Minimizing Embedding Impact in Steganography using Trellis-Coded Quantization

Tomáš Filler, Jan Judas, and Jessica Fridrich

Department of Electrical and Computer Engineering
SUNY Binghamton, Binghamton, NY 13902-6000, USA

ABSTRACT

In this paper, we propose a practical approach to minimizing embedding impact in steganography based on syndrome coding and trellis-coded quantization and contrast its performance with bounds derived from appropriate rate-distortion bounds. We assume that each cover element can be assigned a positive scalar expressing the impact of making an embedding change at that element (single-letter distortion). The problem is to embed a given payload with minimal possible average embedding impact. This task, which can be viewed as a generalization of matrix embedding or writing on wet paper, has been approached using heuristic and suboptimal tools in the past. Here, we propose a fast and very versatile solution to this problem that can theoretically achieve performance arbitrarily close to the bound. It is based on syndrome coding using linear convolutional codes with the optimal binary quantizer implemented using the Viterbi algorithm run in the dual domain. The complexity and memory requirements of the embedding algorithm are linear w.r.t. the number of cover elements. For practitioners, we include detailed algorithms for finding good codes and their implementation. Finally, we report extensive experimental results for a large set of relative payloads and for different distortion profiles, including the wet paper channel.

Keywords: Steganography, embedding impact, matrix embedding, wet paper codes, trellis-coded quantization, convolutional codes

1. INTRODUCTION

In steganography,[1] the sender communicates with the receiver by hiding her messages in generally trusted media, such as digital images, so that it is hard to distinguish between the original (cover) object and the objects carrying the message (stego objects). The message is typically hidden (embedded) in the cover image by slightly modifying individual elements of the cover (pixels, DCT coefficients, etc.). We represent cover and stego objects using binary vectors $\mathbf{x} = (x_1, \dots, x_n), \mathbf{y} = (y_1, \dots, y_n) \in \{0,1\}^n$, via some bit-assignment function, such as $\bmod 2$. Furthermore, we assume that the embedding operation is binary, which means that each cover element is either changed by a specified amount or not changed at all. The cost of making an embedding change at pixel x_i is ρ_i, where $0 \le \rho_i \le \infty$, is the set of *single-letter distortions*. In general, ρ_i may depend on the neighborhood of pixel x_i or some other side-information, such as the quantization error during the genesis of the cover. Alternatively, ρ_i may be obtained from theoretical considerations as $\rho_i \propto -\ln p_i$,[2] where p_i is the probability with which x_i should be changed derived from a cover model that minimizes the root rate.[3] As such, ρ_i may be highly non-uniform. Assuming the embedding changes do not interact (e.g., when the number of changes is small), the total impact of embedding is the sum of the embedding impact at every pixel $D(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{n} \rho_i |x_i - y_i|$.

Under the above assumptions, the sender strives to embed her binary message $\mathbf{m} \in \{0,1\}^m$ so that the total distortion D is minimized. For properly defined ρ_i, this will minimize the *statistical impact of embedding* and thus make the resulting stegosystem less detectable (more secure).[*] This problem is not new to steganography and has been approached using variants of syndrome coding. For example, F5[4] uses matrix embedding (which is limited to $\rho_i = 1$) and nsF5[5] employs wet paper codes, where $\rho_i \in \{1, \infty\}$. The MMx algorithm[6] and

E-mail: {tomas.filler, fridrich}@binghamton.edu; snugar.i@gmail.com; T.F.: http://dde.binghamton.edu/filler

[*] We would like to stress that the important problem of how to define ρ_i so that minimizing distortion indeed corresponds to minimizing the statistical detectability is not investigated in this paper and is left as future work.

Media Forensics and Security II, edited by Nasir D. Memon, Jana Dittmann, Adnan M. Alattar,
Edward J. Delp III, Proc. of SPIE-IS&T Electronic Imaging, SPIE Vol. 7541, 754105
© 2010 SPIE-IS&T · CCC code: 0277-786X/10/$18 · doi: 10.1117/12.838002

perturbed quantization[5] use suboptimal methods to minimize a real-valued single-letter distortion proportional to the quantization error.

Despite the fact that in many practical steganographic schemes the distortions ρ_i are usually defined heuristically, minimizing the distortion indeed seems to lead to more secure stegosystems as witnessed by the comparisons from Ref. 5 and Ref. 7. As the recent work of Sachnev et al.[8] suggests, the improvement in security when compared to other schemes can be substantial. This motivates our current work in which we focus on developing general coding techniques for minimizing the embedding impact for arbitrary distortion.

This paper is organized as follows. Since our method is based on syndrome coding, in the next section, we review this general embedding principle and then state the bound between the relative payload and minimal embedding distortion. Section 3 includes a short review of current state-of-the-art steganographic methods that minimize embedding impact. A complete description of the proposed algorithm is given in Section 4. It starts with a description of the method for practitioners (it requires no prior advanced knowledge) and then connections are made between the proposed approach and the coding theory. Implementation details, including the design of good codes for different distortion profiles, such as the wet paper channel, appear in Section 5. Section 6 contains extensive experimental results for a large set of relative payloads. Finally, the paper is concluded in Section 7.

2. SYNDROME CODING IN STEGANOGRAPHY

The problem of minimizing the embedding impact for single-letter distortion as described above was formulated in Ref. 2, where appropriate bounds were derived. We adhere to the notation defined in this paper and define the embedding and extraction mappings as Emb : $\{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$ and Ext : $\{0,1\}^n \to \{0,1\}^m$ satisfying

$$\mathrm{Ext}(\mathrm{Emb}(\mathbf{x},\mathbf{m})) = \mathbf{m} \quad \forall \mathbf{x} \in \{0,1\}^n, \forall \mathbf{m} \in \{0,1\}^m,$$

respectively. In particular, we do not assume the knowledge of the distortion profile ρ_i at the receiver. In practice, we are interested in practical methods that can embed an m-bit message in an n-element cover, while keeping the expected distortion $E[D(\mathbf{x}, \mathrm{Emb}(\mathbf{x},\mathbf{m}))]$ as small as possible.[†] In syndrome coding, the embedding and extraction mappings are realized using a binary linear code $\mathcal{C}$ of length n and dimension $n-m$:

$$\mathrm{Emb}(\mathbf{x},\mathbf{m}) = \arg \min_{\mathbf{y}\in\mathcal{C}(\mathbf{m})} D(\mathbf{x},\mathbf{y}), \tag{1}$$

$$\mathrm{Ext}(\mathbf{y}) = \mathbb{H}\mathbf{y}, \tag{2}$$

where $\mathbb{H} \in \{0,1\}^{m\times n}$ is a parity-check matrix of the code $\mathcal{C}$, $\mathcal{C}(\mathbf{m}) = \{\mathbf{z} \in \{0,1\}^n | \mathbb{H}\mathbf{z} = \mathbf{m}\}$ is the coset corresponding to syndrome $\mathbf{m}$, and all operations are in binary arithmetic. The most challenging part here is an efficient implementation of the optimal binary coset quantizer (1). In this work, we describe a rich class of codes for which the problem (1) can be solved optimally with linear time and space complexity w.r.t. n.

2.1 Performance bounds

The problem of lossy quantization with distortion criterion was first described by Shannon.[9] Let us first assume that the set of single-letter distortions is bounded by a constant, $0 \le \rho_i < C$. Furthermore, we assume that after sorting ρ_i into a non-decreasing sequence and normalizing ($\sum_i \rho_i = 1$), ρ_i can be written as $\rho_i = \rho(i/n)$, where $\rho(\cdot)$ is a Riemann-integrable non-decreasing function on $[0,1]$ called *the profile*. For a fixed relative payload $\alpha = m/n$ (or code rate $R = (n-m)/n = 1-\alpha$), the minimum expected per-element distortion, $d(\alpha) = E[D(\mathbf{x}, \mathrm{Emb}(\mathbf{x},\mathbf{m}))]/n$, can be written in the following parametric form:[2]

$$d(\lambda) = G_\rho(\lambda), \qquad \alpha(\lambda) = \frac{1}{\ln 2}\Big(\lambda F_\rho(\lambda) + \ln\big(1 + e^{-\lambda\rho(1)}\big)\Big),$$

[†] Here we assume that both the cover and the message are random i.i.d. Bernoulli(1/2) sequences.

