Low-complexity Data Hiding Algorithms with Graph-based Parity Check

Jyun-Ying Huang and Yuh-Ming Huang

Dept. of Computer Science and Information Engineering

National Chi Nan University

Puli, 545, Nantou, Taiwan, ROC

Email: { s100321520@ncnu.edu.tw ; ymhuang@csie.ncnu.edu.tw }

Abstract—This paper presents two low-complexity steganographic algorithms based on the proposed graph-based parity check methodology for gray-scale images. Both algorithms embed a segment of *n* secret bits into *n*+1 and *n* cover pixels, respectively. The performances of both algorithms are verified by theoretical distortion analysis. Algorithm1 can generate an optimal toggle sequence with the least number of 1's under the graph-based parity check model. Algorithm 2 can further enhance the embedding rate and the embedding efficiency of Algorithm1 by the technique of ±1 steganography. The theoretical distortion analyses show that both algorithms are better than the other schemes.

Index Terms—data hiding; tree-based parity check; *±1 steganography*; embedding rate; embedding efficieiency

I. INTRODUCTION

Data (or information) hiding is one kind of *steganographic techniques* to embed the secret information into a cover host, such as an image. Usually, the naked eye of the people cannot perceive any change when the image is modified slightly. A range of applications, including anticounterfeiting and authentication, is explored in [1].

In 1998, Crandall [2] first introduced the idea of matrix embedding. The technique of matrix encoding (or called syndrome coding [3]) was deeply studied in [4-6]. In [5], Zhang *et al.* utilized the available information capacity of "±1 steganography" to enhance the embedding efficiency (the average number of embedded bits per modification; denoted as *E*). This work was extended in [10]. Zhang *et al.* [7] generalized the idea of matrix embedding and defined the steganographic matrix *H* and its corresponding linear steganographic code.

In 2006, Zhang *et al.* [8] proposed a data hiding scheme by exploiting modification direction (EMD), in which each secret digit in a (2*n*+1)-ary notational system is carried by *n* cover pixels and, at most, only one pixel is increased or decreased by 1. Hence, this scheme can provide high stego-image quality, but limited hiding capacity. Basically, a trade off exists between the embedding rate (the average number of embedded bits per pixel; denoted as *R*) and the stego-image quality. In 2010, Chang *et al.* [9] proposed a flexible EMD-based data hiding scheme. Recently, Huang *et al.* [10] further improved it.

Computational complexity is also an important issue for data hiding while applied to real-time applications. Based on a tree structure, Li *et al.* [11] proposed a low-complexity data hiding scheme with tree-based parity check (TBPC). Hou *et al.* [12] proposed a majority vote strategy (MPC) to further reduce the embedding distortion (the average number of embedding modifications per pixel; denoted as *D*). Combining the ideas of [8] and [11], another way of tree-based embedding scheme, called multi-ary tree coding (MTC) with each tree node representing one (2*n*+1)-ary digit, was presented in [13].

In this paper, we propose a new idea of *graph-based parity check (GBPC)*. Based on it, two low-complexity GBPC-based data hiding algorithms are presented.

II. GRAPH-BASED PARITY CHECK

Suppose we want to embed *n* secret bits ($s_1, s_2, \ldots, s_n$) into *n*+1 cover pixels ($P_1, P_2, \ldots, P_{n+1}$), where *n* is even.

