Low-complexity Matrix Embedding Using an Efficient Iterative Search Strategy

CHI-YUAN LIN[1] and JYUN-JIE WANG[2]
Department of Computer Science and Information Engineering,
National Chin-Yi University of Technology, Taichung 41170, Taiwan, ROC
Email: [1]chiyuan@ncut.edu.tw,[2]jjwang@ncut.edu.tw

Abstract: This study proposes a novel suboptimal embedding algorithm for binary messages based on a low-weight search embedding (LWSE) strategy. The suboptimal LWSE strategy involves using algorithm to perform an embedding procedure by using a parity check matrix. The optimal embedding algorithm, which is based on the maximun likelihood (ML) algorithm, aims to locate the coset leader and minimize embedding distortion. The optimal embedding based on linear codes can achieve high embedding efficiency but incurs high computation. Conversely, the LWSE does not need to locate the coset leader, but instead requires suboptimal object. Because its corresponding weight remains close to that of the coset leader, the algorithm proceeds in an efficiently iterative manner. When using the optimal ML algorithm for a situation involving the highest operation complexity, the operation complexity of the suboptimal LWSE is linearly proportional to the number of code dimension.

Key-Words: suboptimal embedding algorithm, data hiding, digital watermarking, informed coding, informed embedding, maximun likelihood algorithm.

1 Introduction

In steganography, a cover image is modified to obtain the stego image. High embedding efficiency is required for steganographic schemes. Embedding efficiency, which is defined as the average number of embedded bits per change, is a critical aspect of steganography. One effective steganographic technique involves matrix embedding (ME).

Crandall [1] and Bierbrauer [2] used excellent linear block codes for an ME scheme derived from covering codes [2-4] that can approach a high embedding efficiency. Previous researchers had implemented ME by using the suboptimal embedding algorithm [5-10]. In 2006, Fridrich et al. [5] showed that the security of steganography for large payloads can be improved by using simplex codes and random codes. However, the computational complexity in [5] is high, and is therefore not suitable for real-time applications. In 2007, Li et al. proposed a tree-based parity check (TBPC) algorithm [7], which is a suboptimal embedding algorithm characterized by a tree structure. Although the TBPC algorithm is simple, its embedding efficiency for steganographic schemes is poor. However, the MPC algorithm proposed by [8] further improves the embedding efficiency of the TBPC algorithm. The MPC method can be formulated as ME and to include a tree structure to optimize the ME method. Therefore, the MPC method can be used to produce efficient embedding algorithms. Although the MPC method is an efficient algorithm, its embedding efficiency is poor. [9] proposed an ME method to reduce the computational complexity of random linear code, and to increase the embedding efficiency of the steganographic scheme. [9] used a random linear code by extending the parity matrix via some by using referential columns to achieve high embedding efficiency; however, this process is computationally expensive. Therefore, reducing the computational complexity of the algorithm and steganographic scheme while maintaining high embedding efficiency is a critical problem. The ME method is efficient for reducing the embedding complexity and increasing the embedding efficiency. Alternatively, high-capacity steganographic scheme introduced in [11-12] can generate an efficient embedding method with arbitrary large relative payloads.

This paper presents an analysis of the trade-off between embedding efficiency and computational complexity. The proposed algorithm is a fast and low-complexity algorithm by using the parity check matrix, called low weight search embedding (LWSE) strategy. LWSE algorithm has demonstrated the advantage of high embedding efficiency for small payloads. Another advantage of LWSE is that its implementation is suitable for various embedding rates. This embedding algorithm can be implemented with linear computation complexity. Embedding algorithms based on ME can improve embedding efficiency. The steganographic scheme based on the parity check matrix can be extracted in the receiver by using a multiplication operation that occurs between the parity check matrix and received setgo.

The remainder of this paper is organized as follows: Section 2 reviews several elementary concepts from coding theory, as well as the optimal embedding algorithm; Section 3 provides a description of the major work for suboptimal embedding strategy; Section 4 describes the embedding complexity for several embedding algorithms; Section 5 provides experimental results and constructive discussions; and finally, Section 6 offers the conclusion of this study.