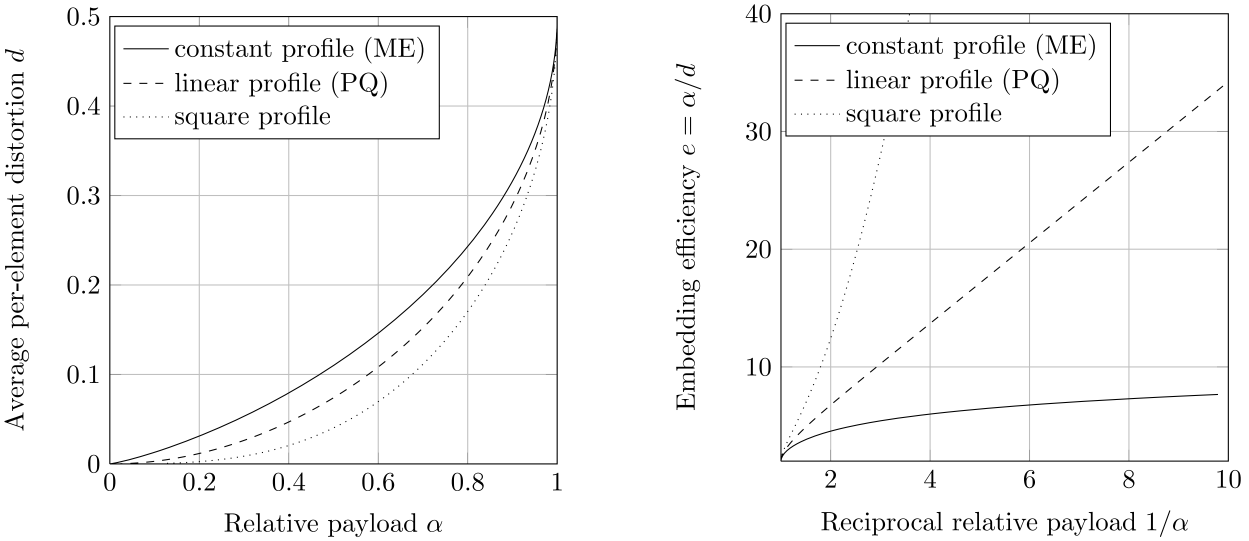

Figure 1. Left: Lower bound on the average per-element distortion $d(\alpha)$ for different distortion profiles. Right: Upper bound on the embedding efficiency e w.r.t. $1/\alpha$ for different embedding profiles (e is the number of bits that can be embedded per unit distortion).

where $\lambda \in [0, \infty)$ and

$$G_\rho(\lambda) = \int_0^1 \frac{\rho(x)e^{-\lambda\rho(x)}}{1 + e^{-\lambda\rho(x)}}\mathrm{d}x, \qquad\qquad F_\rho(\lambda) = \int_0^1 \frac{(\rho(x) + x\rho'(x))e^{-\lambda\rho(x)}}{1 + e^{-\lambda\rho(x)}}\mathrm{d}x.$$

Figure 1 shows the parametric function $(d, \alpha)(\lambda)$ for three distortion profiles described below. We use the *embedding efficiency* $e = \alpha/d$ to compare different syndrome coding algorithms for a given profile.

2.2 Profiles

The following distortion profiles are of interest in steganography (this is not an exclusive list) and will be used for benchmarking the proposed methods: the *constant profile*, $\rho(x) = 1$, when all pixels have the same impact on detectability when changed; the *linear profile*, $\rho(x) = 2x$, when the distortion is related to a quantization error uniformly distributed on $[-Q/2, Q/2]$ for some quantization step $Q > 0$; and the *square profile*, $\rho(x) = 3x^2$, which can be encountered when the distortion is related to a quantization error that is not uniformly distributed.

In practice, some cover elements may be assigned $\rho_i = +\infty$ (the so-called *wet* elements[5, 10, 11]) to force the embedding algorithm to keep such elements unmodified. We exclude these elements from the above analysis (since they are constant), and measure the per-element distortion d and the relative payload α with respect to the set of *dry elements* $\{x_i|\rho_i < \infty\}$. This channel is called the wet paper channel and it is characterized by the profile $\rho(\cdot)$ of dry elements and *relative wetness* $\tau = |\{i|\rho_i = \infty\}|/n$. The code design for the wet paper channel with an arbitrary profile and relative wetness τ is described in Section 5.3.

3. PRIOR ART

The problem of minimizing the embedding impact in steganography, as described above, has been already conceptually described by Crandall[12] in his essay posted on the steganography mailing list in 1998. He suggested that whenever the encoder embeds at most one bit per pixel, it should make use of the embedding impact defined for every pixel and minimize its total sum:

> "Conceptually, the encoder examines an area of the image and weighs each of the options that allow it to embed the desired bits in that area. It scores each option for how conspicuous it is and chooses the option with the best score."

Later, Bierbrauer[13] studied a special case of this problem and described a connection between codes (not necessarily linear) and the problem of minimizing the number of changed pixels (the constant profile). This connection, which has become known as matrix embedding (encoding), was made famous by Westfeld[4] who incorporated it in his F5 algorithm. A binary Hamming code was used to implement the syndrome-coding (1) for the constant profile. Later on, different authors suggested other linear codes, such as Golay,[14] BCH,[15] random codes of small dimension,[16] and non-linear codes based on the idea of blockwise direct sum.[17] The current state-of-the-art method uses codes based on Low Density Generator Matrices (LDGMs)[2] in combination with the ZZW construction.[18] The embedding efficiency of these codes stays rather close to the bound for arbitrarily small relative payloads.[19]

The versatile syndrome coding approach can also be used to communicate via the wet paper channel using the so-called wet paper codes.[11] Wet paper codes with improved embedding efficiency were described in Ref. 15,20–22. Even though other distortion profiles, such as the linear profile, are of great interest to steganography, no general solution with performance close to the bound is currently known. The authors of Ref. 6 minimize the embedding distortion on small blocks using syndrome coding based on Hamming codes and a suboptimal quantizer implemented using a brute-force search that allows up to three embedding changes. Such an approach, however, provides highly suboptimal performance far from the theoretical bound (see Figure 7). A similar approach based on BCH codes and a brute-force quantizer was described in Ref. 8.

4. SYNDROME-TRELLIS CODES

In this section, we give a complete description of the proposed syndrome-coding scheme and the syndrome-trellis codes. To make this material accessible to a wider audience and to practitioners, our description requires no prior knowledge of the coding theory. Connections to the coding theory and other theoretical aspects of the proposed construction appear in Section 4.2.

We show how to create efficient syndrome-coding schemes for an *arbitrary* $\alpha \leq 1/2$. This should not be seen as a limitation, since the relative payload must decrease with increasing size of the cover object in order to maintain the same level of security.[23] Moreover, recent results from steganalysis in both spatial[24] and DCT domains[25] suggest that secure payload for digital image steganography is always far below 1/2. However, if needed, this approach can be extended to payloads $\alpha \geq 1/2$ as well.

4.1 Algorithm description

In a nutshell, we select the parity-check matrix $\mathbb{H}$ in a special form that allows representing every solution of $\mathbb{H}\mathbf{y} = \mathbf{m}$ as a path through a trellis. The optimal $\mathbf{y}$ closest to $\mathbf{x}$ is then found using the Viterbi algorithm. We first describe the algorithm in its general form and later, in Section 4.1.1, we give a simple example. Readers completely unfamiliar with trellis codes may benefit from reading the example first and then returning to this section.

