Practical Methods for Minimizing Embedding Impact in Steganography

Jessica Fridrich and Tomáš Filler

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

ABSTRACT

In this paper, we propose a general framework and practical coding methods for constructing steganographic schemes that minimize the statistical impact of embedding. By associating a cost of an embedding change with every element of the cover, we first derive bounds on the minimum theoretically achievable embedding impact and then propose a framework to achieve it in practice. The method is based on syndrome codes with low-density generator matrices (LDGM). The problem of optimally encoding a message (e.g., with the smallest embedding impact) requires a binary quantizer that performs near the rate-distortion bound. We implement this quantizer using LDGM codes with a survey propagation message-passing algorithm. Since LDGM codes are guaranteed to achieve the rate-distortion bound, the proposed methods are guaranteed to achieve the minimal embedding impact (maximal embedding efficiency). We provide detailed technical description of the method for practitioners and demonstrate its performance on matrix embedding.

Keywords: Steganography, embedding impact, low-density codes, survey propagation, matrix embedding

1. INTRODUCTION

In passive warden steganography, the goal is to communicate as many bits as possible without introducing statistically detectable artifacts into the cover object. In practice, a steganographic scheme is considered secure if no existing attack can distinguish between cover and stego objects with a success better than random guessing. Formal definition of steganographic security was given by Cachin.[3]

The security of a steganographic scheme is a function of its attributes, which are (1) the cover object source whose properties are known to the attacker (Kerckhoffs' principle), (2) the embedding operation that is applied to individual cover elements to embed a message, and (3) the selection rule according to which cover elements are selected for embedding. For the sake of concreteness and without loss of generality, at times we will be referring to the cover object as image and the cover elements as pixels.

The statistical impact of embedding at a given pixel depends on many factors, such as the character of embedding modifications, their amplitude, local pixel neighborhood, etc. Assuming each pixel can be assigned a number that measures the statistical impact of making an embedding change at that pixel, we are interested in the minimal possible impact needed to embed $m \leq n$ bits in a cover object consisting of n pixels. If the embedding impact is defined simply as the number of embedding changes d, it is known[1] that $d \leq nH^{-1}(m/n)$, where $H^{-1}(x)$ is the inverse of the binary entropy function $H(x) = -x\log_2 x - (1-x)\log_2(1-x)$ on $[0, 1/2]$. For a fixed relative message length $\alpha = m/n$, this bound is achievable using syndrome codes of length n and dimension $n - m$ as $n \to \infty$.[8] Practical codes were proposed in.[1,2,6,7,14,15] Unfortunately, their performance is not very close to the bound. In this paper, we study the problem of minimizing the embedding impact, derive appropriate bounds, and propose a general framework using which practical near-optimal embedding schemes can be constructed.

The paper is structured as follows. After introducing some basic concepts, in Section 2 we define embedding impact in steganography and give a precise problem formulation. In Section 3, we describe the building blocks of the proposed framework for constructing embedding schemes with minimal embedding impact. We start Section 4 with explaining the relationship between optimal embedding and achieving the rate-distortion bound

J.F.: E-mail: fridrich@binghamton.edu, Telephone: +1 607 777 6177, Fax: +1 607 777 4464

in source coding. Then, we introduce the central element of near-optimal embedding schemes—the binary quantizer realized using LDGM codes with survey propagation message-passing algorithm. Section 5 explains other missing elements of the embedding scheme and the extraction algorithm. In Section 6, we demonstrate the performance of the proposed framework on matrix embedding. The paper is concluded in Section 7.

2. STEGANOGRAPHIC EMBEDDING SCHEME

Throughout the text, boldface symbols denote vectors or matrices while the caligraphic font is reserved for sets. The cover object is a sequence of n elements $\mathbf{g} = (g_1, \dots, g_n) \in \mathcal{G}^n$, where $\mathcal{G} = \{0, \dots, 2^r - 1\}$ and r is the number of bits needed to describe each element. Most steganographic schemes work with a finite field representation of $\mathbf{g}$ obtained through some symbol-assignment function $symb : \mathcal{G} \to \mathbb{F}_q$. For example, $symb(g_i) = g_i \bmod 2$ ($symb(g_i) = g_i \bmod 3$) assign a bit (ternary symbol) to each cover element. Thus, the cover object $\mathbf{g}$ is represented as a vector $\mathbf{x} \in \mathbb{F}_q^n$.

A steganographic scheme is a pair of embedding and extraction mappings $Emb : \mathbb{F}_q^n \times \mathcal{M} \to \mathbb{F}_q^n$, $Ext : \mathbb{F}_q^n \to \mathcal{M}$ satisfying

$$Ext(Emb(\mathbf{x}, M)) = M, \forall \mathbf{x} \in \mathbb{F}_q^n, \forall M \in \mathcal{M}, \tag{1}$$

where $\mathcal{M}$ is the set of all messages M that can be communicated. We say that the embedding capacity of the scheme is $\log |\mathcal{M}|$ bits. $Emb(\mathbf{x}, M) = \mathbf{y}$ is the finite field representation of the stego object $\mathbf{g}'$ obtained by modifying $\mathbf{g}$ so that $symb(g_i') = y_i$. For example, in LSB embedding with $symb(g_i) = g_i \bmod 2$, the LSB of the binary representation of g_i is flipped. In ± 1 embedding, g_i is randomly changed by 1 or -1. We note that the nature of the modification has a major impact on the security of the steganographic scheme.

The impact of making an embedding change at pixel i will be measured using a scalar value $\rho_i \geq 0$. The total embedding impact is then

$$D(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_D = \sum_{i=1}^{n} \rho_i |x_i - y_i|. \tag{2}$$

We can interpret ρ_i as the cost of making an embedding change at pixel i. This detectability measure should be designed to correlate with the statistical detectability of the embedding changes. In practice, ρ_i is usually proposed using heuristic principles. For example, for a non-negative parameter α and weight factors $\omega_i \geq 0$

$$\rho_i = \omega_i |g_i - g_i'|^\alpha. \tag{3}$$

If the embedding change is probabilistic, then we understand (3) as the expected value. Below, we give a few examples of (3) typically used in steganography.

If $\omega_i = 1$ for all i and $|g_i - g_i'| = 1$, D is the total number of embedding changes. For $\omega_i = 1$ and $\alpha = 2$, D is the energy of modifications. Wet paper coding[6] can be modeled by setting $\omega_i = 1$ for $i \in Dry$ and $\omega_i = 0$ otherwise, for some index set $Dry \subset \{1, \dots, n\}$. In general, the weighting factors may depend on the local texture to reflect the fact that embedding changes in textured (or noisy) areas are more difficult to detect than changes in smooth segments of the cover image.

The impact ρ_i may also be determined from some side-information available to the sender as in Perturbed Quantization steganography (PQ).[5] For example, let us assume that the cover is a TIFF image sampled at 16 bits per channel. The sender wishes to embed a message while decreasing the color depth to a true-color 8-bit per channel image while minimizing the combined quantization and embedding distortion. Let z_i be the 16-bit color value and let $Q = 2^8$ be the quantization step for the color depth reduction. The quantization error is $e_i = Q|z_i/Q - [z_i/Q]|$, $0 \leq e_i \leq Q/2$, and the error when rounding z_i to the opposite direction is $Q - e_i$ leading to embedding distortion as the difference between both errors $\rho_i = Q - 2e_i$. In PQ, coefficients are selected for which $e_i \approx Q/2$ because for such coefficients, the embedding distortion is the smallest. Also note that in this case, since the quantization error is approximately uniform on $[-Q/2, Q/2]$, when sorting ρ_i by their values the resulting profile will be well modeled with a straight line.

We point out that (2) implicitly assumes that the embedding impact is additive because it is defined as a sum of detectability measures at individual pixels. In general, however, the embedding modifications could be

interacting among themselves, reflecting the fact that making two changes to adjacent pixels might be more detectable than making the same changes to two pixels far apart from each other. A detectability measure that takes interaction among pixels into account would not be additive. If the density of embedding changes is low, however, the additivity assumption is plausible because the distances between modified pixels will generally be large and the embedding changes will not interfere much.

