IOSR Journal of Engineering
e-ISSN: 2250-3021, **p**-ISSN: 2278-8719,
Vol. 2, Issue 12 (Dec. 2012), ||V3|| PP 05-11

Tree-Based Parity Check for an Optimal Data Hiding Scheme

M Kiran Kumar, P Sreedhar, L N V Rao
Associate Professor of CSE, CEC, Chirala

***Abstract*—Reducing distortion between the cover object and the stego object is an important issue for steganography. The tree-based parity check method is very efficient for hiding a message on image data due to its simplicity. Based on this approach, we propose a majority vote strategy that results in least distortion for finding a stego object. The lower embedding efficiency of our method is better than that of previous works when the hidden message length is relatively large.**

***Index Terms*—Image coding, image processing, information security.**

I. INTRODUCTION

Stenography studies the scheme to hide secrets into the communication between the sender and the receiver such that no other people can detect the existence of the secrets. A steganographic method consists of an embedding algorithm and an extraction algorithm. The embedding algorithm describes how to hide a message into the cover object and the extraction algorithm illustrates how to extract the message from the stego object. A commonly used strategy for steganography is to embed the message by slightly distorting the cover object into the target stego object. If the distortion is sufficiently small, the stego object will be indistinguishable from the noisy cover object. Therefore, reducing distortion is a crucial issue for stegano graphic methods. In this paper, we propose an efficient embedding scheme that uses the least number of changes over the tree-based parity check model.

The TBPC method can be formulated as a matrix embedding method, but is more efficient than those based on linear codes. Due to its simplicity, the TBPC method provides very efficient embedding and extraction algorithms. Recently, Zhang *et al.* [14] proposed a systematic method to generate codes with an arbitrary small relative payload from any code with a large relative payload. Since our method works naturally with large relative payloads, the result of Zhang *et al.* [14] implies that our method applies to small relative payloads as well.

We observe that the toggle criteria of a node in the TBPC method can be relaxed by the strategy of majority vote. Our strategy inherits the efficiency of the TBPC method and produces a stego object with least distortion under the tree based parity check model. The time complexity of our embedding (extraction as well) algorithm is asymptotically optimal, that is, it is linearly bounded by the hidden message length. The *embedding efficiency* is defined to be the number of hidden message bits per embedding modification. Higher embedding efficiency implies better undetectability for steganographic methods. The *lower embedding efficiency* is defined to be the ratio of the number of hidden message bits to the maximum embedding modifications. The lower embedding efficiency is related to undetectability in the worst case. It implies steganographic security in the worst case. Thus, the lower embedding efficiency is also an important security factor for a steganographic system.

II. PRELIMINARY AND TBPC METHOD

The TBPC method is a least significant bit (LSB) steganographic method. Only the LSBs of the elements pointed by the determined locations are used for embedding and extraction. The TBPC method constructs a complete _-ary tree, called the *master tree*, to represent the LSBs of the cover object. Then it fills the nodes of the master tree with the LSBs of the cover object level by level, from top to bottom and left to right. Every node of the tree corresponds to an LSB in the cover object. Denote the number of leaves of the master tree by _. The TBPC embedding algorithm derives an _-bit binary string, called the *master string*, by performing parity check on the master tree from the root to the leaves (e.g., see Fig. 1.).