2 The bound on embedding efficiency and optimal embedding

The following discussion is an introduction to the embedding algorithm of linear codes: for an embedding scheme using linear codes, consider an (n,k) linear code C at embedding rate $R_e = (n-k)/n = m/n$. Under the assumption that a logo $s \in \{0,1\}^m$, embedded into a cover $u \in \{0,1\}^n$, is transmitted to the receiver, then the optimal stego $l' = u - e_{opt}$, is provided by an embedder to the syndrome s. These can be formulated as a rate-distortion problem. Assume that the linear codes C at embedding rate R_e correspond to an embedding average distortion $d = E[d(l',u)]/n = E[w(e_{opt})]/n$, where $\omega_H(\cdot)$ denote the Hamming distance. The theoretically achievable bound is $k/n \geq 1 - h(d)$, where $h(d) = d\log_2(1/d) + (1-d)\log_2(1/(1-d))$ denotes a binary entropy function. Thus, an embedding rate bound can be obtained as $h(d) \geq R_e$ and the minimal average distortion d reaches

$$d \geq h^{-1}(R_e), \tag{1}$$

where $0 \leq d \leq 0.5$ and $h^{-1}(*)$ is the inverse binary entropy function. Without loss of generality, define the embedding efficiency as follows:

$$\eta \triangleq \frac{R_e}{d} = \frac{m}{D}, \tag{2}$$

where $D = nd$ is the average embedding distortion per block. According to (1), the result of (2) is an asymptotic upper bound as follows:

$$\eta \leq \frac{R_\varepsilon}{h^{-1}(R_\varepsilon)} = \eta_\delta. \tag{3}$$

The term η_δ represents the bound of embedding efficiency.

Two interval measure parameters are defined as $\varepsilon_{opt} = \eta_\delta - \eta_{opt}$ and $\varepsilon_{sub} = \eta_\delta - \eta_{sub}$, where ε_{opt} is the interval measure between theoretical upper bound and embedding algorithm using optimal decoding. In a like manner, a small value of ε_{sub} leads to an improved efficiency when performing the suboptimal decoding algorithm.

2.1 Optimal embedding algorithm

An algorithm is referred to as a matrix embedding due to the use of a parity check matrix. It is built with two main goals: 1) find a well defined coding structure or a well behaved parity check matrix, and 2) perform decoding through maximum likelihood (ML) decoding. Given a cover sequence and a logo sequence intended for embedding, the syndrome of the cover sequence must be found first and then added to that of the logo sequence to acquire a toggle syndrome. Ultimately, the coset leader corresponding to the toggle syndrome can be found using the ML decoding method. The coset leader is then added to the cover sequence to yield the closest sequence into which a secret logo sequence is embedded.

The following is a review of a few elementary concepts from coding theory that are necessary for the optimal embedding algorithm. A (n,k) linear block code C can be characterized using a parity check matrix H of size $(n-k) \times n$ as follows:

$$C = \{r | Hr = 0\}, \tag{4}$$

where the sequence $r \in F_2^n$. According to this equation, the syndrome s of the sequence r, in the case of a nonzero Hr, is defined as $s = Hr$. Furthermore, the set composed of all the sequences r, corresponding to the identical s, is called the coset of the code C, defined as

$$C^s = \{r | Hr = s\} = \{c + e | c \in C\}, \tag{5}$$

where e denotes the coset leader in the standard array. The term s can be derived from an arbitrary sequence r through H, and e can be expressed, in terms of an ML decoding function, as

$$e = f(Hr) = f(s), \tag{6}$$

where $f(\cdot)$ represents the linear code decoding function. The coset leader e, determined through ML decoding, is added to r to recover the code C that is closest to the sequence r.

Consider the embedding process illustrated in Fig. 1. In this figure, the cover sequence corresponds to an arbitrary sequence u of length n bits within the coset C^u of the standard array. The syndrome $s_u = Hu^T$ corresponding to C^u is called the cover sequence syndrome. A known binary sequence s_l of length $n-k$ bits, called the logo sequence, is intended for embedding. The coset leader e_{opt} must be located within a set C^x The syndrome s_x is then determined by the addition of logo sequence s_l to s_u. From a decoding viewpoint, the coset leader e_{opt} can be discovered through maximuml likelihood (ML) decoding, expressed as

$$e_{opt} = f_{opt}(s_u + s_l) = f_{opt}(s_x). \tag{7}$$

Suppose that a sequence $x \in C^x$ exists, and C^x represents a coset of the code C. It is intended to seek x with the minimal weight, that is, $x = e_{opt}$, which is expressed as

$$e_{opt} = \underset{x \in C^x}{\operatorname{argmin}} w_H(x). \tag{8}$$

Once discovered, the coset leader e_{opt} is added to the cover sequence u as $l' = u + e_{opt}$. Essentially, l' is the sequence, closest to the sequence u within F_2^n dimensional space, and contains the logo sequence s_l. The following discussion presents a way to embed a binary linear code through ML decoding based on the standard array shown in Fig. 1.