A. A Low-complexity least significant bit steganographic scheme with R = n/n+1

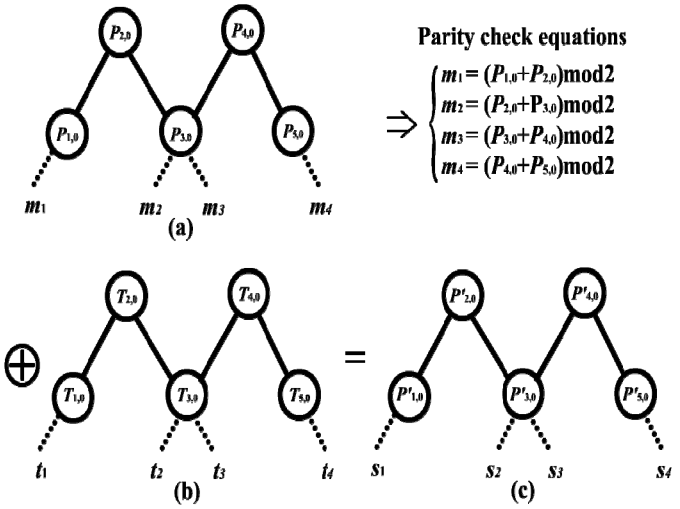

Fig. 1 For n = 4 and the embedding rate R = 0.8. (a) Master graph and its corresponding master sequence and master string (b) Toggle graph and its corresponding toggle sequence and toggle string (c) Stego graph and its corresponding stego sequence and secret string

< *Algorithm* 1 >

Embedding Process:

Step1. According to the *master sequence* ($P_{1,0}, P_{2,0}, \ldots, P_{n+1,0}$), generate the *master string* ($m_1, m_2, \ldots, m_n$) with $m_i = (P_{i,0} + P_{i+1,0}) \bmod 2$ for $1 \leqq i \leqq n$.

Step2. Perform the bitwise exclusive-or operation between the *master string* and the *secret string* to obtain the *toggle string*, i.e. $(t_1, t_2, \ldots, t_n) = (m_1, m_2, \ldots, m_n) \oplus (s_1, s_2, \ldots, s_n)$.

Step3-1. Derive the toggle sequence ($T_{1,0}, T_{2,0}, \ldots, T_{n+1,0}$) with $T_{1,0} = 0$ and $T_{i,0} = (T_{i-1} + t_{i-1}) \bmod 2$ for $2 \leqq i \leqq n+1$.

Step3-2. If the number of 1's in the toggle sequence is greater than $n/2$, then flip each $T_{i,0}$ for $1 \leqq i \leqq n+1$.
Step4. Perform the bitwise exclusive-or operation between the *master sequence* and the *toggle sequence* to obtain the *stego sequence*, i.e. $(P'_{1,0}, P'_{2,0}, \ldots, P'_{n+1,0}) = (P_{1,0}, P_{2,0}, \ldots, P_{n+1,0}) \oplus (T_{1,0}, T_{2,0}, \ldots, T_{n+1,0})$.

Extraction Process:
According to the *stego sequence* $(P'_{1,0}, P'_{2,0}, \ldots, P'_{n+1,0})$, the *secret string* $(s_1, s_2, \ldots, s_n)$ with $s_i = (P'_{i,0} + P'_{i+1,0}) \bmod 2$ for $1 \leqq i \leqq n$ can be extracted directly.

In step 1, a master graph is constructed, as shown in Fig. 1 (a). The *master sequence* $(P_{1,0}, P_{2,0}, \ldots, P_{n+1,0})$ are sequentially filled into the graph nodes from left to right, where $P_{i,0}$ denotes the least significant bit (LSB) of pixel P_i. Then, the master string $(m_1, m_2, \ldots, m_n)$ is generated according to the parity check equations of Fig. 1 (a). In step 2, GBPC derives the *toggle string* $(t_1, t_2, \ldots, t_n)$ by performing the bitwise exclusive-or operation between the *master string* and the *secret string*. In step 3, the toggle graph is constructed with the least number of 1's in the toggle sequence $(T_{1,0}, T_{2,0}, \ldots, T_{n+1,0})$, as shown in Fig. 1 (b). In step 4, the stego graph and its stego sequence $(P'_{1,0}, P'_{2,0}, \ldots, P'_{n+1,0})$ are constructed by performing the bitwise exclusive-or operation between the *master sequence* and the *toggle sequence*, where $P'_{i,0}$ denotes the LSB of stego pixel P'_i.

Example 1: Suppose a block of five pixels $\{P_1, P_2, P_3, P_4, P_5\}$ with respective pixel values = {237, 232, 233, 236, 234} and the secret binary string $\{s_1, s_2, s_3, s_4\}$ = {0, 1, 1, 0}. Hence, we have the least significant bit (LSB) sequence $\{P_{1,0}, P_{2,0}, P_{3,0}, P_{4,0}, P_{5,0}\}$ = {1, 0, 1, 0, 0}. The embedding and extraction processes are illustrated in Fig. 2.