2.1. Problem formulation

The central problem investigated in this paper is design of steganographic schemes whose expected embedding impact $E[D(\mathbf{x},\mathbf{y})]$ is as small as possible for covers of length n, embedding capacity m, and detectability measure ρ_i. The expected value is taken over all cover objects $\mathbf{x}$ and messages of length m.

From now on, we will constrain ourselves to the binary case $\mathbb{F}_q = \mathrm{GF}(2)$ when the symbol-assignment function assigns a bit to each cover element. In Appendix A, we show that for the binary case the minimal expected embedding impact is

$$D(n,m,\rho) = \sum_{i=1}^{n} p_i \rho_i, \tag{4}$$

where

$$p_i = \frac{e^{-\lambda\rho_i}}{1+e^{-\lambda\rho_i}}, \tag{5}$$

and λ is given by the following constraint

$$\sum_{i=1}^{n} H(p_i) = m, \tag{6}$$

where $H(x)$ is the binary entropy function. Moreover, the embedding operation will on average modify the i-th pixel with probability p_i. Thus, if we design an embedding scheme that modifies pixels with these probabilities and communicates m bits, it will leave the minimal possible embedding impact.

3. THE BASIC FRAMEWORK

In this section, we describe the framework for constructing near-optimal embedding schemes using syndrome codes. The individual elements from which the framework is composed are explained in detail in the next two sections. From now on, all vectors are column vectors and all arithmetic operations between binary vectors and matrices are carried in the GF(2). A good text on coding theory is.[17]

Let us assume that the receiver knows the relative message length $\alpha = m/n$ and thus the number of secret message bits m. Indeed, this can be either pre-agreed or a small, key-dependent portion of the cover can be reserved to communicate a suitably quantized α encoded using a few bits. Let $\mathcal{C}$ be an $[n, n-m]$ binary code $\mathcal{C}$ with an $n \times (n-m)$ generator matrix $\mathbf{G}$ and an $m \times n$ parity check matrix $\mathbf{H}$. Both matrices are shared between the sender and the recipient. Let $\mathcal{C}(\mathbf{m}) = \{\mathbf{u} \in \{0,1\}^n | \mathbf{Hu} = \mathbf{m}\}$ be the coset corresponding to syndrome $\mathbf{m} \in \{0,1\}^m$ ($\mathbf{m}$ is the secret message). The following embedding scheme communicates m bits in an n-element cover $\mathbf{x}$

$$\begin{aligned} \mathbf{y} = Emb(\mathbf{x},\mathbf{m}) &\triangleq \arg\min_{\mathbf{u}\in\mathcal{C}(\mathbf{m})} \|\mathbf{x}-\mathbf{u}\|_D \\ Ext(\mathbf{y}) &= \mathbf{Hy} = \mathbf{m}. \end{aligned} \tag{7}$$

Here, $\mathbf{y}$ are the bits assigned to the stego image. In other words, in an attempt to minimize the embedding impact, the sender selects such a member $\mathbf{y}$ of the coset $\mathcal{C}(\mathbf{m})$ that is closest to $\mathbf{x}$ (closest in metric $\|.\|_D$).

Let $\mathbf{v_m} \in \mathcal{C}(\mathbf{m})$ arbitrary. Then,

$$\min_{\mathbf{u}\in\mathcal{C}(\mathbf{m})} \|\mathbf{x}-\mathbf{u}\|_D = \min_{\mathbf{c}\in\mathcal{C}} \|\mathbf{x}-(\mathbf{v_m}+\mathbf{c})\|_D = D(\mathbf{x}-\mathbf{v_m},\mathcal{C}) = \min_{\mathbf{w}\in\{0,1\}^{n-m}} \|\mathbf{x}-\mathbf{v_m}-\mathbf{Gw}\|_D\,, \tag{8}$$

where we denoted by $D(\mathbf{x}-\mathbf{v_m},\mathcal{C})$ the distance between $\mathbf{x}-\mathbf{v_m}$ and $\mathcal{C}$. From (8), we see that embedding is a binary quantization problem. The sender needs to find $\mathbf{w} \in \{0,1\}^{n-m}$ such that $\mathbf{Gw}$ is closest to $\mathbf{x}-\mathbf{v_m}$.

Alternatively, we can say that the sender is compressing the *source* bit sequence $\mathbf{z} = \mathbf{x} - \mathbf{v_m}$ to $n-m$ *information* bits $\mathbf{w}$ so that the reconstructed vector $\mathbf{Gw}$ is as close to the source sequence as possible. Let us denote the closest codeword $\mathbf{Gw}$ as $\mathbf{c_{m,x}}$.

Assuming there exists an efficient algorithm for finding both $\mathbf{v_m}$ and $\mathbf{c_{m,x}}$, the stego object $\mathbf{y}$ is

$$\mathbf{y} = \mathbf{x} + \mathbf{c_{m,x}} - \mathbf{z} = \mathbf{c_{m,x}} + \mathbf{v_m}. \tag{9}$$

Four things need to be supplied to make the description of this embedding scheme complete. We need to describe the process by which we generate the code, the algorithm for finding $\mathbf{v_m}$, and the algorithm for binary quantization. We also need to explain why the distortion of this embedding scheme is close to the bound (4). The most difficult step in the proposed scheme is the binary quantization. In fact, it dictates the choice of the code and determines the computational complexity. This is why we start with it in the next section.

4. BINARY QUANTIZATION USING LDGM CODES

In this section, we give the implementation details for the binary quantizer, which is the central element in our embedding scheme based on syndrome codes. Here, we intentionally focus on a practical description of the method to enable the reader to implement the embedding scheme without being necessarily familiar with all technical details of the underlying material. We refer the reader to the original publications[10,16] for more details.

We lay out the method for the special case when all ρ_i are the same, postponing the non-constant detectability measure to our future work. In fact, we believe that the same framework can be used after some adjustments. The modifications are pointed out in the text.

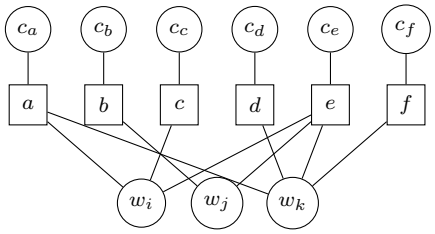
When all ρ_i are the same, D is simply the number of embedding changes and the problem of minimizing the embedding impact turns into what is known in steganography as Matrix Embedding.[4] The quantization task (8) is equivalent to finding the coset leader of $\mathcal{C}(\mathbf{m})$, which is an NP hard problem. From the binary quantization interpretation, the rate-distortion theory implies that the rate $R = 1 - m/n$ of any source encoding algorithm that compresses n bits into $n - m$ bits is bounded by $R = 1 - m/n \leq 1 - H(d)$, where $d = D/n$ is the average distortion per bit. Denoting the average number of message bits embedded per unit distortion by $e = m/D$ and recalling that $\alpha = m/n$ is the relative message length, the rate-distortion bound is recognized in its equivalent form as a bound on the maximal *embedding efficiency* e of any Matrix Embedding scheme

$$e \leq \frac{\alpha}{H^{-1}(\alpha)}. \tag{10}$$

It is known[8] that for fixed α and $n \to \infty$ this bound is saturated for almost all linear codes. This explains why the proposed framework is near-optimal for n sufficiently large. The big problem, of course, is how to find codes for which efficient algorithms exist for large n. Structured codes[1,2,14,15] and random codes[6,7] that were previously proposed do not perform too close to the bound and their complexity grows too quickly.

Wainwright and Maneva[16] recently showed that duals of LDPC codes called Low Density Generator Matrix codes (LDGM) combined with Survey Propagation (SP) message-passing algorithms could be used to construct low-complexity binary quantizers with performance very close to the rate-distortion bound. Subsequent work proved that with $n \to \infty$ compound low density codes saturate the bound with matrices whose number of ones in rows and columns is bounded.[11] We use the construction given in[16] to implement near-optimal embedding schemes.

We start with the description of the generator matrix $\mathbf{G}$. For a given message length α, we select $\mathbf{G}$ as the parity check matrix of an LDPC code optimized for the binary symmetric channel (BSC) with error probability $p = 1 - \alpha$. The matrices are generated randomly but with a constraint that the number of ones in each row follows a prespecified optimized irregular distribution. Description of algorithms for generating the distributions is given in[12] and an interactive practical algorithm for their generation is available from `http://lthcwww.epfl.ch/research/ldpcopt/`. Furthermore, as explained in more detail in Section 5, we additionally preprocess $\mathbf{G}$ by permuting its rows and columns to enable easy finding of the coset member $\mathbf{v_m}$ and fast message extraction.