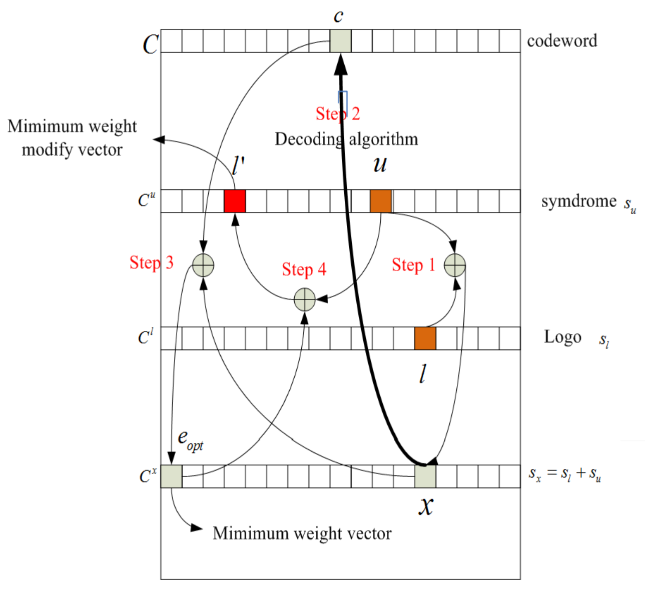

Fig. 1: Standard array for embedding procedures.

The optimal embedding algorithm to embed a binary message sequence is illustrated as follows:

```
========================================
Algorithm  Optimal embedding algorithm: Given a symbol $s_l$ and a cover sequence $u$, a sequence $l'$, closet to the sequence $u$, corresponding to the syndrome $s_l$ is located as follows.
----------------------------------------
Step 1: Derived from $s_l$, the sequence $l$ in $C^l$, is added to $u$ so as to obtain $x$.
Step 2: The sequence $x$ is then decoded through ML algorithm into a codeword $c$ as follows
        $\hat{c} = \text{argmin}_{c \in C} d(c, x)$.
Step 3: The addition of $x$ to $\hat{c}$ yields $e_{opt}$.
Step 4: $l'$ is obtained by adding $e_{opt}$ to $u$.
Step 5: he data embedded is then extracted by performing
        $s_l = Hl'$.
========================================
```

Once e_{opt} is known, the optimal embedding sequence l' can be discovered. However, finding e_{opt} remains difficult in the case of the (n, k) linear codes C with a large value of k because the complexity of the ML decoding increases when 2^k. The following section of this paper presents a suboptimal embedding algorithm to replace the optimal embedding algorithm and to help resolve the disadvantage mentioned above.

3 Suboptimal embedding algorithm

This section presents an efficient algorithm perform the secret hiding. As illustrated in the preceding section, it is unrealistic to perform ML decoding for an arbitrary large linear block code. Here, the coset leader is found in an alternative way to the conventional ML decoding. Instead, a simple method is used to locate a toggle sequence with a lower weight during the search of coset leaders e_{opt}. The proposed algorithm is intended to locate a sequence e_{sub}, and $w_H(e_{sub}) \geq w_H(e_{opt})$, in lieu of the optimal coset leader $w_H(e_{opt})$. However, e_{sub} must stay as close to $w_H(e_{opt})$ as possible. Note that e_{sub} is a sequence defined in C^x. The stego sequence l', which is obtained by the addition of e_{sub} to the cover sequence u, cannot be assured as the optimal sequence. As Fig. 2 shows, this case is referred to as the suboptimal embedding algorithm.