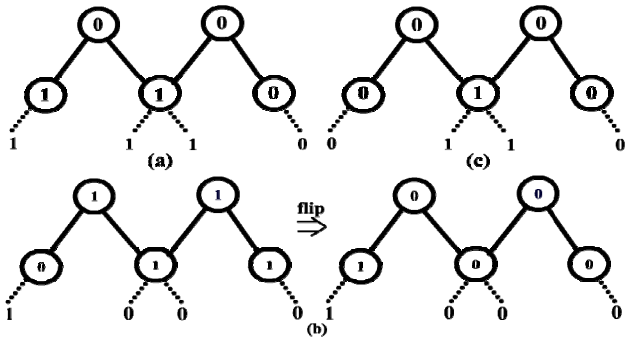

Fig. 2 (a) Master string = (1, 1, 1,0) (b) The construction of a toggle graph with the least number of 1's in the LSB sequence for toggle string = (1, 0, 0, 0) (c) Stego graph is obtained by just flipping the LSB of pixel P_1. Secret string = (0, 1, 1,0).

Thoerem II.1: Algorithm 1 generates an optimal toggle sequence with the least number of 1's under the Graph-based Parity Check model.
Proof:
Assume the toggle sequence $(T_{1,0}, T_{2,0}, \ldots, T_{n+1,0})$ is not optimal. Then, there will exist a better toggle sequence $(T'_{1,0}, T'_{2,0}, \ldots, T'_{n+1,0})$ with $\sum_{i=1}^{n+1} T'_{i,0} < \sum_{i=1}^{n+1} T_{i,0}$ and
$T'_{i,0} + T'_{i+1,0} = T_{i,0} + T_{i+1,0} = t_i$ for $1 \leqq i \leqq n$. (1)
Consider the following two cases:
Case I: $T'_{1,0} = T_{1,0}$
Then, by (1), $T'_{i,0} = T_{i,0}$ for $2 \leqq i \leqq n$, hence $\sum_{i=1}^{n+1} T'_{i,0} = \sum_{i=1}^{n+1} T_{i,0}$, which contradicts the assumption.
Case II: $T'_{1,0} \neq T_{1,0}$
Then, by (1), $T'_{i,0} \neq T_{i,0}$ for $2 \leqq i \leqq n$, hence $T'_{i,0} = (T_{i,0} + 1) \bmod 2$. Therefore, $\sum_{i=1}^{n+1} t'_i = \sum_{i=1}^{n+1} [(t_i + 1) \bmod 2] = (n+1) - \sum_{i=1}^{n+1} t_i > \sum_{i=1}^{n+1} t_i$. This also contradicts the assumption.
By case I and case II, the theorem is proved. □

From the proof of Theorem II.1, we have the following corollary.

Corollary II.1: In GBPC, there exists two toggle sequences associated with a toggle string and algorithm 1 can get the better one through the operation of flipping.

Definition II.1: For a steganographic algorithm that only adopts the technique of LSB replacement. Then we say that the algorithm is perfect if it satisfies $2^s = 1 + \sum_{i=1}^{w} \binom{m}{i}$, where m is the number of cover pixels, s is the number of secret string bits, and w is the maximum number of modified pixels.

Obviously, if the algorithm is perfect, then the algorithm is optimal in terms of the least number of modified pixels.

Lemma II.1: Suppose the probability of each secret string bit with the value of 1 or 0 is equal to 1/2. Whatever the probability of each master string bit 1 or 0 is, the probability of each toggle string bit with the value of 1 or 0 will be equal to 1/2.
Proof: Assume the probability of the occurrence of *master string bit* 1 is p, where $0 \leqq p \leqq 1$. Then, the probabilities of toggle string bit with the value of 1 and 0 are respectively equal to p/2+(1-p)/2 and (1-p)/2+p/2, which are all equal to 1/2. □