$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \qquad \begin{aligned} c_a + w_i + w_k &= 0 \\ c_b + w_j &= 0 \\ c_c + w_i &= 0 \\ c_d + w_k &= 0 \\ c_e + w_i + w_j + w_k &= 0 \\ c_f + w_k &= 0 \end{aligned}$$

Figure 1. Factor graph representation of a linear code with generator matrix $\mathbf{G}$.

4.1. Graph representation of a code

Each code $\mathcal{C}$ can be represented as a factor graph in the following manner. Assuming $\mathbf{G}$ is full rank, for each codeword $\mathbf{c} \in \mathcal{C}$ there exists exactly one $\mathbf{w} \in \{0,1\}^{n-m}, \mathbf{c} = \mathbf{G}\mathbf{w}$. Thus, each codeword $\mathbf{c}$ can be uniquely associated with a vector of $2n-m$ bits $(\mathbf{c}, \mathbf{w})$, called *extended codeword*, satisfying

$$[\mathbf{I}, \mathbf{G}] \begin{pmatrix} \mathbf{c} \\ \mathbf{w} \end{pmatrix} = 0, \tag{11}$$

where $\mathbf{I}$ is the $n \times n$ unity matrix. Therefore, (11) can be viewed as a definition of a code via a bipartite graph with two types of nodes—n check nodes and $2n-m$ variable nodes. Consider, for example, the code defined in Figure 1 with generator matrix $\mathbf{G}$. The bipartite graph on the left has 6 checks $a, \ldots, f$ and 9 variable nodes.

We now introduce the following terminology and notation for factor graphs. The bits connected to the checks "from above" $c_a, c_b, c_c, \ldots$ will be called *source bits* and the bottom bits w_i, w_j, w_k will be called *information bits*. The set of all checks is denoted C and the set of all information bits V. The set of all checks connected to an information bit i is denoted $C(i)$ and the index set of all information bits connected to check a is denoted $V(a)$. We further denote $\overline{V}(a) = V(a) \cup \{a\}$, where index a is used for the source bit z_a associated with check a. For our example above, $C(i) = \{a, c, e\}, V(a) = \{i, k\}, \overline{V}(a) = \{i, k, a\}$.

4.2. Belief-propagation

One way to look at the quantization problem (8) is that $\mathbf{z} = \mathbf{x} - \mathbf{v_m}$ is a noisy codeword in a binary symmetric channel (BSC) with flip probability $p < 1/2$ and we wish to perform maximum likelihood decoding and find the closest codeword $\mathbf{c_{m,x}}$ to $\mathbf{z}$. In LDPC codes, this problem is approached using belief-propagation (BP) message-passing algorithm. It starts by forming the following probability distribution $p(\mathbf{v}, \mathbf{w})$ over the space of all possible binary vectors $(\mathbf{v}, \mathbf{w}) \in \{0,1\}^{2n-m}$. Let $\phi_a(v_a, w_{V(a)})$ be the XOR of all bits from $\overline{V}(a)$. Then,

$$p(\mathbf{v}, \mathbf{w}) = \frac{1}{Z} \prod_{i \in V} \psi_i(w_i) \prod_{a \in C} \psi_a(v_a) \prod_{a \in C} (1 - \phi_a(v_a, w_{V(a)})), \tag{12}$$

where Z is the normalization factor, $\psi_i(w_i) = 1/2, \forall i$, and $\psi_a(v_a) = 1 - p$ when $v_a = z_a$ and $\psi_a(v_a) = p$ when $v_a \neq z_a$. Note that $p(\mathbf{v}, \mathbf{w}) = 0$ for extended codewords not satisfying the XOR-SAT problem $[\mathbf{I}, \mathbf{G}] \begin{pmatrix} \mathbf{v} \\ \mathbf{w} \end{pmatrix} = 0$. Also note that the choice of information bits w_i does not influence the probability while the mismatch between source bits z_a and v_a is penalized according to the probability of bit flipping in the BSC. We denote by $p_i(0)$ the marginal probability

$$p_i(w_i = 0) = \frac{1}{Z} \sum_{\mathbf{v} \in \{0,1\}^n} \sum_{\substack{\mathbf{w} \in \{0,1\}^{n-m} \\ w_i = 0}} p(\mathbf{v}, \mathbf{w}),$$

where Z is a normalization factor. A similar expression can be obtained for $p_i(1)$. The information bits are set to either 0 or 1 based on which marginal probability is larger. When the code graph is cycle-free (a tree) the

marginal probabilities can be calculated efficiently using the belief-propagation algorithm. The algorithm is also used on graphs with cycles (such as our bipartite graph) and gives good results.

The BP algorithm is of iterative nature and consists of rounds in which checks process the messages they receive from their neighboring information bits and send back messages to the information bits. The information bits process the messages received from their checks and send messages back to their checks. The message sent by check a to information bit i in the ℓ-th iteration is the ordered pair $(M^{(\ell)}_{a\to i}(0), M^{(\ell)}_{a\to i}(1))$ and the message from information bit i to check a in the ℓ-th iteration is $(M^{(\ell)}_{i\to a}(0), M^{(\ell)}_{i\to a}(1))$. The update formulas are

$$M^{(\ell)}_{i\to a}(w_i) = \prod_{b\in C(j)\setminus a} M^{(\ell-1)}_{b\to i}(w_i)$$
$$M^{(\ell)}_{a\to i}(w_i) = \sum_{w_{V(a)}\setminus i} (1-\phi_a(v_a, w_{V(a)})) \prod_{j\in\overline{V}(a)\setminus i} M^{(\ell)}_{j\to a}(w_j).$$

After normalizing the messages so that in each round $M^{(\ell)}_{i\to a}(0) + M^{(\ell)}_{i\to a}(1) = 1$, the messages have the following probabilistic interpretation. Check a sends to its neighboring information bit i the probability that it is satisfied given the source sequence $\mathbf{z}$ and the messages received from all information bits other than i in the previous round. Information bit i sends to its neighboring check a the probability that it is 0 (or 1) given the information received from its neighboring checks other than a in the previous round. The source bits always send *the same message* to their checks: $(Pr\{v_a = 0|z_a\}, Pr\{v_a = 1|z_a\})$, which is either $(p, 1-p)$ or $(1-p, p)$. The whole process is initialized by starting with source bits sending their messages to the checks who forward the messages to the information bits (the initial message is denoted $(M^{(0)}_{i\to a}(0), M^{(0)}_{i\to a}(1))$). The BP algorithm is run till it converges (message vectors do not differ in two consecutive iterations) and the marginals are then computed from the fixed point message $\hat{M}$ in the $\hat{\ell}$-th iteration as

$$p(w_i) = \prod_{b\in C(i)} \hat{M}^{(\hat{\ell})}_{b\to i}(w_i).$$

The information bits are finally determined by choosing the value of w_i with a larger $p_i(w_i)$.

4.3. Survey propagation

The problem with the BP is that it converges only when $\mathbf{z}$ is already close (within the error-correcting distance of the associated LDPC code with parity check matrix $\mathbf{G}$), otherwise it does not converge. This is known as the folklore statement "LDPCs are poor quantizers." Because $\mathbf{z}$ is determined by the (random) message $\mathbf{m}$, it is unlikely to be close to a codeword. As a result, the BP algorithm cannot be used for embedding. The space of all codewords essentially breaks up into disjoint clusters inside which the BP will find the closest codeword. Survey propagation is an algorithm for finding the *closest cluster* to $\mathbf{z}$. It is again a message-passing algorithm in which information bits are set to their values through a series of decimation and message-passing steps.

Similar to BP, in the SP algorithm the source and information bits send messages to checks and then checks process the received messages and send messages to information bits. The bits again process the received messages and send messages back to checks, etc. The process stops when the messages sent by information bits in two consecutive passes differ by less than a small predetermined bound. After the message-passing algorithm converged, selected information bits are set to specific bits and the bipartite graph is simplified. The message-passing proceeds again on the simplified graph till convergence, a portion of the information bits are set to bits, the graph is again simplified, and the whole procedure repeats till all information bits are determined. One pass of the message-passing updates in both directions will be called *iteration*. The process of assigning the bits and simplifying the graph is called *decimation*. The whole process of running the message-passing updates till convergence followed by decimation is one *round*.