In our construction, the parity-check matrix $\mathbb{H}$ is obtained by placing a small submatrix $\hat{\mathbb{H}}$ of size $h \times w$ as in Figure 2. The submatrices $\hat{\mathbb{H}}$ are placed next to each other and shifted down by one row, which leads to a sparse, banded $\mathbb{H}$. The height h of the submatrix (called the *constraint height*) is a design parameter that affects the algorithm speed and efficiency (typically, $6 \leq h \leq 15$). The width of $\hat{\mathbb{H}}$ is dictated by the desired relative payload α: if α is equal to $1/k$ for some $k \in \mathbb{N}$, select $w = k$. For general payloads α, find k such that $1/(k+1) < \alpha < 1/k$. The matrix $\mathbb{H}$ will contain a mix of submatrices of width k and $k+1$ so that the final matrix $\mathbb{H}$ is of size $\lceil \alpha n \rceil \times n$. In this way, we can create a parity-check matrix for an arbitrary rational $\alpha \leq 1/2$. The submatrix $\hat{\mathbb{H}}$ acts as an input parameter shared between the sender and the receiver and its choice is discussed in more detail in Section 5.2. For the sake of simplicity, in the following description we assume $\alpha = 1/w$ and thus the matrix $\mathbb{H}$ is of the size $b \times (b \cdot w)$, where b is the number of copies of $\hat{\mathbb{H}}$ in $\mathbb{H}$. The generalization to other sizes is discussed in Section 5.1.

Following the syndrome coding approach described in Section 2, our goal is to find a vector $\mathbf{y}$ such that $\mathbb{H}\mathbf{y} = \mathbf{m}$, where $\mathbf{m}$ is the message we want to communicate, and $\sum \rho_i |x_i - y_i|$ is minimal. Due to the form of the matrix $\mathbb{H}$, only the first w bits of the vector $\mathbf{y}$ can affect the first bit of the message $\mathbf{m}$ (the remaining elements in the first row of $\mathbb{H}$ are zeros). Therefore, we need to select $(y_1, \ldots, y_w)$ such that $(\mathbb{H}_{11}, \ldots, \mathbb{H}_{1w}) \cdot (y_1, \ldots, y_w)^T = m_1$.

$$\hat{\mathbb{H}} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \quad \mathbb{H} = \begin{pmatrix} 1 & 0 & & & & & & & \\ 1 & 1 & 1 & 0 & & & & & \\ & & 1 & 1 & 1 & 0 & & & \\ & & & & 1 & 1 & & & \\ & & & & & \ddots & 1 & 0 & \\ & & & & & & 1 & 1 & 1 \; 0 \end{pmatrix}$$

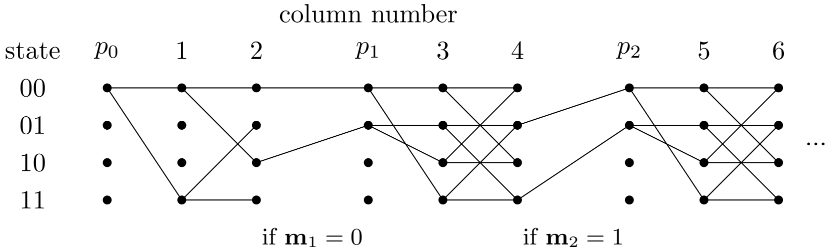

Figure 2. Example of the parity-check matrix $\mathbb{H}$ formed from the submatrix $\hat{\mathbb{H}}$ ($h = 2, w = 2$) and its corresponding syndrome trellis. The last $h - 1$ submatrices in $\mathbb{H}$ are cropped to achieve the desired relative payload α. The syndrome trellis consists of repeating blocks of $w + 1$ columns, where "p_0" and "p_i", $i > 0$, denote the starting and pruning columns, respectively. The column labeled $l \in \{1, 2, \ldots\}$ corresponds to the lth column in the parity-check matrix $\mathbb{H}$.

Similarly, the second bit of $\mathbf{m}$ is only affected by the first $2w$ bits of $\mathbf{y}$, etc. Moreover, any given bit y_i of the vector $\mathbf{y}$ cannot affect more than h message bits because no column in $\mathbb{H}$ contains more than h nonzero elements. This key observation will allow us to find the optimal solution of (1) with a low complexity.

The encoding process can be best explained using a *syndrome trellis* of the parity-check matrix shown in Figure 2. The syndrome trellis (simply a trellis) is a graph consisting of b blocks, each containing $2^h(w + 1)$ nodes organized in a grid of $w + 1$ columns and 2^h rows. The nodes between two adjacent columns form a bipartite graph, i.e., all edges only connect nodes from two adjacent columns. Each block of the trellis represents the submatrix $\hat{\mathbb{H}}$ used to obtain the parity-check matrix $\mathbb{H}$. The nodes in every column are called *states* and are labeled by a number written in a binary notation starting from the top with 0 and ending with $2^h - 1$. The columns in the trellis are labeled as seen in Figure 2. All columns, except for the first column in every block, will be labeled sequentially with an integer $l \in \{1, 2, \ldots, n\}$. The connectivity of the graph, along with the main motivation for the trellis, is explained next.

Each $\mathbf{y}$ satisfying $\mathbb{H}\mathbf{y} = \mathbf{m}$ is represented as a path through the trellis. Each path starts in the leftmost all-zero state in the trellis and extends to the right. The path shows the step-by-step calculation of the (partial) syndrome using more and more bits of $\mathbf{y}$. For example, the first two edges in Figure 2, that connect the state 00 from column p_0 with states 11 and 00 in the next column, correspond to adding ($y_1 = 1$) or not adding ($y_1 = 0$) the first column of $\mathbb{H}$ to the syndrome, respectively.[‡] In general, we call the state in the trellis *reachable*, if there is a path connecting this state with the leftmost all-zero state. From each reachable state, two edges extend to the right; they correspond to adding or not adding the next column of matrix $\mathbb{H}$ to the current partial syndrome and are labeled 1 and 0, respectively. The edge labels along the path in the trellis thus correspond to individual bits of the stego object $\mathbf{y}$.

The states of nodes in columns labeled l, $l \in \{1, \ldots, n\}$, stand for the partial syndromes that can be obtained using the first l bits of the vector $\mathbf{y}$. Fortunately, it is not necessary to enumerate all 2^l possible partial syndromes in each column. Instead, because each column of $\mathbb{H}$ has at most h nonzero values, we need to focus only on those h bits of the syndrome that can actually be changed by adding or not adding the lth column of $\mathbb{H}$ to the partial syndrome. Thus, each column in the trellis has only 2^h nodes. For example, the first w columns of the trellis correspond to the bits $1, \ldots, h$ of the syndrome. The next w columns can affect the $h + 1$st bit of the syndrome as well but not the first bit because all such columns have 0 as their first element. Because we know the value of the first bit of the syndrome, m_1, we do not need to represent this bit in the trellis anymore. Based on the known value of m_1, we prune some of the states to "make room" for the new ones.

At the end of the first block, we terminate all paths (and their end-states) for which the first bit of the partial syndrome does not match m_1. Consequently, the number of states will decrease from 2^h to 2^{h-1}. Next, 2^{h-1} new states are added corresponding to the previously unreachable syndromes with the $h + 1$st bit of the partial syndrome set to one. This way, we obtain a new column of the trellis, which will serve as the starting, "zeroth" column of the next block. This column does not have a number, because it does not correspond to adding/not adding a column of $\mathbb{H}$ to the syndrome (see Figure 2). It merely illustrates the "state transition" of the trellis,