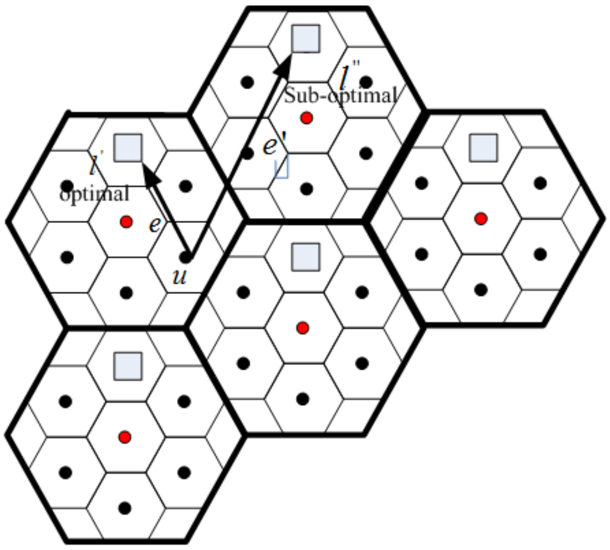

Fig. 2: Geometric interpretation of sub-optimal information embedding

3.1 Embedding using code with systematic form

This subsection presents a matrix embedding scheme based on systematic linear block codes. To reduce the computational complexity of the embedding procedure, use the systematic parity check matrix to obtain a toggle sequence. Consider a systematic parity check matrix $H = [I \quad H_p]$. Suppose that there exists a logo sequence l of length n bits within the coset C^l of the code C. Regard s_l as the logo sequence of length m bits intended for embedding. With k number of 0's added to its right, a sequence l of length n bits is formed as $l = [s_l 0 \cdots 0]$ and $Hl^T = s_l$. The sequence $l \in C^l$, which corresponds to the sequence of length m bits, can thus be found. For systematic reasons, a sequence $l \in F_2^n$, corresponding to the syndrome s_l, can be discovered with ease. As Fig. 1 shows, the addition of l to the cover sequence $u \in F_2^n$ yields $x \in C^x$, which is decoded as an approach to acquiring e_{opt} within C^x.

3.2 Weight Approach Strategy

Although the ME is a simple and low-complexity method, this paper further proposes an embedding algorithm based on the parity check matrix. The proposed algorithm not only has low complexity but also performs the algorithm calculations quickly. This section also presents several feasible binary data embedding algorithms and states the suboptimal algorithm, that is, the LWSE algorithm. Unlike the ML embedding algorithm, this iterative technique is used to search a minimum weighted toggle sequence. Specifically, in the ML algorithm, the minimum weight is obtained through a full search of codewords within the linear embedding code. However, the LWSE algorithm seeks the lower-weighted toggle sequence in several iterations. By performing hardware operations in each iteration, LWSE proceeds in an iterative manner until it reaches convergence. The LWSE algorithm provides a lower operation complexity than the ML algorithm, but this comes at the cost of lower distortion efficiency.

The cover sequence is used to carry the logo sequence in ME, which is based on the parity check matrix. Assume that generating a binary matrix H of size $m \times n$ consists of two parts as follows:

$$H = [I \quad H_p]. \tag{9}$$

The toggle sequence x is divided into two parts as $x = (x_M, x_P)$, where $x_M = (x_1, x_2, \cdots, x_m)$ and $x_P = (x_{m+1}, x_{m+2}, \cdots, x_n)$. To locate the low weight of toggle sequence x, toogle sequence x must meet the condition

$$s_x = Hx^T = x_M + H_p x_P^T. \tag{10}$$

The weight $w_H(x)$ is a suboptimal candidate and can be expressed as

$$w_H^{sub}(x) = \arg\min_{\forall x_M, x_P} (w_H(x_M) + w_H(x_P))$$

$$= \underset{\forall x_M}{\operatorname{argmin}} w_H(x_M) + \underset{\forall x_P}{\operatorname{argmin}} w_H(x_P). \tag{11}$$

To minimize the weight of x, (11) must simultaneously minimize the weight of sequence x_M and x_P. This study presents the LWSE algorithm to obtain a toggle sequence x with a low weight. The LWSE algorithm uses two steps to minimize (11). The first is to minimize x_P. A systematic encoder is applied to the toggle sequence x, which is formed as the sequence $x = (x_M, 0)$, where $x_M = H(l+u)^T = s_l + s_u$. According to (10), the x_P is all zeros that meet the requirement, and x_M is equal to s_x. The second step is to minimize $x_M = s_x$ based on the part of the parity check matrix $H_p = [h_{m+1}, \cdots, h_n]$. By using (10), let $x_P^{(i)}$ sequence of k bits is as following:

$$x_p^{(k)} = e_k = (0,0,\cdots,1)$$

$$x_p^{(1)} = e_1 = (1,0,\cdots,0)$$

$$x_p^{(2)} = e_2 = (0,1,\cdots,0)$$

$$\vdots$$

$$x_p^{(k)} = e_k = (0,0,\cdots,1) \tag{12}$$

To satisfy (10), column h_i, which corresponds to e_i out of matrix H_p, is added to toggle sequence $x_M = s_x$ as $x_{M'}$; that is, $x_{M'} = x_M + h_i$. Consequently, the weight of $x_{M'}$ is altered through the columns of H_p, and $x_{M'}$ still falls within coset C^x. Therefore, the solutions of (10) can be constructed as follows:

$$s_x = H(x_{M'}, x_p^{(i)})^T$$

$$= I(x_{M'})^T + H_p e_i^T$$

$$= (x_M + h_i) + h_i$$

$$= x_{M'} \tag{13}$$

where the newly modified toggle sequence is $x_{sub} = (x_{M'}, x_p^{(i)})$.