The messages exchanged by bits and checks are five-dimensional vectors of non-negative real numbers. In the ℓ-th iteration, the i-th information bit sends to check a the vector $\mathbf{M}^{(\ell)}_{i\to a} = (M^{0f\,(\ell)}_{i\to a}, M^{1f\,(\ell)}_{i\to a}, M^{0w\,(\ell)}_{i\to a}, M^{1w\,(\ell)}_{i\to a}, M^{*\,(\ell)}_{i\to a})$

Bits to checks update rules

$$
\begin{aligned}
M_{i\to a}^{0f\,(\ell)} &= \prod_{b\in C(i)\setminus\{a\}} \Big[M_{b\to i}^{0f\,(\ell-1)} + M_{b\to i}^{0w\,(\ell-1)}\Big] - \prod_{b\in C(i)\setminus\{a\}} M_{b\to i}^{0w\,(\ell-1)} \\
M_{i\to a}^{1f\,(\ell)} &= \prod_{b\in C(i)\setminus\{a\}} \Big[M_{b\to i}^{1f\,(\ell-1)} + M_{b\to i}^{1w\,(\ell-1)}\Big] - \prod_{b\in C(i)\setminus\{a\}} M_{b\to i}^{1w\,(\ell-1)} \\
M_{i\to a}^{0w\,(\ell)} &= \prod_{b\in C(i)\setminus\{a\}} \Big[M_{b\to i}^{0f\,(\ell-1)} + M_{b\to i}^{0w\,(\ell-1)}\Big] - \prod_{b\in C(i)\setminus\{a\}} M_{b\to i}^{0w\,(\ell-1)} - \sum_{c\in C(i)\setminus\{a\}} M_{c\to i}^{0f\,(\ell-1)} \prod_{b\in C(i)\setminus\{a,c\}} M_{b\to i}^{0w\,(\ell-1)} \\
M_{i\to a}^{1w\,(\ell)} &= \prod_{b\in C(i)\setminus\{a\}} \Big[M_{b\to i}^{1f\,(\ell-1)} + M_{b\to i}^{1w\,(\ell-1)}\Big] - \prod_{b\in C(i)\setminus\{a\}} M_{b\to i}^{1w\,(\ell-1)} - \sum_{c\in C(i)\setminus\{a\}} M_{c\to i}^{1f\,(\ell-1)} \prod_{b\in C(i)\setminus\{a,c\}} M_{b\to i}^{1w\,(\ell-1)} \\
M_{i\to a}^{*\,(\ell)} &= w_{\text{info}} \prod_{b\in C(i)\setminus\{a\}} M_{b\to i}^{*\,(\ell-1)}
\end{aligned}
\tag{13}
$$

Checks to bits update rules

$$
\begin{aligned}
M_{a\to i}^{0f\,(\ell)} &= \frac{1}{2}\Big(\prod_{j\in\bar V(a)\setminus\{i\}} \Big[M_{j\to a}^{0f\,(\ell)} + M_{j\to a}^{1f\,(\ell)}\Big] + \prod_{j\in\bar V(a)\setminus\{i\}} \Big[M_{j\to a}^{0f\,(\ell)} - M_{j\to a}^{1f\,(\ell)}\Big]\Big) \\
M_{a\to i}^{1f\,(\ell)} &= \frac{1}{2}\Big(\prod_{j\in\bar V(a)\setminus\{i\}} \Big[M_{j\to a}^{0f\,(\ell)} + M_{j\to a}^{1f\,(\ell)}\Big] - \prod_{j\in\bar V(a)\setminus\{i\}} \Big[M_{j\to a}^{0f\,(\ell)} - M_{j\to a}^{1f\,(\ell)}\Big]\Big) \\
M_{a\to i}^{0w\,(\ell)} &= \prod_{j\in\bar V(a)\setminus\{i\}} \Big[M_{j\to a}^{*\,(\ell)} + M_{j\to a}^{1w\,(\ell)} + M_{j\to a}^{0w\,(\ell)}\Big] - w_{\text{sou}} \prod_{j\in V(a)\setminus\{i\}} \Big[M_{j\to a}^{1w\,(\ell)} + M_{j\to a}^{0w\,(\ell)}\Big] \\
M_{a\to i}^{1w\,(\ell)} &= M_{a\to i}^{0w\,(\ell)} \\
M_{a\to i}^{*\,(\ell)} &= \prod_{j\in\bar V(a)\setminus\{i\}} \Big[M_{j\to a}^{*\,(\ell)} + M_{j\to a}^{1w\,(\ell)} + M_{j\to a}^{0w\,(\ell)}\Big]
\end{aligned}
\tag{14}
$$

Bias equations calculated in ℓ-th iteration

$$
\begin{aligned}
\mu_i(0) &= \prod_{a\in C(i)} \Big[M_{a\to i}^{0f\,(\ell)} + M_{a\to i}^{0w\,(\ell)}\Big] - \prod_{a\in C(i)} M_{a\to i}^{0w\,(\ell)} - \sum_{b\in C(i)} M_{b\to i}^{0f\,(\ell)} \prod_{a\in C(i)\setminus\{b\}} M_{a\to i}^{0w\,(\ell)} \qquad \mu_i(*) = w_{\text{info}} \prod_{a\in C(i)} M_{a\to i}^{*\,(\ell)} \\
\mu_i(1) &= \prod_{a\in C(i)} \Big[M_{a\to i}^{1f\,(\ell)} + M_{a\to i}^{1w\,(\ell)}\Big] - \prod_{a\in C(i)} M_{a\to i}^{1w\,(\ell)} - \sum_{b\in C(i)} M_{b\to i}^{1f\,(\ell)} \prod_{a\in C(i)\setminus\{b\}} M_{a\to i}^{1w\,(\ell)}
\end{aligned}
\tag{15}
$$

Figure 2. Update equations for message-passing in the SP algorithm.

and the a-th check sends to the i-th information bit the vector $\mathbf{M}_{a\to i}^{(\ell)} = (M_{a\to i}^{0f\,(\ell)}, M_{a\to i}^{1f\,(\ell)}, M_{a\to i}^{0w\,(\ell)}, M_{a\to i}^{1w\,(\ell)}, M_{a\to i}^{*\,(\ell)})$. The source bits *always send the same message to their checks*:

$$\mathbf{M}_{z_a\to a}^{(\ell)} = \big(\psi_a(0), \psi_a(1), 0, 0, w_{sou}\big), \tag{16}$$

where w_{sou} is a constant, typically $w_{sou} = 1.1$, and $\psi_a(1) = z_a e^{\gamma} + (1 - z_a)e^{-\gamma}, \psi_a(0) = \frac{1}{\psi_a(1)}$. We note that $\gamma > 0$ is a constant and z_a is the a-th component of vector $\mathbf{z}^{(r)}$ to be compressed in the r-th round, $\mathbf{z}^{(1)} = \mathbf{z}$.

The parameter γ reflects the effort of the message-passing algorithm to find a codeword $\mathbf{c}_{\mathbf{m},\mathbf{x}}$ as close to $\mathbf{z}$ as possible. The larger the γ, the stronger is the effort. On the other hand, the structure of the code $\mathcal{C}$ imposes a limit on how strong this effort can be. By assigning to each source bit z_a its own parameter γ_a, we could control the probability of each source bit being preserved and thus control the probability of making an embedding change at that pixel. This should enable us to construct embedding schemes for an arbitrarily defined (e.g., non-constant) detectability measure ρ. We leave this direction to our future work.