Theorem II.2: Algorithm1 is perfect and its embedding efficiency is equal to $n2^n / \sum_{i=1}^{\frac{n}{2}} i \binom{n+1}{i}$.
Proof: The first part can be easily proved by binomial series $(a+b)^n = \sum_{i=0}^{n} \binom{n}{i} a^{n-i} b^i$. That is, we have $2^n = \sum_{i=0}^{\frac{n}{2}} \binom{n+1}{i}$. By Lemma II.1, the expected number (M_E) of modified LSBs is equal to $\sum_{i=1}^{\frac{n}{2}} i \binom{n+1}{i} / 2^n$. Hence, the average distortion $D = M_E / (n+1)$ and the embedding efficiency $E = R/D = n / M_E = n2^n / \sum_{i=1}^{\frac{n}{2}} i \binom{n+1}{i}$, where the embedding rate $R = n/n+1$. □

Suppose we want to embed $n+1$ secret bits (s_1, s_2, ..., s_{n+1}) into n+1 cover pixels ($P_1, P_2, \ldots, P_{n+1}$), where n is even.

B. A low-complexity ±2 steganographic scheme with R = 1

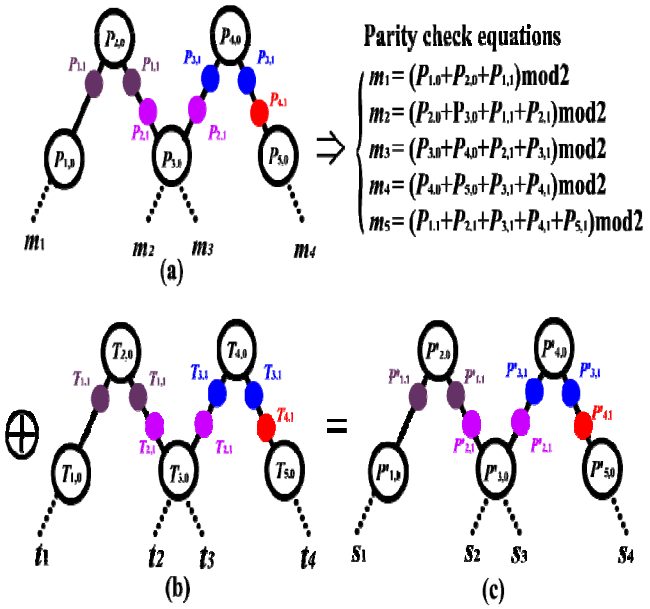

Fig. 3 For n = 4 and the embedding rate R = 1. (a) Master graph and its corresponding master sequence and master string (b) Toggle graph and its corresponding toggle sequence and toggle string (c) Stego graph and its corresponding stego sequence and secret string

< *Algorithm* 2 >

Embedding Process:

Step1. According to the *master graph,* we generate the *master string* $(m_1, m_2, \ldots, m_n)$ with $m_1 = (P_{1,0} + P_{2,0} + P_{1,1})\bmod 2$ and $m_i = (P_{i,0} + P_{i+1,0} + P_{i-1,1} + P_{i,1})\bmod 2$ for $2 \leqq i \leqq n$.

Step2. Perform the bitwise exclusive-or operation between the *master* and *secret strings* to obtain the *toggle string*, i.e. $(t_1, t_2, \ldots, t_n) = (m_1, m_2, \ldots, m_n) \oplus (s_1, s_2, \ldots, s_n)$.

Step3. Initially, set $T_{i,1} = 0$ for $1 \leqq i \leqq n+1$. Then, construct the toggle graph through the following three sub-steps:

Step3-1. $T_{1,0} = 0$, $T_{2,0} = t_1$ and $T_{i,0} = (T_{i-1,0} + T_{i-2,1} + T_{i-1,1} + t_{i-1}) \bmod 2$ for $3 \leqq i \leqq n+1$, but in this step we just need to calculate $T_{i,0} = (T_{i-1,0} + t_{i-1}) \bmod 2$ for $3 \leqq i \leqq n+1$.

Step3-2. If the number of 1's in the sequence $T_{i,0}$, where $1 \leqq i \leqq n+1$, is greater than $n/2$, then flip each bit $T_{i,0}$.

Step3-3. For i = 1 to n { if $T_{i,0} = 1$ and $T_{i+1,0} = 1$, then set $T_{i,1} = 1$ and $T_{i+1,0} = 0$}//I.e. The value of pixel P_{i+1} does not need to be modified.