Although the optimal solution is based on the combination of arbitrary columns of H_p, it is unrealistic to choose all of the columns. For the LWSE algorithm, only k number of columns is selected from among the 2^k column sets. By adding h_i to x_M, the toggle sequence $x_{sub} = (x_M + h_i, e_i)$ can reduce more changes than the original toggle sequence $x = (x_M, 0)$. The changes can eventually be improved by a small weight variation. The modified toggle sequence $x_{sub} = (x_M + h_i, e_i)$ is gained through an appropriate weight variation of toggle sequence x, with the primary goal of obtaining the weight of x_{sub} of that closest to the coset leader. This toggle sequence can be expressed as

$$x_{sub} = \underset{\forall h_i}{\operatorname{argmin}} w_H(x_M + h_i) + w_H(e_i)$$

$$= \underset{\forall h_i}{\operatorname{argmin}} w_H(x_M + h_i) + 1, \tag{14}$$

where the modified toggle sequence x_{sub} based on the sequence $\{e_i, i = 1,2,\cdots,k\}$ has the minimum changes after k tests. In addition, if $w_H(x_{opt}) \le w_H(x_{sub}) \le \mu$, where μ is a constant, then the changes of modified toggle sequence x_{sub} remain closer to the optimal solution x_{opt}. This LWSE algorithm is capable of minimizing the changes of toggle sequence in an iterative manner. This algorithm can described as follows:

====================================

LWSE algorithm: Given a (n,k) systematic linear code C with $H = [IH_p] = [i_1,\cdots,i_m,h_1,h_2,\cdots,h_k]$, a symbol s_l of m bits and a cover sequence u of n bits, a sequence l' of n bits, closest to the sequence u, corresponding to the syndrome s_l is located as follows.

Step 1: First, evaluate the sequence x_M corresponding to the syndrome s_x.

$$s_M = Hu^T + s_l^T$$

Step 2: Let $x_P = (0,\cdots,0)$ and Evaluate an initial toggle sequence

$$x = (s_M x_P) = (x_M 0 \cdots 0).$$

Step 3: Evaluate the modified sequence $x_{M'}$ using tests of k times with (13)

$$x_{M'} = x_M + H_p e_i^T \text{ and } x_P' = x_P + e_i.$$

and evaluate the modified toggle sequence with (14)

$x' = \text{argmin}_{\forall x_{M'}} w_H(x_{M'}) + 1.$

Step 4: If $w_H(x') < w_H(x)$, then let $x_M = x_{M'}$, $x_p = x_{p'}$ and skip to Step 3. Otherwise, proceed to the next step.

Step 5: Estimate $l' = x' + u$, and output l'.

Step 6: $H(l')^T = s_l$ is executed when extracting at the receiver.

=====================================

Consider an example with random codes to show test the LWSE algorithm. Take an $(8,4)$ random code, cover sequence $u = (11111000)$, logo sequence $s_l = (1100)$. The parity check matrix is

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix} = [I \;\; H_p].$$

After conducting Steps 1 and 2 of the LWSE algorithm, the toggle sequence is $x = (x_M 0000) = (11110000)$, where $x_M = (0011) + (1100)$. For Step 3 in the LWSE algorithm, the modified toggle sequence in the first iteration is evaluated as follows:

$$x_{M'} = x_M + H_p e_1^T = (1111)^T + (1100)^T = (0011)^T$$

$$x_{M'} = x_M + H_p e_2^T = (1111)^T + (0111)^T = (1000)^T$$

$$x_{M'} = x_M + H_p e_3^T = (1111)^T + (0011)^T = (1100)^T$$

$$x_{M'} = x_M + H_p e_4^T = (1111)^T + (1000)^T = (0111)^T.$$

These calculations produce four weights $\{3,2,3,4\}$, in which the minimal weight 1 is calculated from $x' = \text{argmin}_{\forall x_{M'}} w_H(x_{M'}) + 1$. The minimal weight of modified $x_{M'}$ is (1000) and the new toggle sequence is $x' = (x_{M'} e_2) = (10000100)$. For the condition $w_H(x') < w_H(x)$, the second iteration can be evaluated as follows:

$$x_{M'} = x_M + H_p e_1^T = (1000)^T + (1100)^T = (0100)^T$$

$$x_{M'} = x_M + H_p e_2^T = (1000)^T + (0111)^T = (1111)^T$$

$$x_{M'} = x_M + H_p e_3^T = (1000)^T + (0011)^T = (1011)^T$$

$$x_{M'} = x_M + H_p e_4^T = (1000)^T + (1000)^T = (0000)^T.$$

In the second iteration, the Step 4 of the LWSE algorithm is not satisfied. Finally, we obtain the modified toggle sequence $x' = (00000101)$ and the result in the LWSE algorithm is

$l' = u + x' = (11111000) + (00000101) = (11111101)$

In the receiver, the log message can be extracted by $H(l')^T = (0011)^T$.