[‡] The state corresponds to the partial syndrome.

```
// forward part of the Viterbi algorithm
wght[0] = 0
wght[1,...,2^h-1] = infinity
indx = indm = 1
for i = 1,...,num of blocks (submatrices in H) {
  for j = 1,...,w {           // for each column
    for k = 0,...,2^h-1 {     // for each state
      w0 = wght[k] + x[indx]*rho[indx]
      w1 = wght[k XOR H_hat[j]] + (1-x[indx])*rho[indx]
      path[indx][k] = w1 < w0 ? 1 : 0     // C notation
      newwght[k] = min(w0, w1)
    }
    indx++
    wght = newwght
  }
  // prune states
  for j = 0,...,2^(h-1)-1
    wght[j] = wght[2*j + message[indm]]
  wght[2^(h-1),...,2^h-1] = infinity
  indm++
}
```

```
// backward part of the Viterbi algorithm
embedding_cost = wght[0]
state = 0, indx--, indm--
for i = num of blocks,...,1 (step -1) {
  for j = w,...,1 (step -1) {
    y[indx] = path[indx][state]
    state = state XOR (y[indx]*H_hat[j])
    indx--
  }
  state = 2*state + message[indm]
  indm--
}
////////////  LEGEND  ////////////////////
// INPUT: x, message, H_hat
// x = (x[1],...,x[n]) cover object
// message = (message[1],...,message[m])
// H_hat[j] = j th column in int notation
//
// OUTPUT: y, embedding_cost
// y = (y[1],...,y[n]) stego object
```

Figure 3. Pseudocode of the Viterbi algorithm as described in Section 4.1.

meaning that its states shift from representing the partial syndrome $(s_1, \ldots, s_h)$ to $(s_2, \ldots, s_{h+1})$.[§] This state pruning is repeated at each block transition in the matrix $\mathbb{H}$. This also implies that the number of vertices in each column is 2^h throughout the whole trellis.

In general, the trellis may consist of blocks of different widths corresponding to possibly different submatrices $\hat{\mathbb{H}}$ used to form $\mathbb{H}$. Every two adjacent blocks are connected by edges which describe the elimination of the ith syndrome bit from the state space and the inclusion of the $i+h$th. These edges do not have any label, because they are not directly connected to any bit of the stego vector $\mathbf{y}$. The structure of these connecting edges is determined by m_i.

It should be clear to the reader that every stego object $\mathbf{y}$ satisfying $\mathbb{H}\mathbf{y} = \mathbf{m}$ can be represented as a path through the whole trellis. To find the closest stego object, we assign weights to all trellis edges and thus transform the problem (1) to the problem of finding the shortest path through the trellis. The weights of the edges entering the column with label l, $l \in \{1, \ldots, n\}$, in the trellis depend on the lth bit of the original cover object $\mathbf{x}$. If $x_l = 0$, then the horizontal edges (corresponding to not adding the lth column of $\mathbb{H}$) have a weight of 0 and the edges corresponding to adding the lth column of $\mathbb{H}$ have a weight of ρ_l (our distortion measure), because to add this column, we would have to change x_l from 0 to 1. If $x_l = 1$, the roles of the edges are reversed. All edges connecting the individual blocks of the trellis have zero weight.

Even though the number of paths through the trellis is exponential in n, the problem of finding the shortest path can be efficiently solved by a form of dynamic programming called *the Viterbi algorithm*. This algorithm consists of two parts, the *forward* and the *backward* part. The forward part of the algorithm consists of $n+b$ steps. Upon finishing the ith step, we know the shortest path between the leftmost all-zero state and every state in the ith column of the trellis (there are 2^h states in every column). Thus in the final, $n+b$th step, we discover the shortest path through the whole trellis. If some state is not reachable (there is no path going to this state), we assign the weight of ∞ to such a path. During the backward part, the shortest path is traced back and the closest stego object $\mathbf{y}$ is recovered from the edge labels.

We now explain in more detail the forward part of the Viterbi algorithm and how the trellis is represented in the computer. Let us assume that we know the shortest path (and its weight) to every state in the column with label $l-1$ in the trellis and we wish to update the data for the column with label l, $l \in \{1, \ldots, n\}$. If we are in the column labeled with l in a state with two incoming edges, it follows from the construction of the trellis

[§] Note that, with the exception of $s_1 = m_1$, the partial syndrome will not, in general, match the message bits.

$$\hat{\mathbb{H}} = \begin{pmatrix} 1\,0 \\ 1\,1 \end{pmatrix} \quad \mathbb{H} = \begin{pmatrix} 1 & 0 & & & & & & \\ 1 & 1 & 1 & 0 & & & & \\ & & 1 & 1 & 1 & 0 & & \\ & & & & 1 & 1 & 1 & 0 \end{pmatrix}$$

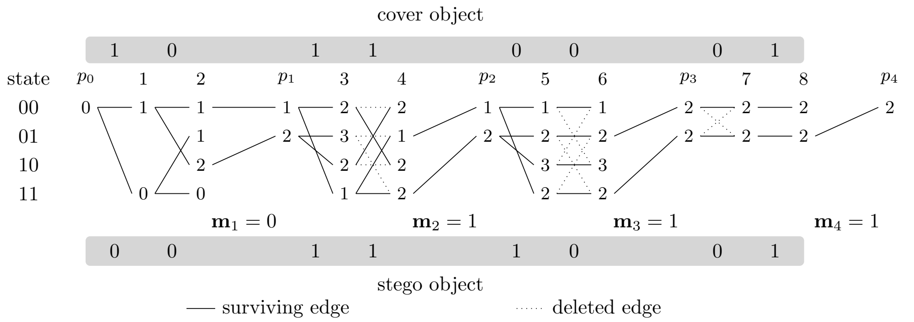

Figure 4. Example of the syndrome trellis processed by the Viterbi algorithm with the cover object $\mathbf{x} = (1, 0, 1, 1, 0, 0, 0, 1)$ and the message $\mathbf{m} = (0, 1, 1, 1)$. The stego object with the minimal number of changed elements is obtained from the backward part of the algorithm by following the path from the end of the trellis, $\mathbf{y} = (0, 0, 1, 1, 1, 0, 0, 1)$.

that the one of the edges is labeled 1 and the other is labeled 0. This means that the syndrome represented by this state can be obtained using two different vectors $(y^1_1, \dots y^1_l)$, $(y^2_1, \dots, y^2_l)$, where $y^1_l = 1$ and $y^2_l = 0$. These two vectors belong to two different paths entering the state. Let the first path have a total weight of w_1 and the second of w_2. We remove the edge that corresponds to the path with the bigger weight from the trellis (ties can be resolved arbitrarily). This leaves us with only one edge coming into the state and, by induction, with a unique path with minimal weight going from the beginning of the trellis to this state.

To store the processed trellis, we represent each state of the trellis using one bit. This bit is set to the label of the incoming edge (there is at most one incoming edge as explained above) and it is undefined if there is no incoming edge. (This should not cause any problem, since there are no outgoing edges either and we can never reach that state in the backward step.)

The above Viterbi algorithm is described in Figure 3 using a pseudocode. An *integer notation* is used to represent the submatrix $\hat{\mathbb{H}}$ – the columns are interpreted as binary numbers, with the first row as the least significant bit. For example, the matrix $\hat{\mathbb{H}} = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$ would be described as $(3, 2, 1)$. This notation is more compact and easier to work with on the implementation level. As we can see from the pseudocode, the algorithm is linear in the number of columns in the trellis (which is equal to the cover size) and exponential in h, the height of the submatrix $\hat{\mathbb{H}}$, because we need to go through 2^h states of the trellis column in each step.

4.1.1 Example of the Viterbi algorithm

In this section, the embedding algorithm is explained on a small example. Let $\mathbb{H}$ be as in Figure 4, the cover vector $\mathbf{x} = (1, 0, 1, 1, 0, 0, 0, 1)$, and the message $\mathbf{m} = (0, 1, 1, 1)$. We will be minimizing the number of embedding changes, which corresponds to a constant profile, $\rho_i = 1$ for all i. The algorithm starts with the leftmost column, denoted by p_0 in Figure 4, in the only reachable state – the all-zero syndrome. Then, two edges are drawn from this state. The first edge connects the state $p_0 s_{00}$ (column p_0, state 00) with the state $c_1 s_{00}$ (column 1, state 00). This edge is labeled "0" (which corresponds to not adding the first column of $\mathbb{H}$ to the syndrome) and is assigned a weight of 1, because not adding the first column means $y_1 = 0$, which implies an embedding change $x_1 = 1 \rightarrow y_1 = 0$. The second edge starting in $p_0 s_{00}$ and ending in $c_1 s_{11}$ (adding the first column of $\mathbb{H}$ to the syndrome) is assigned the label "1" and a weight of 0 (we do not have to change x_1). Thus, the first column has two reachable states – 00 with the shortest path of weight 1 and 11 with a shortest path of weight 0. Similarly, in the second column, all four states are reachable, 00 with a shortest path of weight 1, 01 with weight 1, 10 with 2, and 11 with 0.