4.4. Detailed description of SP algorithm

We now give a detailed description of the SP algorithm. As a template, we will use the pseudocode from Figure 3. It defines two procedures: the main function `SP()` and `SP_iter()` that implements the message-passing iterations.

```
procedure w = SP(G, z)
  while not all_bits_fixed(w)
    bias = SP_iter(z, G)
    bias = sort(bias)
    if max(bias)>t
      num = min(num_max, num_of_bits(bias>t))
    else
      num = num_min
    [G,z,w] = dec_most_biased_bits(G,z,w,num)
  end
end
```

```
procedure bias = SP_iter(z, G)
  M_zaa = normalize(calc_source_message(z))
  M_ai = send_src_message(G, M_zaa)
  while |M_ai_old-M_ai|<e OR iter<max_iter
    M_ai_old = M_ai
    M_ia = normalize(calc_ia(M_ai))
    M_ai = normalize(calc_ai(M_ia, M_zaa))
    if iter>start_damp then M_ai = normalize(damp(M_ai))
    iter = iter+1
  end
  bias = calc_bias(M_ai)
end
```

Figure 3. Pseudocode for the SP algorithm. This code is discussed in detail in Section 4.4.

The SP algorithm (`SP()` function) starts its first round with a bipartite graph $\mathbb{G}^{(1)}$ representing the factor graph of the linear code with generator matrix $\mathbf{G}$ and a vector of source bits $\mathbf{z}^{(1)} = \mathbf{z}$. Using these parameters, we run `SP_iter()` to calculate the bias $B_i = |\mu_i(1) - \mu_i(0)|$ for each free info bit (in the beginning, all info bits are free). The bias B_i expresses the tendency of each free info bit to be set to a specific value. In the next step, we use this information to sort the free info bits according to their bias and we select `num` most biased info bits to be set by decimation in this round. We use the following decimation strategy: set `num` to the number of free info bits with $B_i >$ `t` (constant threshold), but no more than `num_max`. If there are no $B_i >$ `t`, set `num` to some small constant `num_min`. The final step is the decimation function `dec_most_biased_bits()`. The values of the constants `num, num_max, num_min` will be discussed in Section 6.

The purpose of the decimation function is to set a given number of the most biased info bits, reduce the graph $\mathbb{G}^{(1)}$ and the vector $\mathbf{z}^{(1)}$, and obtain a new graph $\mathbb{G}^{(2)}$ and vector $\mathbf{z}^{(2)}$ for the next round. The process of graph reduction is as follows: set the `num` most biased info bits to one if $\mu_i(1) > \mu_i(0)$, otherwise set them to zero. For each info bit i and its set value w_i^{set}, do the following operation: $z_a^{(2)} = \mathrm{XOR}(z_a^{(1)}, w_i^{set})$, $\forall a \in C(i)$, where $z_a^{(2)} = z_a^{(1)}$ for each unchanged check. This operation creates an equivalent source vector for the next round. Finally, the graph $\mathbb{G}^{(2)}$ is obtained from $\mathbb{G}^{(1)}$ by removing all info bits that were set including their incident edges.

After the decimation step, we obtain a new pair of input parameters $\mathbb{G}^{(2)}$ and $\mathbf{z}^{(2)}$ prepared for the next round of the `SP_iter()` function. Applying these steps again, we obtain a smaller graph $\mathbb{G}^{(3)}$ and a new source vector $\mathbf{z}^{(3)}$. The SP algorithm ends in the r-th round when the graph $\mathbb{G}^{(r)}$ does not contain any edges (all info bits were set).

To finalize the description of the algorithm, we need to describe the `SP_iter()` function in some round r. This function takes the source vector $\mathbf{z}^{(r)}$ and graph $\mathbb{G}^{(r)}$ and returns a vector of biases for each free info bit. The core of this function is the message-passing iteration process. This process is initiated by sending messages $\mathbf{M}^{(0)}_{z_a \to a}$, defined in (16), from source bits in graph $\mathbb{G}^{(r)}$ to their checks. Checks forward these messages to their neighboring info bits and the process continues by applying the update equations from Figure 2. Each iteration consists of applying equations (13) for updating messages $\mathbf{M}^{(\ell)}_{i \to a}$ using messages $\mathbf{M}^{(\ell-1)}_{a \to i}$ from the previous iteration and applying equations (14) to obtain new $\mathbf{M}^{(\ell)}_{a \to i}$ messages from $\mathbf{M}^{(\ell)}_{i \to a}$. In (14), the constant message $\mathbf{M}^{(\ell)}_{z_a \to a} = \mathbf{M}^{(0)}_{z_a \to a}$ is used. All messages are *always normalized* so that the sum of all elements of the five-dimensional message vector is equal to 1. This is expressed using the `normalize()` pseudofunction. To speed up the iterations, after a few initial iterations (`start_damp`), the damping process is used. This process adjusts the $\mathbf{M}^{(\ell)}_{a \to i}$ messages using the the following equation: $\mathbf{M}^{(\ell)}_{a \to i} = \left(\mathbf{M}^{(\ell)}_{a \to i} \cdot \mathbf{M}^{(\ell-1)}_{a \to i}\right)^{1/2}$, where the product and square root are elementwise operations. The adjusted messages must be again normalized.

After the message-passing algorithm converged or the maximum number of iteration was reached, the biases $B_i = |\mu_i(1) - \mu_i(0)|$ are calculated for each free info bit i, where the three-dimensional vector $\big(\mu_i(0), \mu_i(1), \mu_i(*)\big)$ defined in (15) is normalized to sum to 1.

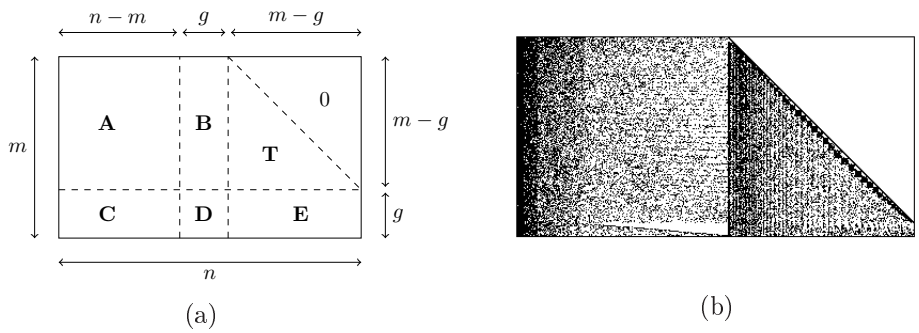


Figure 4. (a) Structure of matrix $\mathbf{G}^T$ after row and column permutation. Matrix $\mathbf{T}$ is lower triangular, $\mathbf{D}$ should be as small as possible. (b) The result after applying 'Greedy Algorithm A'.[13] The size of matrix $\mathbf{D}$ is $0.033 \cdot n$.

Stepping back for a while, the SP algorithm when run for the source vector $\mathbf{z} = \mathbf{x} - \mathbf{v_m}$ gives us the vector of info bits $\mathbf{w}$ and thus the codeword $\mathbf{c_{m,x}} = \mathbf{Gw}$ needed for embedding (9). The next section explains the last missing ingredient—how to obtain an arbitrary coset member $\mathbf{v_m}$.

5. DETERMINING THE COSET MEMBER AND CALCULATING SYNDROMES

During embedding, the sender needs to find an arbitrary member of the coset $\mathcal{C}(\mathbf{m})$ for the message $\mathbf{m}$. This requires knowledge of the parity check matrix $\mathbf{H}$. The extraction mapping (7) also needs the parity check matrix to obtain the message. The problem is that we only have the (sparse) generator matrix $\mathbf{G}$ and not $\mathbf{H}$. Finding $\mathbf{H}$ using Gaussian elimination would have cubic complexity and $\mathbf{H}$ would become dense along the process. Fortunately, since we are dealing here with a dual LDPC code, our task is in essence equivalent to *encoding* using LDPC codes for which efficient algorithms exist. In this paper, we briefly describe the approach based on partial diagonalization of sparse matrices using permutations of rows and columns.[13]

Suppose that $\mathbf{G}$ can be brought into the following form by permuting its rows and columns

$$\mathbf{G}^T = \begin{pmatrix} \mathbf{A} & \mathbf{B} & \mathbf{T} \\ \mathbf{C} & \mathbf{D} & \mathbf{E} \end{pmatrix},$$

where $\mathbf{T}$ is regular lower diagonal. Here, we hope that the square matrix $\mathbf{D}$ is relatively small. The dimensions of the matrices are shown in Figure 4. Denoting $\Phi^{-1} = (-\mathbf{E}\mathbf{T}^{-1}\mathbf{B} + \mathbf{D})^{-1}$, the matrix

$$\mathbf{H} = \left(\mathbf{I}, \Phi^{-1}(-\mathbf{E}\mathbf{T}^{-1}\mathbf{A} + \mathbf{C}), \mathbf{T}^{-1}[\mathbf{A} + \mathbf{B}\Phi^{-1}(-\mathbf{E}\mathbf{T}^{-1}\mathbf{A} + \mathbf{C})]\right)$$

is a parity check matrix of the code in *systematic form*. This can be easily seen by verifying that $\mathbf{G}^T\mathbf{H}^T = 0$. Because $\mathbf{H}$ is in systematic form, one easily find one member of the coset $\mathcal{C}(\mathbf{m})$ as $\mathbf{v_m} = (\mathbf{m}, \mathbf{0})^T$, where the zero vector has length $n - m$.