4 Complexity for suboptimal embedding algorithm

An (n,k) random linear code suffers the greatest disadvantage in embedding complexity in a high k dimensional space. [5] proposed a way to improve embedding efficiency based on ME for large payloads using simplex codes and random codes. If the embedding rate of linear codes is large, the dimension of linear codes is small enough to enable fast decoding. The coset leader can also be found in ML decoding. If the code dimension is large, ML decoding, that is, the optimal embedding algorithm, is unrealistic. Researchers have presented several suboptimal embedding algorithms, including [7] and [8] , [9]. These suboptimal algorithms can reduce the embedding complexity much more than the ML embedding algorithm, at the expense of embedding efficiency. Specifically, the complexity of [7] can reach up to $O(m)$ or $O(m\log_N m)$, and [8] is required to locate the majority vote value, which results increases the time complexity. However, the LWSE suboptimal embedding algorithm requires that complexity of $O(\lambda k)$ be associated with the iteration times λ. As stated in [5], an (n,k) random linear code can be used to gain an improved embedding efficiency for a large length, and it is enabled to perform the LWSE algorithm in a large code dimension.

Table 3 lists the embedding times and embedding efficiency for various suboptimal embedding algorithms under the same relative payload. [5] proposed two embedding methods based on random linear codes and simplex codes. In this case, the time complexity of embedding algorithms for matrix embedding is bounded by the complexity of the decoding algorithms for the linear codes; that is, the complexity of finding the coset leader. The decoding algorithms for (n,k) simplex codes in [5] have a time complexity of $O(n\log n)$. Although the embedding efficiency of [5] is near the upper bound for large relative payloads, the time complexity is still prohibitive. [7] and [8] have proposed suboptimal embedding algorithms to improve the embedding complexity. [7] proposed a scheme to reduce embedding changes based on a tree structure. This method is also represented as the ME method, which was improved by [8] using the majority-vote parity check (MPC). Although [7] and [8] can be used to achieve low embedding complexity, their embedding efficiency is poor. [9] proposed a fast embedding by appending columns to the parity check matrix. This method can significantly reduce the computational complexity compared with existing ME, which involve using random codes [5]. The extended matrix in [9] can be used to increase the embedding efficiency by using ML decoding, and the computational complexity of [9] can exponentially increase from $O(n2^{k})$ to $O(n2^{k+h})$. By using the proposed algorithm in [9], it must only search for a small solution space with a small size of the extended matrix and then the computational complexity is equal to $O((n+h)2^{k})$, which linearly increases with h; thus this method has a faster embedding speed compared with the original ME [5]. Based on the method presented in [9], the embedding complexity is further improved using a low-weight search method for determining the toggle sequence. This study proposes the LWSE method to achieve an embedding algorithm with low complexity. Random codes and Hamming codes are used to embed at fixed relative payloads. Because the proposed algorithm uses (n,k) linear codes with parity check matrix of size $(n-k)\times n$, the memory requires $(n-k)\times kn$ in storage. Thus, the order of computations is $O(\lambda k)$. However, to keep the complexity and memory requirement slow, the number of toggle sequences in the LWSE is low. Because the LWSE searches only a few search column vectors of the parity check matrix, the computational complexity requires that $O(\lambda k)$, where λ is a constant.

5 Simulation results

The algorithms presented in this study were simulated on the operational time and embedding efficiency. The programs were developed in MATLAB-R2007, and executed on a CPU-Intel E8300 2.83G computer with 2G DRAM. In this experiment, the cover and logo sequence were selected randomly. The cover was divided into non-overlapping blocks, and each block was embedded with the logo sequence of m bits.

Table 1 shows the results for the $(32,16)$ random linear code. The complexity of the LWSE algorithm with respect to average iterative times λ running the algorithm was $O(\lambda k)$. Table 1 illustrates the results of performing the LWSE algorithm, indicating the number of iterations at various iterative times by comparing the complexity regarding the number of iterations. These results show that the majority of iterations of the LWSE suboptimal algorithm was approximately 8 times. These experimental results show that the average number of iterations for each block was approximately to 7.142 when applying the LWSE algorithm.

The following discussion presents is a CPU time comparison between the performance of the LWSE and optimal embedding algorithms on a (40, 20) random linear code, a constant embedding capacity $R_{e}=0.5$, and a dimension k ranging from 14 to 20. In Table 2, the complexity required in the optimal embedding algorithm varies exponentially with the dimension k of the code, whereas this value varies linearly with λk in the LWSE algorithm. Although performing the optimal embedding algorithm for a large value of k in unlikely reality, it is feasible, when performing the LWSE algorithm, to discover the toggle sequence. In Table 2, performing the optimal embedding algorithm on a (40, 20) random linear code requires 68.89 sec, while in dimension $k = 20$, it takes only 0.0021 sec, far below 68.89 sec, to perform LWSE algorithm.