After processing the first two columns of $\mathbb{H}$, the first bit of $\mathbf{m}$ must already be set to the desired value (in this case, $m_1 = 0$). This means that the states 01 and 11 can be discarded, as their least significant bit is 1 and will not change anymore. Therefore, no edges come out of these states. Next, the trellis state shifts from representing bits (s_1, s_2) to bits (s_2, s_3). This is illustrated by the trellis column labeled p_1 and edges that connect it to the previous column. These edges have a weight of 0 and no label and represent the "prune states" section in the pseudocode.

The column with label 3 is constructed identically to columns 1 and 2. In column 4, we observe that there are two edges coming into each state. Let us take a closer look at the state $c_4 s_{00}$. The first edge is coming from the state $c_3 s_{00}$, which can be reached by a path of weight 2. The edge itself has a weight 1, so by using this edge, the state $c_4 s_{00}$ can be reached by a path of weight 3. The other option is to take the path through $c_3 s_{02}$, which has a total weight of 2 (2 for the path to $c_3 s_{02}$, 0 for the edge). As this is a better choice, the first edge is eliminated from the trellis. The other states are evaluated in a similar manner.

One last thing worth mentioning is the last portion of the trellis, where we have only two states in each column due to the "cropping" of matrix $\mathbb{H}$ in the last $h-1$ sections. This ensures that we are only left with a single state after traversing the whole trellis.

When the forward part of the Viterbi algorithm is finished (when we reach the end of the trellis), we go back from the rightmost state using the edges that were not deleted and construct the stego object $\mathbf{y}$ from their labels (note that the edge in the column with label 8 is labeled "1" although it is horizontal, because it corresponds to adding the zero column to the syndrome).

4.2 Connection to prior work

Readers familiar with the coding theory may recognize the above construction as a particular instance of a convolutional code represented by its syndrome trellis (see Chapters 25 and 48 in Ref. 26). In this case, we use the optimal decoder, the Viterbi algorithm, to implement the optimal binary quantizer. This particular version of the Viterbi algorithm was suggested by Sidorenko and Zyablov.[27]

Since Shannon[9] introduced the problem of source coding with fidelity criterion in 1959, convolutional codes were probably the first "practical" codes used for this problem. This is because the gap between the bound on the expected per-element distortion and the distortion obtained using the optimal encoding algorithm (the Viterbi algorithm) decreases exponentially[28, 29] with the constraint length of the code. The complexity of the Viterbi algorithm is linear in the blocklength of the code, but exponential in its constraint length (the number of trellis states grows exponentially in the constraint length). This makes convolutional codes (of small constraint length) suitable for our application because the entire cover object can be used and the speed can be traded for performance by adjusting the constraint length of the code. By increasing the constraint length, we can achieve the average per-element distortion that is arbitrary close to the bounds mentioned in Section 2.

Convolutional codes, when implemented with linear-feedback shift-registers (see Chapter 48 in Ref. 26), are not well suited for small relative payloads α (code rates $R = 1 - \alpha$ close to 1). This is because k registers are needed in order to implement a scheme of $\alpha = 1/k$ and the complexity of the Viterbi algorithm grows exponentially in k. Puncturing (see Chapter 48 in Ref. 26), which is often used with high-rate convolutional codes, is not a good approach either as it gives sub-optimal results. It has been shown by many authors (such as the work of Sidorenko and Zyablov[27]) that optimal decoding of convolutional codes (our binary quantizer) with high rates can be carried out in the dual domain on the syndrome trellis with a much lower complexity. The constraint length of the convolutional code translates to the bound on the number of rows in the submatrix $\hat{\mathbb{H}}$ and is thus directly related to the constraint height of the syndrome-trellis code.

5. ANALYSIS OF THE PROPOSED SCHEME

In this section, we analyze the embedding algorithm, its complexity, and design, and supply additional implementation details for practitioners. Additionally, a modification of the algorithm is described that is suitable for the wet paper channel.

5.1 Implementation details

In Section 4.1, we described the Viterbi algorithm for the parity-check matrix $\mathbb{H}$ of size $b \times wb$. In fact, the only assumption used by the algorithm was the fact that no column of the parity-check matrix contained nonzero elements in a segment longer than the constraint height h. This fact enabled us to construct the syndrome trellis with 2^h states, which means that the number of steps of the Viterbi algorithm per one column is bounded by 2^h.

In practice, the trellis is built on the fly because only the structure of the submatrix $\hat{\mathbb{H}}$ is needed (see the pseudocode in Figure 3). As can be seen from the last two columns of the trellis in Figure 2, the connectivity

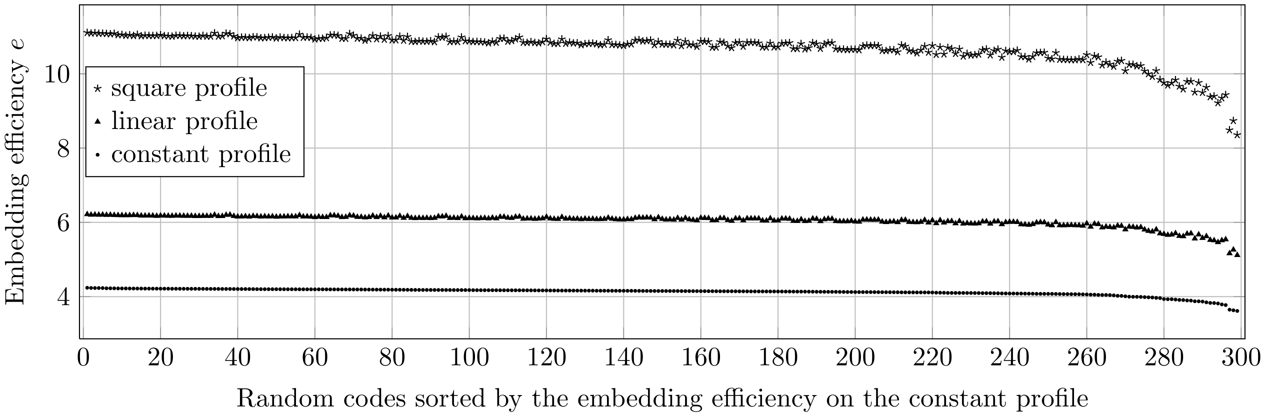

Figure 5. Embedding efficiency of 300 random syndrome-trellis codes satisfying the design rules for relative payload $\alpha = 1/2$ and constraint height $h = 10$. All codes were evaluated by the Viterbi algorithm with a random cover object of $n = 10^6$ elements and a random message on the constant, linear, and square profile. Codes are shown in the order determined by their embedding efficiency evaluated on the constant profile. This experiment suggests that codes good for the constant profile will be good for any other profile. Codes designed for different relative payloads have similar behavior.

between trellis columns is highly regular which can be used to speed up the implementation. For example, consider the column labeled by "6" in this trellis. To find the shortest path leading to state 00 and 10, only the shortest paths from these states in the previous column are needed. This idea was also used to "vectorize" the calculations done over all the states.

In the forward part of the algorithm, we need to store one bit (the label of the incoming edge) to be able to reconstruct the path in the backward run. This space complexity is linear and should not cause any difficulty, since for $h = 10$, $n = 10^6$, the total of $2^{10} \cdot 10^6/8$ bytes ($\approx$ 122MB) of space is required. If less space is available, we can always run the algorithm on smaller blocks, say $n = 10^4$, without a noticeable performance drop. If we are only interested in the total distortion $D(\mathbf{x}, \mathbf{y})$ and not the stego object itself, this information does not need to be stored at all.