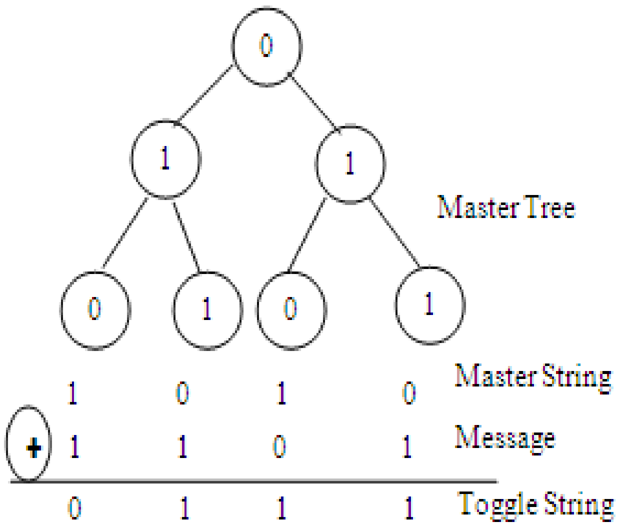
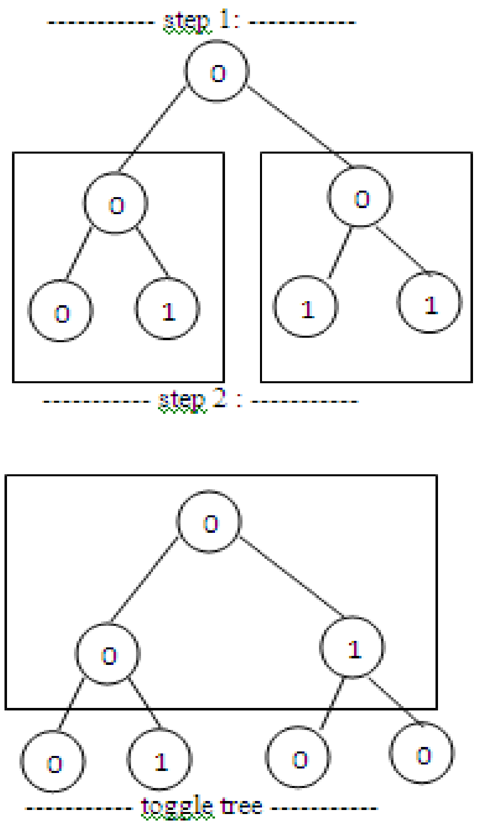


Fig. 1. Master and toggle strings of a master tree

The embedding algorithm hides the message by modifying the bit values of some nodes in the master tree. Assume that the length of the message is also L. Performing the bitwise exclusive-or (XOR) operation between the message and the master string, we obtain a *toggle string*

Then, the embedding algorithm constructs a new complete N-ary tree, called the *toggle tree* in the bottom-up order and fills the leaves with the bit values of the toggle string and the other nodes with 0. Then, level by level, from the bottom to the root, each nonleaf node together with its child nodes are flipped if all its child nodes have bits 1 (e.g., see Fig. 2). The embedding algorithm obtains the *stego tree* by performing XOR between the master tree and the toggle tree (e.g., see Fig. 3). The TBPC extraction algorithm is simple. We can extract the message by performing parity check on each root-leaf path of the stego tree from left to right.


----------- step 1: -----------
0
0
0
0
1
1
1
----------- step 2 : -----------
0
0
1
0
1
0
0
----------- toggle tree -----------