Step4. Let $m_{n+1} = (P_{1,1} + P_{2,1} + \cdots + P_{n+1,1})\bmod 2$ If ($s_{n+1} = m_{n+1}$ and $(T_{1,1} + T_{2,1} + \cdots + T_{n,1})\bmod 2 = 1$) or ($s_{n+1} \neq m_{n+1}$ and $(T_{1,1} + T_{2,1} + \cdots + T_{n,1})\bmod 2 = 0$) {
if $T_{n+1,0} = 1$ then just set $T_{n+1,1} = 1$ // I.e. pixel P_{n+1} is increased or decreased by 1.
else if there exists at least one $T_{i,1} = 1$ for $1 \leqq i \leqq n$, then randomly choose one and restore $T_{i,1} = 0$ and $T_{i+1,0} = 1$ //I.e. more one pixel needs to be modified.
else set $T_{n+1,1} = 1$ //I.e. pixel P_{n+1} needs to be increased or decreased by 2.}

Step5. The stego graph can be obtained after performing the following loop operation.
For i = 1 to n+1 { if $T_{i,0} = 1$ and $T_{i,1} = 1$, then increase or decrease the value of pixel P_i by 1; else just flip $P_{i,0}$ }// I.e. the first and second least significant bits of P_i will be flipped simultaneously just by increasing or decreasing the value of P_i by 1.

Extraction Process:

According to the *stego graph,* the *secret string* $(s_1, s_2, \ldots, s_n)$ with $s_1 = (P'_{1,0} + P'_{2,0} + P'_{1,1})\bmod 2$ and $s_i = (P'_{i,0} + P'_{i+1,0} + P'_{i-1,1} + P'_{i,1})\bmod 2$ for $2 \leqq i \leqq n$ and $s_5 = (P'_{1,1} + P'_{2,1} + \ldots + P'_{n,1} + P'_{n+1,1})\bmod 2$ can be extracted directly.

//Note that in algorithm 2, where $P_{i,1}$ and $P'_{i,1}$ respectively denote the second least significant bits (SLSBs) of original pixel P_i and stego pixel P'_i.

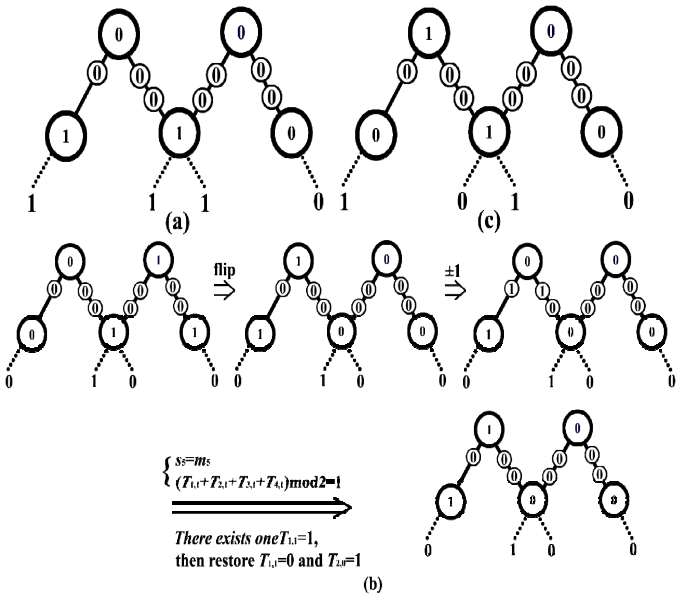

Fig. 4 (a) Master string = (1, 1, 1, 0) (b) The construction of a toggle graph with the least embedding distortion for toggle string = (0, 1, 0, 0) (c) Stego graph is obtained by flipping the LSBs of pixel P_1 and P_2. The first four secret bits = (1, 0, 1, 0).

Example 2: Suppose a block of five pixels $\{P_1, P_2, P_3, P_4, P_5\}$ with respective pixel values = {237, 232, 233, 236, 234} and the secret binary string $\{s_1, s_2, s_3, s_4, s_5\}$ = {1, 0, 1, 0, 1}. Hence, we have the LSB sequence $\{P_{1,0}, P_{2,0}, P_{3,0}, P_{4,0}, P_{5,0}\}$ = {1, 0, 1, 0, 0} and the SLSB sequence $\{P_{1,1}, P_{2,1}, P_{3,1}, P_{4,1}, P_{5,1}\}$ = {0, 0, 0, 0, 1}. The embedding and extraction processes are illustrated in Fig. 4. Note: s_5 = (0+0+0+0+1) = 1.