The LWSE algorithm has a faster computational time compared with other embedding algorithms. The mentioned algorithms are simulated to determine their embedding efficiency. The simulation results show that the LWSE algorithm requires a lower operational complexity than does the ML algorithm at the cost of degraded efficiency. The embedding efficiency corresponding to various systematic linear block codes is comparable with both the LWSE and numerous suboptimal algorithms. The proposed method can be used to embed messages for linear codes with large dimensions; that is, embedding for a small payload. The LWSE algorithm can also be used to embed messages at various embedding rates. Considering the low rate in Fig. 3, the LWSE algorithm still follows the upper bound for embedding efficiency.

6 Conclusion

Performing ML decoding in an optimal embedding algorithm is unrealistic because the decoding complexity increases exponentially with code dimension $k = n - m$ as $O(2^k)$. This paper proposes a suboptimal embedding algorithm, called the LWSE algorithm, to reduce embedding complexity. Although the proposed scheme demonstrates decreased embedding efficiency compared with using ME with ML decoding, it is also suitable for various embedding rates and reduces the computational complexity. Moreover, the proposed algorithm demonstrates high embedding efficiency for embedding in small payloads compared with other embedding algorithms, as shown in Table 3. In the experiment, random codes and Hamming codes were used to implement the LWSE algorithm. Although LWSE can cause a loss of some embedding efficiency because ME involves using a suboptimal algorithm, the experimental results of this study confirm that the LWSE algorithm demonstrates lower computational complexity compared with other embedding schemes.

Acknowledgment

This study is supported in part by the National Science Council of the Republic of China under contract numbers NSC-101-2221-E-167-026 and NSC-102-2221-E-167-001.

References

[1] R. Crandall. Some notes on steganography. *Steganography Mailing List*, 1998.

[2] J. Bierbrauer, On Crandall's Problem [Online]. Available: http://www.ws.binghamton.edu/fridrich/covcodes.pdf 1998.

[3] F. Galand and G. Kabatiansky. "Information hiding by coverings," In *Proceedings ITW2003, Paris, France, 2003*, pp. 151-154.

[4] J. Bierbrauer and J. Fridrich, "Constructing good covering codes for applications in steganography," *LNCS Transactions on Data Hiding and Multimedia Security*, vol. 4920, pp. 1-22, 2008.

[5] J. Fridrich and D. Soukal, "Matrix embedding for large payloads," *IEEE Trans. Inf. Theory*, vol. 1, no. 3, pp. 390-395, Sep. 2006.

[6] R. Y. M. Li, O. C. Au, C. K. M. Yuk, S. K. Yip, and S. Y. Lam, "Halftone Image Data Hiding with Block-Overlapping Parity Check," *Proc. IEEE*, vol. 2, pp. 193–196, Apr. 2007.

[7] R. Y. M. Li, O. C. Au, K. K. Lai, C. K. M. Yuk, and S. Y. Lam, "Data Hiding with Tree Based Parity Check," *IEEE International Conference*, pp. 635-638, Jul, 2007.

[8] C. Hou, C. Lu, S. Tsai, and W. Tzeng, "An optimal data hiding scheme with tree-based parity check," *IEEE Trans. Image Pro.*, vol. 20, no.3, pp. 880-886, Mar. 2011.

[9] C. Wang, W. Zhang, J. Liu, and N. Yu,"Fast Matrix Embedding by Matrix Extending," *IEEE Trans. Inf. Theory*, vo7. 1, no. 1, pp. 346-350, Feb. 2012.

[10] S.M. Walke, V.G. Puranik, J.G. Rana, and R.S. Deshpande, "A Novel Technique of Steganography And Watermarking," *National Conference on Signal and Image Processing Applications*, pp. 22-29, Jan. 2009.

[11] G. Brisbane, R. Safavi-Naini, and P. Ogunbona,"High-capacity Steganography Using A Shared Colour Palette," *IEE Proceedings. Image and Signal Processing*, vol. 152, no. 6, pp. 787-792, Dec. 2005.

[12] Y.K. Lee and L.H. Chen,"High capacity image steganographic model," *IEE Proceedings. Image and Signal Processing*, vol. 147, no. 3, pp. 288-294, June 2000.

Table 1: The number of iteration times versus iteration times for peforming (32,16) Random code using LWSE algorithm.