5.2 Design of good syndrome-trellis codes

A natural question regarding a practical application of syndrome-trellis codes is how to find submatrices $\hat{\mathbb{H}}$ that would give the embedding algorithm performance close to the bound (see Section 2.1). If $\hat{\mathbb{H}}$ depended on the distortion profile, the profile would have to be somehow communicated to the receiver. Fortunately, this is not the case and a submatrix $\hat{\mathbb{H}}$ optimized for one profile seems to be good for other profiles as well. In this section, we study these issues experimentally and describe a practical algorithm for obtaining good submatrices.

Let us suppose that we wish to design a submatrix $\hat{\mathbb{H}}$ of size $h \times w$ for a given constraint height h and relative payload $\alpha = 1/w$. The authors were unable to find a better algorithm than an exhaustive search guided with some simple design rules.

First, $\hat{\mathbb{H}}$ should not have identical columns because the syndrome trellis would then contain two or more different paths with exactly the same weight, which would lead to an overall decrease in performance. By running an exhaustive search over small matrices, we have observed that the best submatrices $\hat{\mathbb{H}}$ had ones in its first and last rows. For example, when $h = 7$ and $w = 4$, more than 97% of the best 1000 codes obtained from the exhaustive search satisfied this rule. Thus, we searched for good matrices among those that do not contain identical columns and with all bits in the first and last rows set to 1 (the remaining bits are assigned at random). In practice, we randomly generated $10 - 1000$ submatrices satisfying these rules and estimated their performance (embedding efficiency) experimentally by running the Viterbi algorithm with random covers and messages. For a reliable estimate, cover objects of size at least $n = 10^6$ are required.

To investigate the stability of the design w.r.t. to the profile, the following experiment was conducted. We fixed $h = 10$ and $w = 2$, which correspond to code design for $\alpha = 1/2$. The code design procedure was simulated by randomly generating 300 submatrices $\hat{\mathbb{H}}_1, \ldots, \hat{\mathbb{H}}_{300}$ satisfying the above design rules. The goodness of the code was evaluated using the embedding efficiency ($e = \alpha/d$) by running the Viterbi algorithm on a random cover object (of size $n = 10^6$) and a random message. This was repeated independently for all three profiles from Section 2.2. Figure 5 shows the embedding efficiency after ordering all 300 codes by their performance on the constant profile. Because the codes with a high embedding efficiency on the constant profile exhibit high efficiency for the other profiles, the code design is stable w.r.t. the profile and one matrix may be used for multiple profiles.

5.3 Wet paper channel

As described in Section 3, the wet paper channel is an important case for steganography. Unfortunately, no practical solution that can work with an arbitrary distortion profile close to the bound is currently known. The syndrome-trellis coding and the Viterbi algorithm can be applied to this channel as well. However, the probability of not being able to embed a message with a finite cost (without flipping some wet elements) may be positive and depends on the number of wet elements, the payload, and the code. The goal is to make this probability very small or to make sure that the number of wet elements that must be changed is small (e.g., one or two). We now describe two different approaches to address this problem.

Let us assume that the wet channel is iid with probability of an element being wet $0 \leq \tau < 1$. This assumption is plausible because the cover elements can be permuted using a stego key before embedding. For the wet paper channel, the relative payload is defined w.r.t. the dry elements as $\alpha = m/|\{i|\rho_i < \infty\}|$. When designing the code for the wet paper channel with n element covers, relative wetness τ, and desired relative payload α, the parity-check matrix $\mathbb{H}$ has to be of the size $[(1-\tau)\alpha n] \times n$.

The random permutation makes it less likely for the Viterbi algorithm to fail to embed a message without having to flip some wet elements. The probability of failure, p_{w}, decreases with decreasing α and τ and it also depends on the constraint height h. From practical experiments with $n = 10^6$ cover elements, $\tau = 0.8$, and $h = 10$, we estimated from 10^3 independent runs $p_{\mathrm{w}} \doteq 0.24$ for $\alpha = 1/2$, $p_{\mathrm{w}} \doteq 0.009$ for $\alpha = 1/4$, and $p_{\mathrm{w}} \doteq 0$ for $\alpha = 1/10$. In practice, the sender can reserve a small number of bits, say k bits, a small portion of the image (using a stego key), and communicate in it the pseudo-random permutation (out of 2^k possible permutations) that leads to a finite embedding cost. This way, the probability of having to modify a wet element is at most p_{w}^{2k}, which can be made arbitrarily small.

Alternatively, the sender may be content with modifying a small number of wet elements, say one or two, without affecting the statistical detectability in any significant manner. Making use of this fact, one can set the distortion of all wet cover elements to $\hat{\rho}_i = C$, $C > \sum_{\rho_i < \infty} \rho_i$ and $\hat{\rho}_i = \rho_i$ for i dry. The weight c of the best path through the syndrome trellis obtained by the Viterbi algorithm with distortion $\hat{\rho}_i$ can be written in the form $c = n_{\mathrm{c}}C + c'$, where n_{c} is the smallest number of wet cover elements that had to be changed and c' is the smallest weight of the path over the elements that are allowed to be changed. From 10^3 independent experiments with $\alpha = 1/2$, $\tau = 0.5$, $n = 10^6$, and $h = 10$, in 82 cases one wet element had to be changed and three cases required changing two wet elements (out of total $5 \cdot 10^5$ wet elements). This suggests that this approach can be very useful in practical steganography, since the number of wet elements forced to be changed is very small and its distribution will be highly concentrated around zero.

6. EXPERIMENTAL RESULTS

We implemented the Viterbi algorithm in C++ and optimized its performance by using Streaming SIMD Extensions instructions. Based on the distortion profile, the algorithm chooses between the float and 1 byte unsigned integer data type to represent the weight of the paths in the trellis. Optimized version of the Viterbi algorithm and several examples in Matlab and C++ can be obtained from `http://dde.binghamton.edu/download/syndrome/`. The following results were obtained using an Intel Core2 X6800 2.93GHz CPU machine utilizing a single CPU core.

By using the search method described in Section 5.2, we found good syndrome-trellis codes of constraint height $h \in \{6, \ldots, 12\}$ and relative payloads $\alpha = 1/w$, $w \in \{2, \ldots, 20\}$. Some of these codes, described in an integer notation, are shown in Table 1. Codes of arbitrary rational relative payload $\alpha \leq 1/2$ can be obtained by

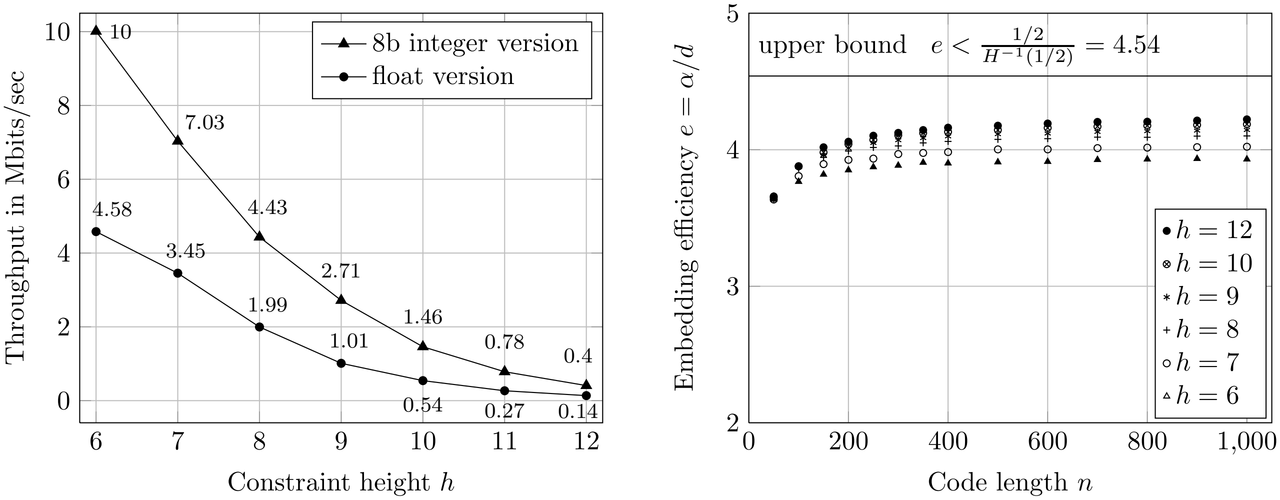

Figure 6. Results for the syndrome-trellis codes designed for relative payload $\alpha = 1/2$. (left) Average number of cover elements ($\times 10^6$) quantized per second (throughput) shown for different constraint heights and two different implementations. (right) Average embedding efficiency of the embedding scheme shown for different code lengths n (number of cover elements) and a constant distortion profile. Codes of length $n > 1000$ have similar performance as for $n = 1000$. Each point was obtained as an average over 1000 samples.

combining suitable submatrices as described in Section 4.1. In practice, almost every code satisfying the design rules is equally good. This fact can also be seen from Figure 5, where 300 random codes are evaluated over different profiles.