We now turn our attention to the extraction mapping and obtaining the syndrome $\mathbf{m}$ (message). According to (7), the message extraction amounts to calculating the product $\mathbf{Hy}$ for the stego object $\mathbf{y}$, which is achieved by multiplying

$$\mathbf{m} = \mathbf{y}_1 + \Phi^{-1}(-\mathbf{E}\mathbf{T}^{-1}\mathbf{A} + \mathbf{C})\mathbf{y}_2 + \mathbf{T}^{-1}[\mathbf{A} + \mathbf{B}\Phi^{-1}(-\mathbf{E}\mathbf{T}^{-1}\mathbf{A} + \mathbf{C})]\mathbf{y}_3, \tag{17}$$

where we decomposed $\mathbf{y}$ into three shorter vectors $\mathbf{y} = (\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3)$ with lengths $n-m, g$, and $m-g$, respectively. Because $\mathbf{T}$ is regular, lower-triangular, and sparse, calculating $\mathbf{T}^{-1}\mathbf{u}$ for some vector $\mathbf{u}$ can be achieved efficiently by back-substitution. Also, all matrices are sparse with the exception of Φ^{-1}. The inverse of Φ can be pre-calculated and is only a one-time cost. Moreover, Φ is $g \times g$, where g is small, and the multiplication by Φ^{-1} has a low complexity proportional to g^2. Thus, the two multiplications in (17) have the following complexity $O(n + g^2)$ and $O(n)$.

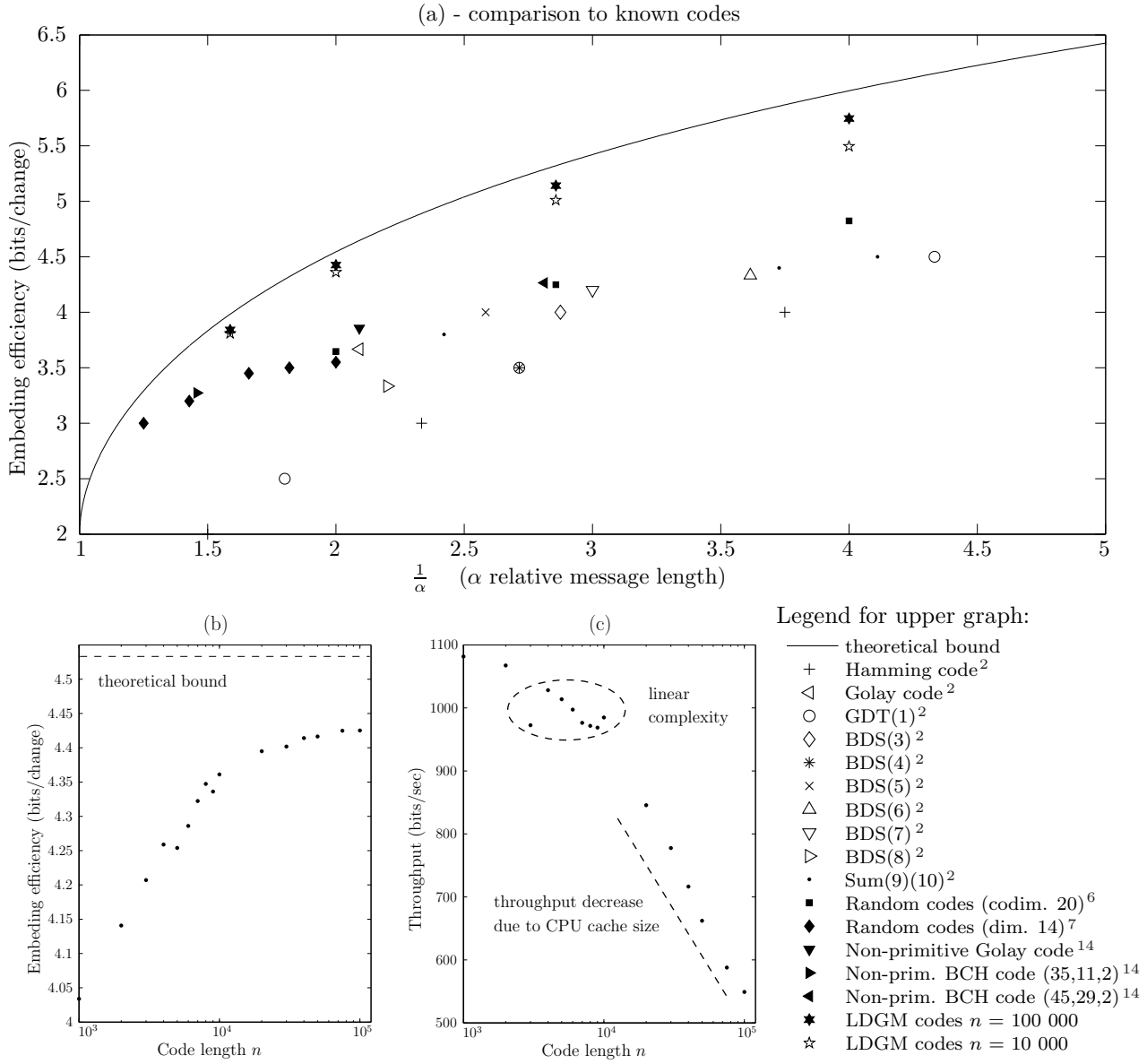


Figure 5. Results of running the SP algorithm for Matrix Embedding. (b) and (c) are for relative message length $\frac{1}{2}$.

For example, for $(3, 6)$ regular LDPC codes, the average complexity for calculating the syndrome (extracting the message) is $0.017^2 n^2 + O(n)$, where n is the code length.

In practice, we ran the 'Greedy Algorithm A' many times (each time for a slightly different value of the parameter α, see[13] for details) and found a row and column permutation that gave us the smallest size of the matrix $\mathbf{D}$. The output of the greedy algorithm is our generator matrix $\mathbf{G}$ that is used in the binary quantizer. An example of the permuted matrix is in Figure 4(b).

6. RESULTS

We have implemented the whole framework for the special case of a constant detectability measure $\rho_i = const.$ (Matrix Embedding) to be able to compare the results with previously published practical codes.

The SP algorithm was implemented in C++ using Intel C++ 9.0 compiler. Each update equation was manually optimized for using float data type with SSE instructions. The following results were obtained using an Intel Core2 X6800 2.93GHz CPU machine with 64bit Linux, where both CPU cores were utilized.

We now give the exact values for each parameter mentioned in Section 4.4. The following parameters do not depend on the code or the message length: `start_damp`=4, `t`=0.8, w_{sou}=1.1, and w_{info}=1. The number of iterations can be controlled using the parameter `e` (we used `e`=0.001) or by limiting the maximal number of iterations. Limiting the number of iterations `max_iter` to 40–100 produced very similar results while providing a speed up linear in `max_iter`. We have experimentally determined the parameter γ to maximize the performance for each relative message length α, for example, $\gamma_{0.63} = 0.94$, $\gamma_{0.5} = 1.13$, $\gamma_{0.35} = 1.37$, $\gamma_{0.25} = 1.65$. The parameters `num_max` and `num_min` were set to 1% and 0.1% of the total number of pixels in the cover object. These values *did not change* during runtime. We have also experimented with a general decimation strategy defined as the percentage of the total number of free bits while setting `num_min` as 0.1× `num_max`. The algorithm can be sped up by enlarging `num_max` at the cost of losing some embedding efficiency. For practical usage, this trade off should be considered and further explored.