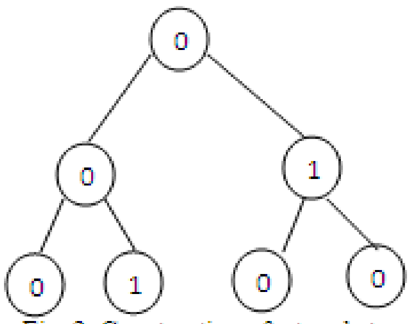


Fig. 2. Construction of a toggle tree

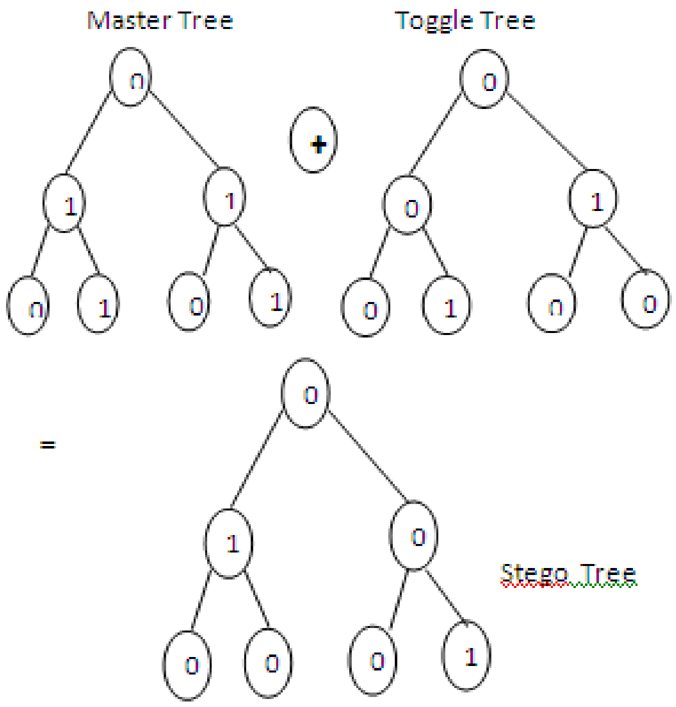


Fig. 3. Modify the master tree into the stego tree by the toggle tree construction

III. MAJORITY VOTE STRATEGY

Two critical issues for a steganographic method are: 1) reducing distortion on cover objects and 2) better efficiency for embedding and extraction. We give a majority vote strategy on building the toggle tree. It uses the least number of 1's under the TBPC model. Since the number of 1's in the toggle tree is the number of modifications on the master tree (i.e., the cover object), the majority vote strategy can produce a stego tree with least distortion on the master tree.

A. Algorithm

Hereafter, we use majority-vote parity check (MPC) to denote our method due to its use of majority vote in deriving the parity check bit. We construct the toggle tree with the minimum number of 1's level by level in the bottom-up order using the following algorithm.

Algorithm MPC

Input: a toggle string of length L;

1. Index the nodes of the initial toggle tree;

2. Set the leaves of the toggle tree from left to right and bit by bit with the toggle string and the other nodes 0;

```
3.  for i = 1 to h
        for each internal node on level i do
            if the majority of its unmarked child nodes holds 1
            then flip the bit values of this node and its child nodes;
            else if the numbers of 0 and 1 in its unmarked child nodes are the same
            then mark this internal node;

4.  if N is even then
        for i = h-1 for 1
            for each marked internal node holding 1 on level i do
                flip the bit values of this node and its child nodes;
```

First, index all nodes of a complete N-ary tree with L leaves from top to bottom and left to right. Set the L-bit toggle string bit by bit into the L leaves from left to right and the other nodes 0. Assume that the level of the tree is h. Traverse all nonleaf nodes from level 1 to h. A nonleaf node and its child nodes form a simple complete subtree. For each simple complete subtree, if the majority of the child nodes hold 1, then flip the bit values of all nodes in this subtree. Since the construction is bottom-up, the bit values of the child nodes in every simple complete subtree are set after step 3. Note that marking a node at step 4 applies only for N being even. When N is even, after step 3, there may exist a two-level simple complete subtree with N/2 1's in the child nodes and 1 in its root. In this case, flipping the bit values in this simple complete subtree results in one fewer node holding 1 and keeps the result of related root-leaf path parity check unchanged. Step 4 takes care of this when the condition applies, and it is done level by level from top to bottom. Also note that for the root of the whole toggle tree, the bit value is always 0 when half of its child nodes hold 1. Thus, after step 4, the bit values of the child nodes in each simple complete subtree are determined.

The number of 1's in the toggle tree is the number of modifications. When constructing the toggle tree, the original TBPC method flips a simple complete subtree only if all of child nodes have 1. We prove that the majority vote strategy actually obtains toggle trees with the least number of 1's.

We call a toggle tree with the least number of 1's corresponding to a toggle string an *optimal toggle tree*. We say that a toggle tree is in *majority form* if for each internal node at least half of its child nodes have bit value 0 and the internal node holds 0 when exactly half of its child nodes holding 1. The output of the algorithm is a toggle tree in majority form. The majority vote guarantees that at least half child nodes of an internal node hold 0. Note that every optimal toggle tree can be transformed into majority form. It is obvious whenN is even. When N is odd, we can check each 2-level simple complete subtree level by level in the top-down order and flip the bit values of the root node and its N child nodes if exactly (N+1)/2 of the child nodes hold 1. Note that, when this situation applies, the root node must hold 0 before flipping, otherwise the toggle tree is not optimal. This rearrangement does not introduce an extra 1 and the result of each root-leaf path parity check is not affected.