Figure 7 shows the comparison of syndrome-trellis codes for three profiles and for the wet paper channels with the constant profile and a varying degree of wetness τ. For a given relative payload α and constraint height h, the same submatrix $\hat{\mathbb{H}}$ was used for all profiles. This demonstrates the versatility of the proposed construction, since the information about the profile does not need to be shared, or, perhaps more importantly, the profile does not need to be known a priori.

Figure 6 shows the average throughput (the number of cover elements n quantized per second) based on the used data type. In practice, 1–5 seconds were enough to process a cover object with $n = 10^6$ elements. In the same figure, we show the embedding efficiency obtained from very short codes for the constant profile. This result shows that the average performance of the syndrome-trellis codes quickly approaches its maximum. This is again an advantage, since some applications may require using short blocks.

7. CONCLUSION

The principle of minimal embedding impact is an important design rule for constructing steganographic systems. The idea is to assign each cover element with an embedding cost (distortion) and then embed a given payload with as small distortion as possible. In practice, syndrome codes are typically used to achieve this goal. Our current work is a significant contribution in this direction because it gives steganographers a general and very flexible tool. The four main highlights that distinguish the proposed coding method from prior art are: (1) our approach can achieve the payload-distortion bound for an arbitrary distortion profile and is flexible enough to work with wet cover elements (the wet paper channel), (2) the scheme can be used to design embedding algorithms for arbitrary values of relative payload, (3) the time complexity (speed) can be traded for performance, giving the practitioners the option of a flexible design, (4) the linear complexity w.r.t. the code length allows using the entire cover object at once when embedding.

The proposed coding method can be instantly incorporated into the design of embedding algorithms that utilize side information at the sender, such as variants of perturbed quantization, MMx, or the method of Ref.[8] These methods can now be implemented in a much more rigorous manner. The designers can use the proposed near-optimal codes as an off-the-shelf method and instead concentrate on the distortion assignment as the only unresolved design choice. This will lead to a more streamlined steganography design. As part of our future research, we plan to study the important problem of selecting the embedding distortion at each cover element so that minimizing distortion corresponds to minimizing statistical detectability.

ACKNOWLEDGMENTS

The work on this paper was supported by the Air Force Office of Scientific Research under the research grants FA9550-08-1-0084 and FA9550-09-1-0147. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation there on. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied of AFOSR or the U.S. Government.

REFERENCES

1. Fridrich, J., [*Steganography in Digital Media: Principles, Algorithms, and Applications*], Cambridge University Press (2009).
2. Fridrich, J. and Filler, T., "Practical methods for minimizing embedding impact in steganography," in [*Proceedings SPIE, Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents IX*], Delp, E. J. and Wong, P. W., eds., **6505**, 02–03 (January 29–February 1, 2007).
3. Filler, T. and Fridrich, J., "Fisher information determines capacity of ϵ-secure steganography," in [*Information Hiding, 11th International Workshop*], Katzenbeisser, S. and Sadeghi, A.-R., eds., Lecture Notes in Computer Science **5806**, 31–47, Springer-Verlag, New York, Darmstadt, Germany (June 7–10, 2009).
4. Westfeld, A., "High capacity despite better steganalysis (F5 – a steganographic algorithm)," in [*Information Hiding, 4th International Workshop*], Moskowitz, I. S., ed., Lecture Notes in Computer Science **2137**, 289–302, Springer-Verlag, New York, Pittsburgh, PA (April 25–27, 2001).
5. Fridrich, J., Pevný, T., and Kodovský, J., "Statistically undetectable JPEG steganography: Dead ends, challenges, and opportunities," in [*Proceedings of the 9th ACM Multimedia & Security Workshop*], Dittmann, J. and Fridrich, J., eds., 3–14 (September 20–21, 2007).
6. Kim, Y., Duric, Z., and Richards, D., "Modified matrix encoding technique for minimal distortion steganography," in [*Information Hiding, 8th International Workshop*], Camenisch, J. L., Collberg, C. S., Johnson, N. F., and Sallee, P., eds., Lecture Notes in Computer Science **4437**, 314–327, Springer-Verlag, New York, Alexandria, VA (July 10–12, 2006).
7. Kodovský, J. and Fridrich, J., "Influence of embedding strategies on security of steganographic methods in the JPEG domain," in [*Proceedings SPIE, Electronic Imaging, Security, Forensics, Steganography, and Watermarking of Multimedia Contents X*], Delp, E. J. and Wong, P. W., eds., **6819**, 2 1–2 13 (January 27–31, 2008).
8. Sachnev, V., Kim, H. J., and Zhang, R., "Less detectable JPEG steganography method based on heuristic optimization and BCH syndrome coding," in [*Proceedings of the 11th ACM Multimedia & Security Workshop*], Dittmann, J., Craver, S., and Fridrich, J., eds., 131–140 (September 7–8, 2009).
9. Shannon, C. E., "Coding theorems for a discrete source with a fidelity criterion," *IRE Nat. Conv. Rec.* **4**, 142–163 (1959).
10. Fridrich, J., Goljan, M., Soukal, D., and Lisoněk, P., "Writing on wet paper," *IEEE Transactions on Signal Processing, Special Issue on Media Security* **54**(10), 3923–3935 (2005).
11. Fridrich, J., Goljan, M., and Soukal, D., "Perturbed quantization steganography," *ACM Multimedia System Journal* **11**(2), 98–107 (2005).
12. Crandall, R., "Some notes on steganography," *Steganography Mailing List*, available from `http://os.inf.tu-dresden.de/~westfeld/crandall.pdf` (1998).
13. Bierbrauer, J., "On Crandall's problem." Personal communication available from http://www.ws.binghamton.edu/fridrich/covcodes.pdf (1998).
14. van Dijk, M. and Willems, F., "Embedding information in grayscale images," in [*Proceedings of the 22nd Symposium on Information and Communication Theory*], 147–154 (May 15–16, 2001).