III. ANALYSIS AND COMPARSION

A. Time Complexity

Theorem III.1: Both of the time complexities respectively for algorithm 1 and algorithm 2 are linear.

Proof: (Embedding) Step 1, step 2, step 3-1, and step 4 respectively need n, n, n+1, and n+1operations. Step 3-2 needs at most $2n$+2 operations. Hence the total number of required operations is equal to $6n$+4. (Extraction) it just needs n operations. Similar proof is omitted here for algorithm 2. □

Table I. Performance Comparison between GBPC and MPC [12]

Algorithm			R	GBPC		
				Distortion	PSNR	Efficiency
GBPC	n= 2		0.6667	0.2500	54.1514	2.6667
	n= 4		0.8000	0.3125	53.1823	2.5600
	n= 6		0.8571	0.3438	52.7684	2.4935
	n= 8		0.8889	0.3633	52.5284	2.4468
	n= 10		0.9091	0.3770	52.3679	2.4117
	n= 12		0.9231	0.3872	52.2514	2.3839
	n= 14		0.9333	0.3953	52.1619	2.3613
MPC	h=3	N=3	0.6750	0.2563	54.0442	2.6341
	h=3	N=4	0.7529	0.2888	53.5251	2.6073
	h=3	N=5	0.8013	0.3137	53.1656	2.5543
	h=3	N=6	0.8340	0.3309	52.9344	2.5207
	h=3	N=7	0.8575	0.3441	52.7634	2.4917
	h=3	N=8	0.8752	0.3550	52.6279	2.4651
	h=3	N=9	0.8890	0.3634	52.5264	2.4461
	h=3	N=10	0.9001	0.3712	52.4351	2.4250
	h=3	N=11	0.9092	0.3786	52.3490	2.4014
	h=3	N=12	0.9167	0.3829	52.3002	2.3942
	h=3	N=13	0.9231	0.3873	52.2508	2.3837
	h=3	N=14	0.9286	0.3919	52.1992	2.3696
	h=3	N=15	0.9334	0.3954	52.1602	2.3604

For the time complexity of MPC algorithm with h = 3. Since there are $(N^{h+1}-1)/(N-1)$ nodes in the toggle tree, so step 2 needs $(N^{h+1}-1)/(N-1)$ operations. Since the toggle tree consists of $(1+N+N^2)$ subtrees, so step 3 needs at most $(2N+2)(1+N+N^2)$ operations.

Therefore, for the GBPC and MPC algorithms, the time-complexities spent in constructing the toggle sequence are respectively equal to $(n+1)+(2n+2)$ and $(N^{h+1}-1)/(N-1)+(2N+2)(1+N+N^2)$. For n =12 and (h=3, N = 13, the number of secret bits is equal to 2197), the complexities are respectively equal to equal to 39 and 7504 (both algorithms have the same embedding rate R = 0.9231). Since 2197/12 ≈ 183.08 and 183.08×39 ≈ 7140 < 7504. Hence, the complexity of GBPC is lower than that of MPC. Notice that the complexity of MPC will become much higher when N is even.

B. PSNR and Efficiency

The embedding efficiencies of algorithm 1 listed in Table I can be calculated by Theorem II.2. Although the embedding efficiency of algorithm 2 is not presented in a closed-form formula, it still can be easily obtained from algorithm 2, as shown in in Table II, by feeding the algorithm with 2^n different toggle strings.

The theoretical analysis shows that the embedding efficiencies are higher than those of MPC and MTC.