B. Experimental Results for MPC and TBPC

We implemented our MPC method and the TBPC method for a comparison between their *pToggle* values. We constructed _-ary toggle trees with more than 15000 leaf nodes for N = 2, 3, … 15. For each N, we randomly generated 200 distinct toggle strings. The results are shown in Fig. 4 and Table I. The results show that MPC is always better than TBPC for $N \geq 3$. When N = 2, they are the same.

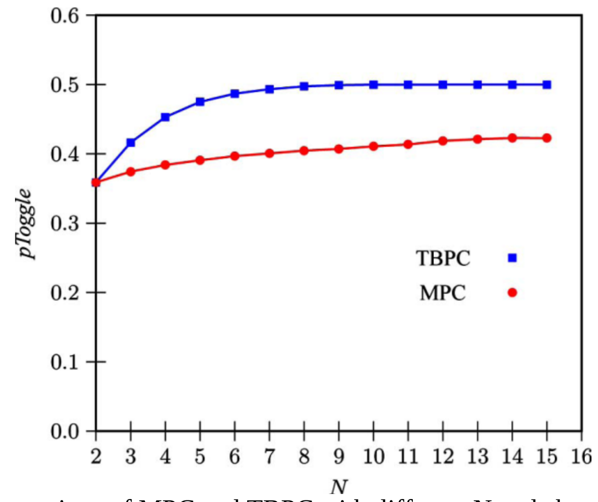


Fig. 4. pToggle comparison of MPC and TBPC with different N and about 15000 leaf nodes.

TABLE I : EXPERIMENTAL RESULTS OF *PTOGGLE*

N	TBPC	MPC		N	TBPC	MPC
2	0.3589	0.3589		9	0.4991	0.4071
3	0.4164	0.3744		10	0.4999	0.4108
4	0.4531	0.384		11	0.4998	0.4136
5	0.475	0.3908		12	0.4999	0.4187
6	0.4869	0.3769		13	0.4999	0.4212
7	0.4933	0.4007		14	0.4999	0.4229
8	0.4974	0.4047		15	0.4999	0.4228

To make it clear, we define the percentage of reduced modifications as follows:

$$pReduce = R_t / D_t$$

where R_t is the reduced number of 1's in the toggle tree and D_t is the number of 1's in the toggle string. The *pReduce* values of both methods are shown in Fig. 5 and Table II. The results show that the MPC method significantly improves previous TBPC results.

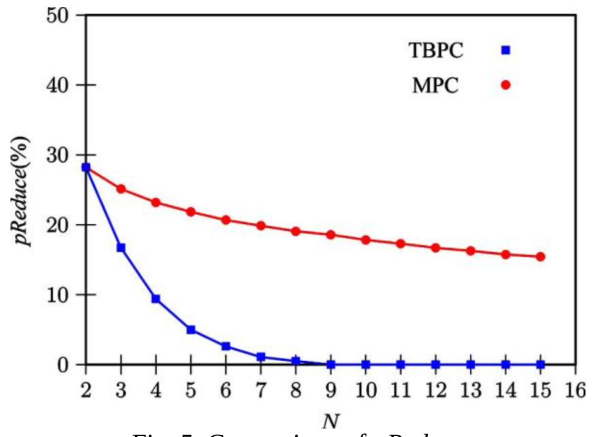


Fig. 5. Comparison of *pReduce*

TABLE II : EXPERIMENTAL RESULTS OF *PREDUCE*

N	MPC	TBPC		N	MPC	TBPC
2	28.2%	28.2%		9	18.5%	0.00%
3	25.1%	16.7%		10	17.8%	0.00%
4	23.1%	9.3%		11	17.2%	0.00%
5	21.8%	4.98%		12	16.6%	0.00%
6	20.6%	2.60%		13	16.2%	0.00%
7	19.8%	1.08%		14	15.7%	0.00%
8	19.0%	0.50%		15	15.4%	0.00%

C. Applications

Our method is based on an N-ary complete tree structure. Fixed the level of the tree, given a larger N we can hide more message bits and the relative payload is larger. Like the previous works proposed by Fridrich and Soukal [8], our method can be applied to the situation that the relative payload is large. On the other hand, since our method is asymptotically optimal, the embedding and extraction algorithms are efficient and can be used on online communications.