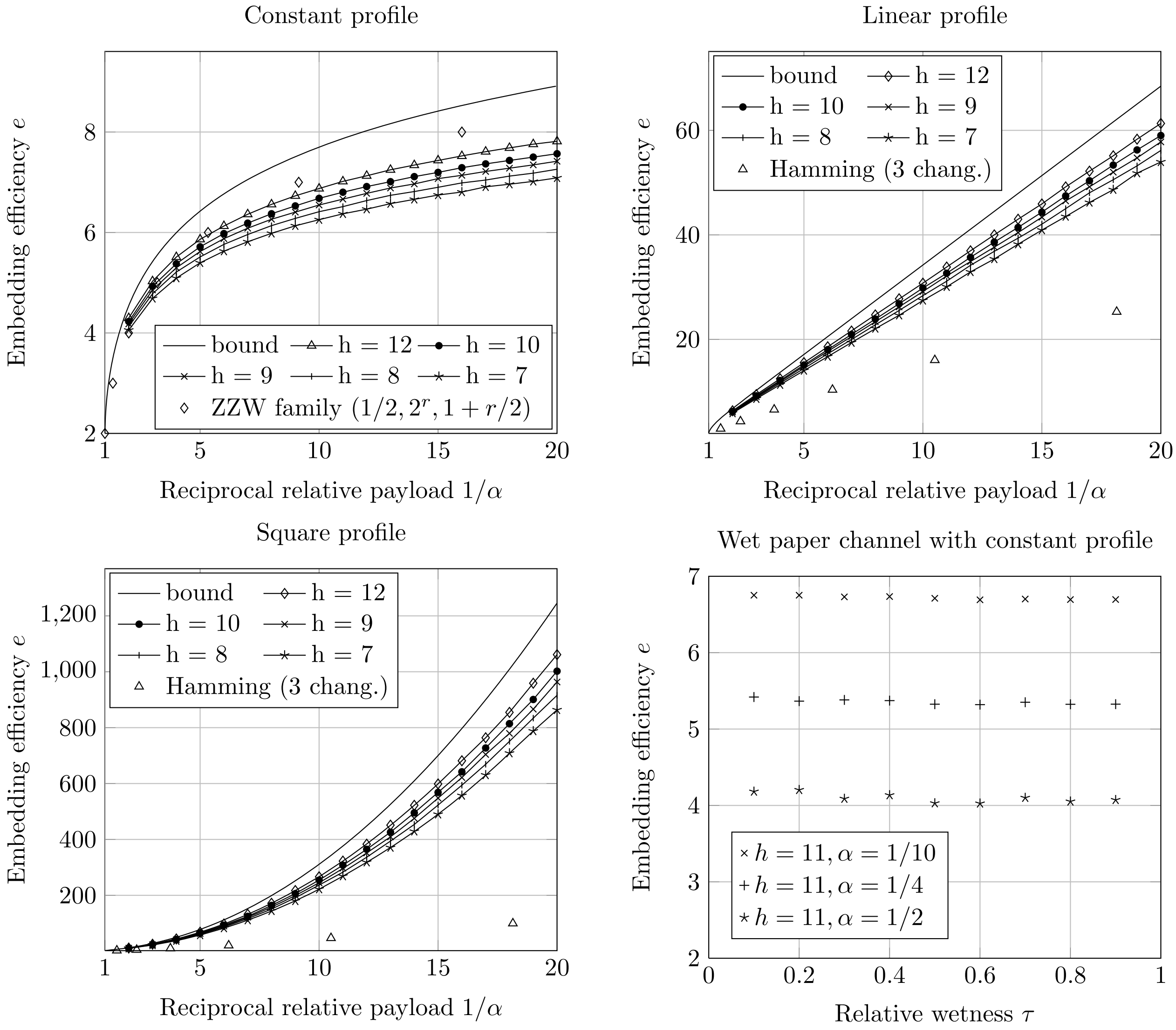

Figure 7. Embedding efficiency of syndrome-trellis codes for three distortion profiles and a family of wet paper channels. Each point was obtained by running the Viterbi algorithm with $n = 10^6$ cover elements. Hamming codes were applied on a block-by-block basis on cover objects with $n = 10^5$ elements with a brute-force search making up to three changes as proposed in Ref. 6.

$h = 7$	$\alpha = 1/2$	$(71, 109)$	$\alpha = 1/6$	$(73, 83, 95, 103, 109, 123)$
	$\alpha = 1/3$	$(95, 101, 121)$	$\alpha = 1/7$	$(69, 77, 93, 107, 111, 115, 121)$
	$\alpha = 1/4$	$(81, 95, 107, 121)$	$\alpha = 1/8$	$(69, 79, 81, 89, 93, 99, 107, 119)$
	$\alpha = 1/5$	$(75, 95, 97, 105, 117)$	$\alpha = 1/9$	$(69, 79, 81, 89, 93, 99, 107, 119, 125)$
$h = 10$	$\alpha = 1/2$	$(627, 997)$	$\alpha = 1/6$	$(573, 659, 747, 793, 959, 973)$
	$\alpha = 1/3$	$(621, 659, 919)$	$\alpha = 1/7$	$(589, 601, 679, 751, 831, 851, 989)$
	$\alpha = 1/4$	$(601, 703, 893, 955)$	$\alpha = 1/8$	$(589, 631, 689, 713, 821, 899, 971, 1013)$
	$\alpha = 1/5$	$(635, 725, 775, 929, 975)$	$\alpha = 1/9$	$(531, 589, 603, 699, 735, 789, 857, 903, 1017)$

Table 1. A list of good syndrome-trellis codes obtained from a computer search over 1000 random codes as described in Section 5.2. The columns of the submatrix $\hat{\mathbb{H}}$ are written in an integer notation – the least significant bit of the number is the bit in the first row (see Section 4.1 for a detailed description and an example).

15. Schönfeld, D. and Winkler, A., “Embedding with syndrome coding based on BCH codes,” in [*Proceedings of the 8th ACM Multimedia & Security Workshop*], Voloshynovskiy, S., Dittmann, J., and Fridrich, J., eds., 214–223 (September 26–27, 2006).
16. Fridrich, J. and Soukal, D., “Matrix embedding for large payloads,” *IEEE Transactions on Information Forensics and Security* **1**(3), 390–394 (2006).
17. Bierbrauer, J. and Fridrich, J., “Constructing good covering codes for applications in steganography,” *LNCS Transactions on Data Hiding and Multimedia Security* **4920**, 1–22 (2008).
18. Zhang, W. and Wang, X., “Generalization of the ZZW embedding construction for steganography,” *Information Forensics and Security, IEEE Transactions on* (2009).
19. Fridrich, J., “Asymptotic behavior of the ZZW embedding construction,” *IEEE Transactions on Information Forensics and Security* **4**(1), 151–153 (2009).
20. Fridrich, J., Goljan, M., and Soukal, D., “Wet paper codes with improved embedding efficiency,” *IEEE Transactions on Information Forensics and Security* **1**(1), 102–110 (2006).
21. Zhang, W. and Zhu, X., “Improving the embedding efficiency of wet paper codes by paper folding,” *IEEE Signal Processing Letters* **16**(9), 794–797 (2009).
22. Filler, T. and Fridrich, J., “Wet ZZW construction for steganography,” in [*First IEEE International Workshop on Information Forensics and Security*], (December 6–9 2009).
23. Filler, T., Ker, A. D., and Fridrich, J., “The Square Root Law of steganographic capacity for Markov covers,” in [*Proceedings SPIE, Electronic Imaging, Security and Forensics of Multimedia XI*], Memon, N. D., Delp, E. J., Wong, P. W., and Dittmann, J., eds., **7254**, 08 1–08 11 (January 18–21, 2009).
24. Pevný, T., Bas, P., and Fridrich, J., “Steganalysis by subtractive pixel adjacency matrix,” in [*Proceedings of the 11th ACM Multimedia & Security Workshop*], Dittmann, J., Craver, S., and Fridrich, J., eds., 75–84 (September 7–8, 2009).
25. Kodovský, J. and Fridrich, J., “Calibration revisited,” in [*Proceedings of the 11th ACM Multimedia & Security Workshop*], Dittmann, J., Craver, S., and Fridrich, J., eds., 63–74 (September 7–8, 2009).
26. MacKay, D., [*Information Theory, Inference, and Learning Algorithms*], Cambridge University Press (2003). Can be obtained from `http://www.inference.phy.cam.ac.uk/mackay/itila/`.
27. Sidorenko, V. and Zyablov, V., “Decoding of convolutional codes using a syndrome trellis,” *IEEE Transactions on Information Theory* **40**(5), 1663–1666 (1994).
28. Viterbi, A. and Omura, J., “Trellis encoding of memoryless discrete-time sources with a fidelity criterion,” *IEEE Transactions on Information Theory* **20**(3), 325–332 (1974).
29. Hen, I. and Merhav, N., “On the error exponent of trellis source coding,” *IEEE Transactions on Information Theory* **51**(11), 3734–3741 (2005).