We evaluate the performance of the codes by their embedding efficiency. Figure 5(a) shows the comparison with other previously proposed codes. Our results are labeled as 'LDGM codes' and each embedding efficiency was obtained by averaging over 20 randomly generated messages. For each relative message length, we ran the SP algorithm for two different code lengths $n = 10000$ and $n = 100000$. The codes labeled as 'random codes' are obtained from[6] and.[7] The remaining codes were taken from[2] and consist primarily of block-wise direct sum (BDS) of non-linear factor codes constructed using Preparata codes.

The generator matrix of each code $\mathbf{G}$ was generated randomly, where the number of ones in each row and column was given by a specific (degree) probability distribution. We have tested degree distributions optimized for ordinary message-passing over the BSC and the binary erasure channel (BEC). While codes for both channels gave satisfactory results, degree distributions optimized for the BSC provided higher embedding efficiency. Thus, all results reported here are for codes with matrices optimized for the BSC channel.

As an example, in Figures 5(b) and (c), we show the performance and speed for codes obtained for the following degree distribution

$$\begin{aligned}\lambda(x) = \quad & 0.44676014278323x + 0.2936700938561x^2 + 0.085704194057476x^5 + 0.0819921690616x^6 + \\ & +0.0046931938076233^{12} + 0.017115184041391x^{13} + 0.038637012348276x^{14} + 0.0314280100443x^{39} \\ \rho(x) = x^9.\end{aligned}$$

Figure 5(b) shows how the proposed embedding algorithm with random codes based on this distribution approaches the upper bound on embedding efficiency as the code length grows (for relative message length $\alpha = \frac{1}{2}$). The final gap in embedding efficiency between the theoretical bound and codes based on the code length $n = 10^5$ is less than 0.1 bits per change.

To illustrate the computational complexity of our implementation, we define *throughput* as a number of embedded bits per second and plot this variable for different code lengths and $\alpha = \frac{1}{2}$ in Figure 5(c). The complexity grows *linearly* with the code length. The decrease in throughput for code lengths larger than 10^4 is caused by the limited size of the CPU cache and not by the nature of the SP algorithm.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we present a general approach for minimizing embedding impact in steganography. We start by defining the detectability measure at each element of the cover object and derive theoretical bounds on the minimal embedding impact needed to embed a given payload. Then, we study embedding schemes realized using syndrome codes and show that the problem of minimizing embedding impact is equivalent to binary quantization using appropriately weighted distance. We use the binary quantizer based on low density generator matrices and the survey propagation algorithm recently proposed by Wainwrigth and Maneva.[16] The performance of the proposed embedding algorithm is compared to previous art on Matrix Embedding where the embedding impact is the same for all elements of the cover object. The algorithm performance achieves near-optimal embedding efficiency (within 0.1 bits per change) with the speed of embedding at 1000 bits per second. Another important

advantage of the proposed approach over structured codes is that it provides an essentially continuous family of codes for arbitrarily chosen relative message length α.

We postpone practical application of this framework to non-constant embedding impact profiles to our future work. We believe that the proposed framework will provide near-optimal performance in this case as well after appropriately adjusting the parameter γ for each element of the cover. Our results were obtained using degree distributions optimized for the BSC channel, but it is not clear whether these distributions are optimal in any sense. Optimization of the degree distribution is another interesting future direction we plan to pursue.

APPENDIX A.

In this appendix, we derive the expression for the minimal embedding impact for any steganographic scheme that communicates m bits in n pixels with detectability measure $\rho_i, i = 1, \dots, n$. We do so for the more general case when the embedding impact is an arbitrary (i.e., not necessarily additive) function of the detectability measure ρ_i. For $\mathbf{x}, \mathbf{y} \in \{0,1\}^n$, we define the modification pattern $\mathbf{s} \in \{0,1\}^n$ as $s_i = \delta(x_i, y_i)$, where $\delta(a,b) = 1$ when $a = b$ and $\delta(a,b) = 0$, otherwise. Furthermore, we define $D(\mathbf{s}) = D(\mathbf{x}, \mathbf{y})$ as the embedding impact of making embedding changes at pixels with $s_i = 1$. Let us assume that the recipient also knows the cover $\mathbf{x}$. By the Gelfand-Pinsker theorem,[9] the conclusions reached here do not depend on this assumption. The sender then basically communicates the modification pattern $\mathbf{s}$. Assuming the sender selects each pattern $\mathbf{s}$ with probability $p(\mathbf{s})$, the amount of information that can be communicated is the entropy of $p(\mathbf{s})$

$$H(p) = -\sum_{\mathbf{s}} p(\mathbf{s}) \log_2 p(\mathbf{s}).$$

Our problem is now reduced to finding the probability distribution $p(\mathbf{s})$ on the space of all possible flipping patterns $\mathbf{s}$ that minimizes the expected value of the embedding impact

$$\sum_{\mathbf{s}} D(\mathbf{s}) p(\mathbf{s})$$

subject to the constraints

$$H(p) = \sum_{\mathbf{s}} p(\mathbf{s}) \log_2 p(\mathbf{s}) = m, \sum_{\mathbf{s}} p(\mathbf{s}) = 1,$$

This problem can be solved using Lagrange multipliers. Let

$$F(p(\mathbf{s})) = \sum_{\mathbf{s}} p(\mathbf{s}) D(\mathbf{s}) + \mu_1 \left(m - \sum_{\mathbf{s}} p(\mathbf{s}) \log_2 p(\mathbf{s}) \right) + \mu_2 \left(\sum_{\mathbf{s}} p(\mathbf{s}) - 1 \right).$$

Then,

$$\frac{\partial F}{\partial p(\mathbf{s})} = D(\mathbf{s}) - \mu_1 (\log_2 p(\mathbf{s}) + 1/\ln(2)) + \mu_2 = 0$$

if and only if $p(\mathbf{s}) = A e^{-\lambda D(\mathbf{s})}$, where $A^{-1} = \sum_{\mathbf{s}} e^{-\lambda D(\mathbf{s})}$ and λ is determined from

$$-\sum_{\mathbf{s}} p(\mathbf{s}) \log_2 p(\mathbf{s}) = m.$$

Thus, the probabilities $p(\mathbf{s})$ follow an exponential distribution with respect to the embedding impact $D(\mathbf{s})$.

If the embedding impact of the pattern $\mathbf{s}$ is an additive function of "singleton" patterns (patterns for which only one pixel is modified), then $D(\mathbf{s}) = s_1 \rho_1 + \cdots + s_n \rho_n$ and $p(\mathbf{s})$ accepts the form

$$p(\mathbf{s}) = A e^{-\lambda \sum_{i=1}^n s_i \rho_i} = A \prod_{i=1}^n e^{-\lambda s_i \rho_i}, A^{-1} = \sum_{\mathbf{s}} \prod_{i=1}^n e^{-\lambda s_i \rho_i} = \prod_{i=1}^n (1 + e^{-\lambda \rho_i}),$$

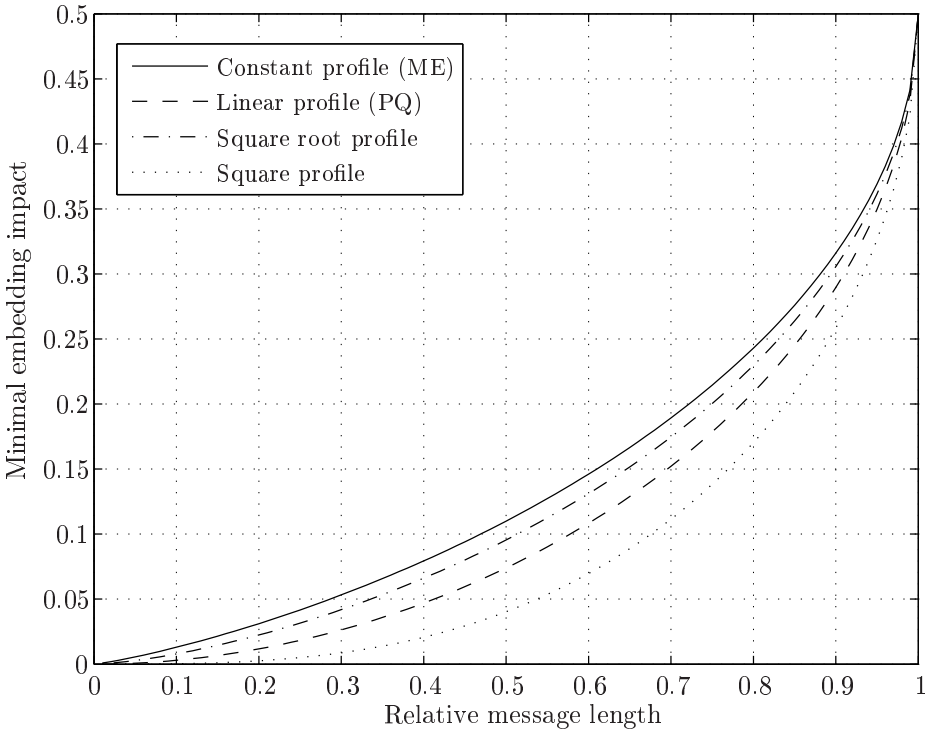


Figure 6. Minimal embedding impact vs. relative message length for four detectability profiles ρ.