IV. CONCLUSION

By introducing the majority vote strategy, we effectively construct the stego object with least distortion under the tree structure model. We also show that our method yields a binary linear stego-code. In comparison with the TBPC method, our method significantly reduces the number of modifications on average.

REFERENCES

1). S. Arora, L. Babai, J. Stern, and Z. Sweedyk, "The hardness of approximate optima in lattices, codes, and systems of linear equations," *J. Comput. Syst. Sci.*, vol. 54, no. 2, pp. 317–331, 1997.

2). G. Cohen, I. Honkala, S. Litsyn, and A. Lobstein, *Covering Codes*. Amsterdam, The Netherlands: North-Holland, 1997.

3). G. Cohen, M. Karpovsky, H. Mattson, and J. Schatz, "Covering radiussurvey and recent results," *IEEE Trans. Inf. Theory*, vol. IT-31, no. 3, pp. 328–343, May 1985.

4). R. Crandall, "Some Notes on Steganography, Posted on Steganography Mailing List," 1998 [Online]. Available: http://os.inf.tu-dresden.de/ westfeld/crandall.pdf

5). J. Fridrich, "Asymptotic behavior of the ZZW embedding construction," *IEEE Trans. Inf. Forensics Security*, vol. 4, no. 1, pp. 151–154, Mar. 2009.

6). J. Fridrich, M. Goljan, P. Lisonek, and D. Soukal, "Writing on wet paper," *IEEE Trans. Signal Process.*, vol. 53, no. 10, pp. 3923–3935, Oct. 2005.

7). J. Fridrich, M. Goljan, and D. Soukal, "Efficient wet paper codes," in *Proc. 7th Int. Workshop Inf. Hiding (IHW 05), Lecture Notes in Computer Science*, 2005, vol. 3727, pp. 204–218.

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

9). M. Khatirinejad and P. Lisonek, "Linear codes for high payload steganography," *Discrete Applied Math.*, vol. 157, no. 5, pp. 971–981, 2009.

10). 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 (ICME 07)*, 2007, pp. 635–638.

AUTHOR BIOGRAPHY

Mr M Kiran Kumar received B.Tech. degree in Information Technology from Acharya Nagarjuna University, Guntur, in 2005, received M.Tech. degree in Computer Science Engineering from JNTU, Hyderabad, in 2010. He has 07 Years of rich Teaching Experience in reputed Engineering Colleges & He is currently working as Associate Professor in Computer Science & Engineering department in Chirala Engineering College, Chirala. He Published 2 Research papers in various Conferences. Guided many UG & PG students for projects. Conducted successfully many Workshops, Seminars, conferences, FDPs and National Level Technical Symposiums.

Mr Sreedhar Pulipati received B.Tech. degree in Computer Science Engineering from JNT University, Kakinada, in 2005, received M.Tech. degree in Computer Science Engineering from JNT University, Kakinada, in 2010. He has 06 Years of rich Teaching Experience in reputed Engineering Colleges & He is currently working as Assistant Professor in Computer Science & Engineering department in Chirala Engineering College, Chirala.

Mr L N V RAO received B.Tech. degree in information Technology from JNT University, Hyderabad, in 2006, received M.Tech. degree in Computer Science Engineering from ANU, Guntur, in 2010. He has 06 Years of rich Teaching Experience in reputed Engineering Colleges & He is currently working as Assistant Professor in Computer Science & Engineering department in Chirala Engineering College, Chirala. He Published 1 Research papers in various Conferences. Guided many UG & PG students for projects.