Iteration times	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Number of iteration times	3	13	50	187	341	334	74	8

Table 2: The optimal embedding algorithm and LWSE algorithm consume in time.

Information bits	k=14	k=15	k=16	k=17	k=18	k=19	k=20
(40, k) random code using ML algorithm (in second)	1.03	3.02	6.33	8.74	16.54	34.37	68.89
(40, k) random code using LWSE algorithm (in second)	0.01	0.015	0.0094	0.016	0.014	0.032	0.0021

Table 3: Performance comparisons of various embedding algorithms

Code	$1/R_e$	ε_{sub}	sec
Hamming(31, 5)	6.36	1.51	1.93
[6](Lx=32, *n*=4225)	4.48	2.51	2.05
[6](Lx=64, *n*=16641)	4.38	2.49	1.81
[6](Lx=128, *n*=66049)	4.34	2.33	1.55
Random(16, 4)	4	2.75	0.48

Random_ex(160, 40)	4	2.68	4.41
Hamming(15, 4)	3.64	1.66	1.39
[7](*n*=7, 4)	1.88	1.43	0.77
[7](*n*=15, 8)	1.75	1.67	0.4
[7](*n*=31, 16)	1.96	1.68	0.21
BCH(15, 8)	1.88	1.39	0.69
Golay(24, 12)	2	1.48	0.81

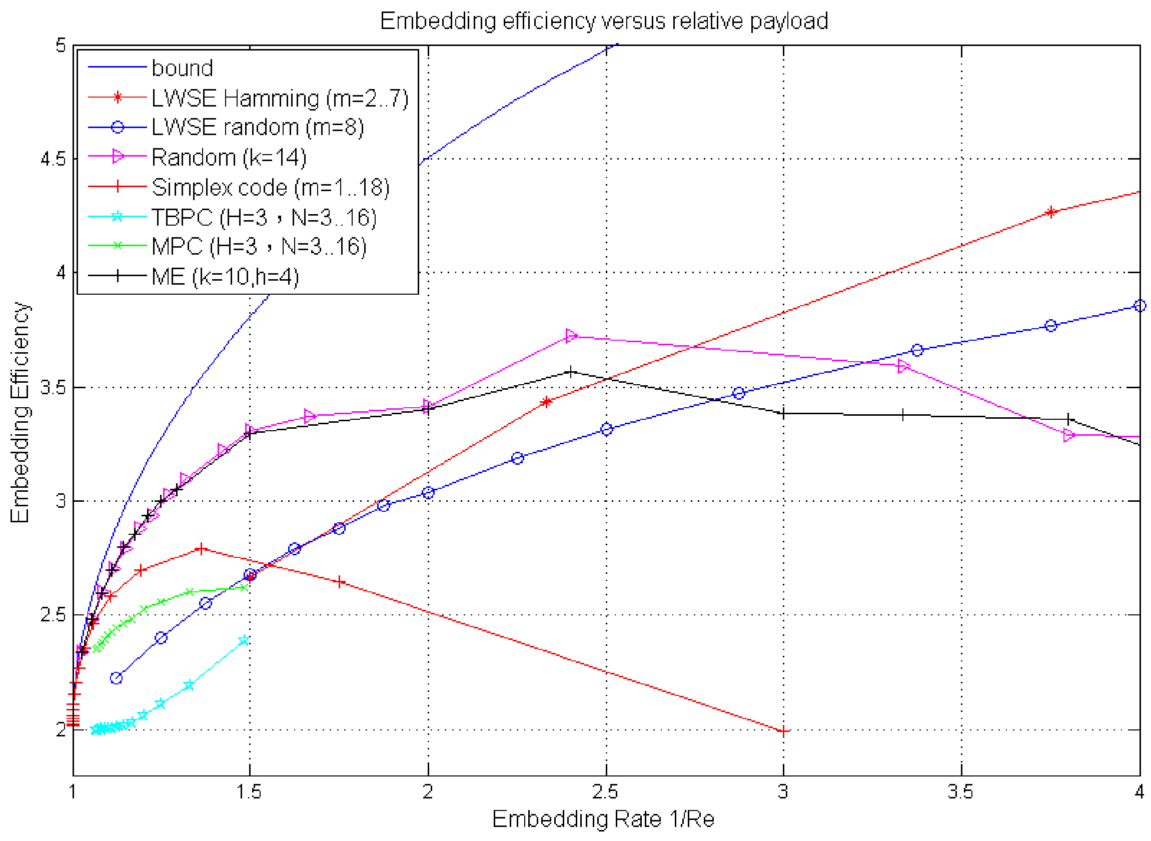

Fig. 3: Embedding effciency of various algorithms.