which further implies

$$p(\mathbf{s}) = \prod_{i=1}^{n} p_i(s_i),$$

where $p_i(1)$ and $p_i(0)$ are the probabilities that the i-th pixel is (is not) modified during embedding

$$p_i(0) = \frac{1}{1+e^{-\lambda\rho_i}}, p_i(1) = \frac{e^{-\lambda\rho_i}}{1+e^{-\lambda\rho_i}}.$$

Of course, this further implies that the joint probability distribution $p(\mathbf{s})$ can be factorized and thus we only need to know the marginal probabilities p_i that the i-th pixel is modified. It also enables us to write for the entropy

$$H(p) = \sum_{i=1}^{n} H(p_i),$$

where in the sum the function H applied to a scalar is the binary entropy function.

Note that when $\rho_i = 1, \forall i$, we obtain

$$m = \sum_{i=1}^{n} H(p_i) = \sum_{i=1}^{n} H\left(\frac{e^{-\lambda}}{1+e^{-\lambda}}\right),\ E\left(\sum_{i=1}^{n} p_i\rho_i\right) = \frac{ne^{-\lambda}}{1+e^{-\lambda}}.$$

Thus, in agreement with the result derived in[1] we obtain the following relationship between the embedding impact per pixel d/n and the relative message length m/n

$$\frac{d}{n} = H^{-1}\left(\frac{m}{n}\right).$$

Let us sort ρ_i from the smallest to the largest and normalize so that $\sum_i \rho_i = 1$. Let ρ be a Riemann-integrable non-decreasing function on $[0,1]$ such that $\rho(i/n) = \rho_i$. Then for $n \to \infty$, the average distortion per element $d = D/n = \frac{1}{n}\sum_{i=1}^{n} p_i\rho_i \to \int_0^1 p(x)\rho(x)dx$, where $p(x) = \frac{e^{-\lambda\rho(x)}}{1+e^{-\lambda\rho(x)}}$. By the same token, $\alpha = m/n = \frac{1}{n}\sum_{i=1}^{n} H(p_i) \to \int_0^1 H(p(x))dx$. By direct calculation

$$\ln 2 \times \int_0^1 H(p(x))dx = \lambda \int_0^1 \frac{\rho(x)e^{-\lambda\rho(x)}}{1+e^{-\lambda\rho(x)}}dx + \int_0^1 \ln(1+e^{-\lambda\rho(x)})dx = \lambda \int_0^1 \frac{(\rho(x)+x\rho'(x))e^{-\lambda\rho(x)}}{1+e^{-\lambda\rho(x)}}dx + \ln(1+e^{-\lambda\rho(1)}).$$

The second equality is obtained by integrating the second integral by parts. Thus, we can obtain the embedding capacity-distortion relationship in a parametric form

$$\begin{aligned} d(\lambda) &= G_\rho(\lambda) \\ \alpha(\lambda) &= \frac{1}{\ln 2}\left(\lambda F_\rho(\lambda) + \ln(1+e^{-\lambda\rho(1)})\right), \end{aligned}$$

where λ is a non-negative parameter and

$$\begin{aligned} G_\rho(\lambda) &= \int_0^1 \frac{\rho(x)e^{-\lambda\rho(x)}}{1+e^{-\lambda\rho(x)}}dx \\ F_\rho(\lambda) &= \int_0^1 \frac{(\rho(x)+x\rho'(x))e^{-\lambda\rho(x)}}{1+e^{-\lambda\rho(x)}}dx. \end{aligned}$$

Acknowledgements

The work on this paper was supported by Air Force Research Laboratory, Air Force Material Command, USAF, under the research grant FA8750-04-1-0112 and AFOSR grant number FA9550-06-1-0046. 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 Air Force Research Laboratory, or the U.S. Government. Special thanks belong to Martin Wainwright for fruitful discussions and Emin Martinian for directing our attention to quantizers based on LDGM codes.

REFERENCES

1. J. Bierbrauer. On Crandall's problem. *Personal Communication, (available from http://www.ws.binghamton.edu/fridrich/covcodes.pdf)*, 1998.
2. J. Bierbrauer and J. Fridrich. Constructing good covering codes for applications in Steganography. *In preparation, preprint available from http://www.ws.binghamton.edu/fridrich/stegocovsurveyOct06.pdf)*, 2006.
3. C. Cachin. An information-theoretic model for steganography. In D. Aucsmith, editor, *Information Hiding, 2nd International Workshop*, volume 1525 of *LNCS*, pages 306–318. Springer-Verlag, New York, 1998.
4. R. Crandall. Some notes on steganography. *Available from http://os.inf.tu-dresden.de/∼westfeld/crandall.pdf*, 1998.
5. J. Fridrich, M. Goljan, and D. Soukal. Perturbed quantization steganography. *ACM Multimedia and Security Journal*, 11(2):98–107, 2005.
6. J. Fridrich, M. Goljan, and D. Soukal. Wet paper codes with improved embedding efficiency. *IEEE Transactions on Information Security and Forensics*, 1(1):102–110, 2006.
7. J. Fridrich and D. Soukal. Matrix embedding for large payloads. *IEEE Transactions on Information Security and Forensics*, 1(3):390–394, 2006.
8. F. Galand and G. Kabatiansky. Information hiding by coverings. In *Proceedings ITW2003, Paris, France, 2003*, pages 151–154.
9. S. I. Gel'fand and M. S. Pinsker. Coding for channel with random parameters. *Probl. Pered. Inform. (Probl. Inform. Trans.)*, 9(1):19–31, 1980.

10. E. Maneva, E. Mossel, and M. J. Wainwright. A new look at survey propagation and its generalizations. In *Proceedings of the 16th Annual Symposium on Discrete Mathematics (SODA)*, pages 1089–1098, 2005.
11. E. Martinian and M. J. Wainwright. Analysis of LDGM and compound codes for lossy compression and binning. In *Workshop on Information Theory and its Applications, San Diego*, February 2006.
12. T. Richardson, A. Shokrollahi, and R. Urbanke. Design of capacity-approaching irregular low-density parity check codes. *IEEE Transactions on Information Theory*, 47:619–637, February 2001.
13. T. Richardson and R. Urbanke. Efficient encoding of low-density parity check codes. *IEEE Transactions on Information Theory*, 47(2):638–656, February 2001.
14. A. Schneidewind and D. Schönfeld. Embedding with syndrome coding based on BCH codes. In J. Dittman and J. Fridrich, editors, *Proceedings ACM Multimedia and Security Workshop, Geneva, Switzerland, September 26–27, 2006*, pages 214–223. ACM Press, New York.
15. M. van Dijik and F. Willems. Embedding information in grayscale images. In *Proceedings of the 22nd Symposium on Information and Communication Theory in the Benelux, Enschede, The Netherlands, May 15–16, 2001*, pages 147–154.
16. M. J. Wainwright and E. Maneva. Lossy source encoding via message-passing and decimation over generalized codewords of LDGM codes. In *Proceedings of the International Symposium on Information Theory, Adelaide, Australia*, September 2005.
17. F. J. M. Williams and N. J. Sloane. *The Theory of Error-Correcting Codes*. North-Holland, Amsterdam, 1977.