Table II. Performance Comparison between GBPC and MTC [13]

Algorithm		R	GBPC		
			Distortion	PSNR	Efficiency
GBPC	n= 2	1.0000	0.5000	51.1411	2.0000
	n= 4	1.0000	0.3563	52.6133	2.8070
	n= 6	1.0000	0.3304	52.9410	3.0270
	n= 8	1.0000	0.3223	53.0487	3.1030
	n= 10	1.0000	0.3189	53.0938	3.1354
	n= 12	1.0000	0.3172	53.1172	3.1524
	n= 14	1.0000	0.3163	53.1304	3.1620
	n= 16	1.0000	0.3157	53.1380	3.1675
	n= 18	1.0000	0.3154	53.1422	3.1706
	n= 20	1.0000	0.3152	53.1443	3.1721
	n= 22	1.0000	0.3152	53.1450	3.1726
MTC					
Complete binary tree coding	h=2	1.0570	0.3701	52.4476	2.8560
Complete 2-3 hybrid tree coding	h=2	1.1887	0.4075	52.0294	2.9170
	h=3	1.0566	0.3472	52.7251	3.0433
	h=4	1.0010	0.3263	52.9946	3.0677
Complete ternary tree coding	h=2	1.1887	0.4064	52.0413	2.9250
	h=3	1.1000	0.3558	52.6183	3.0913
	h=4	1.0700	0.3416	52.7952	3.1320

IV. CONCLUSIONS

It's easy to check that Algorithm 1 is equivalent to an (n+1, n, n/2) binary linear stego-code. Furthermore, each of the proposed two algorithms can be easily implemented with a combinational logic circuit fo real- time applications.

REFERENCES

[1] W. Bender, W. Butera, D. Gruhl, R. Hwang, F. J. Paiz, and S. Pogreb, "Applications for data hiding," IBM Systems Journal, vol. 39, nos.3&4, pp. 547-568, 2000.

[2] R. Crandall, "Some notes on steganograph," *http://of.inf.tu-duresden.de/~westfeld/crandall.pdf*, 1998.

[3] M. Khatirinejad and P. Lisonek, "Linear codes for high payload steganography," *Discrete Applied Mathematics,* vol. 157, pp.971-981, Mar. 2009.

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

[5] Weiming Zhang, Shuozhong Wang, and Xinpeng Zhang, "Improving Embedding Efficiency of Covering Codes for Applications in Steganography," IEEE Communications Letters, vol. 11, no. 8, pp. 680 – 682, Aug. 2007.

[6] W. Zhang, X. Zhang, and S. Wang, "Maximizing steganographic embedding efficiency by combining hamming codes and wet paper codes," in *Proc. Int. Workshop Inf. Hiding (IH 08)*, 2008, vol. LNCS 5284, pp. 60–71.

[7] W. Zhang and S. Li, "A coding problem in steganography," *Designs, Codes Cryptogr.*, vol. 46, no. 1, pp. 68–81, 2008.

[8] X. Zhang and S. Wang, "Efficient steganographic embedding by exploiting modification direction," *IEEE Communications Letters*, vol. 10, no. 11, pp.781-783, Nov. 2006.

[9] K. N. Chen, C. C. Chang, and H. C. Lin, "A large payload EMD embedding scheme with high stego-image quality," In *Proc. Int. Conf. Computational Aspects of Social Networks*, 2010, pp.126-130.

[10] Y. M. Huang and P. W. Jhan, "Two Improved Data Hiding Schemes," in *Proc. of the 4th IEEE International Congress on Image and Signal Processing*, Shanghai, China, Oct. 15-17, 2011, pp. 1784-1787.

[11] R. Y. M. Li, O. C. Au, K. K. Lai, C. K. Yuk, and S.-Y. Lam, "Data hiding with tree based parity check," in *Proc. IEEE Int. Conf. Multimedia and Expo*, Beijing, China, Jul. 2-3, 2007, pp. 635–638.

[12] Chung-Li Hou, Chang-Chun Lu, Shi-Chun Tsai and Wen-Guey Tzeng,"An optimal data hiding scheme with tree-based parity check," *IEEE Trans. Image Processing*, vol. 20, no. 3, pp. 880–886, Mar. 2011.

[13] Fengyong Li, Liu Shi, Xinpeng Zhang, Yuejun Chen, "Data Embedding with Multi-ary Tree Coding," in *Proc. IEEE Int. Conf. Multimedia Technology,* Hangzhou, China, Jul. 26-28, 2011, pp. 267-270