector. This strategy is promising because of its low complexity and large applicability. This has a similar effect as in message-passing decoding methods which are proved to work only for cycle-free Tanner graphs but can still be used in graphs with large girth.

References

[1] A. Abolpour, M.-R. Sadeghi, D. Panario, Extended bit-flipping algorithm for solving large sparse linear systems of equations modulo a prime number p, in: Proc. ITW, July 2011, pp. 688–692.
[2] J. Cho, W. Sung, Adaptive threshold technique for bit-flipping decoding of low-density parity-check codes, IEEE Commun. Lett. 14 (9) (2010) 857–859.
[3] D. Coppersmith, Solving linear equations over $GF(2)$: block Lanczos algorithm, Linear Algebra Appl. 192 (1993) 33–60.
[4] D. Coppersmith, Solving homogeneous linear equations over $GF(2)$ via block Wiedemann algorithm, Math. Comp. 62 (1994) 333–350.
[5] D. Coppersmith, A. Odlyzko, R. Schroeppel, Discrete logarithms in $GF(p)$, Algorithmica 1 (1986) 1–15.
[6] A. Das, C.E. Veni Madhavan, Public-Key Cryptography Theory and Practice, Dorling Kindersley, India, 2009.
[7] M. Fossorier, M. Mihaljevic, H. Imai, Reduced complexity iterative decoding of low density parity check codes based on belief propagation, IEEE Trans. Commun. 47 (May 1999) 673–680.
[8] R.G. Gallager, Low Density Parity Check Codes, MIT Press, Cambridge, MA, 1963.
[9] B.A. LaMacchia, A.M. Odlyzko, Solving large sparse linear systems over finite fields, in: A. Menezes, S. Vanstone (Eds.), Advances in Cryptology, CRYPTO '90, in: Lecture Notes in Comput. Sci., vol. 537, Springer-Verlag, 1991, pp. 109–133.
[10] S. Lin, D.J. Costello Jr., Error Control Coding: Fundamentals and Applications, Prentice Hall, ISBN 0-13-283796-X, 1983.
[11] D.J.C. MacKay, R.M. Neal, Near Shannon limit performance of low density parity check codes, Electron. Lett. 32 (18) (1996) 1645–1646.
[12] R.M. Tanner, A recursive approach to low complexity codes, IEEE Trans. Inform. Theory 27 (1981) 533–547.
[13] D.H. Wiedemann, Solving sparse linear equations over finite fields, IEEE Trans. Inform. Theory 32 (1) (1998) 54–62.