Reducing Invertible Embedding Distortion Using Graph Matching Model

Hanzhou Wu†,‡ ***and Xinpeng Zhang***†,‡
†***School of Communication and Information Engineering, Shanghai University, Shanghai 200444, China***
‡***Shanghai Institute for Advanced Communication and Data Science, Shanghai 200444, China***
Email: h.wu.phd@ieee.org (Corresponding author), xzhang@shu.edu.cn

Abstract

Invertible embedding allows the original cover and embedded data to be perfectly reconstructed. Conventional methods use a well-designed predictor and fully exploit the carrier characteristics. Due to the diversity, it is actually hard to accurately model arbitrary covers, which limits the practical use of methods relying heavily on content characteristics. It has motivated us to revisit invertible embedding operations and propose a general graph matching model to generalize them and further reduce the embedding distortion. In the model, the rate-distortion optimization task of invertible embedding is derived as a weighted bipartite graph matching problem. In the bipartite graph, the nodes represent the values of cover elements, and the edges indicate the candidate modifications. Each edge is associated with a weight indicating the corresponding embedding distortion for the connected nodes. By solving the minimum weight maximum matching problem, we can find the optimal embedding strategy under the constraint. Since the proposed work is a general model, it can be incorporated into existing works to improve their performance, or used for designing new invertible embedding systems. We incorporate the proposed work into a part of state-of-the-arts, and experiments show that it significantly improves the rate-distortion performance. To the best knowledge of the authors, it is probably the first work studying rate-distortion optimization of invertible embedding from the perspective of graph matching model.

Introduction

Information hiding is an emerging and interdisciplinary field involving different applications, among which digital watermarking and steganography [1] are the most popular. In particular, the ease of tampering and distribution of digital products has led digital watermarking to a major activity in digital media processing. In terms of embedding strategy, information hiding could be organized into two categories, i.e., *non-invertible embedding* and *invertible embedding*. Though both modify the cover without introducing significant distortion, the former permanently distorts the cover whereas the latter allows the original cover to be restored after the embedded data was extracted.

Invertible embedding, also called reversible data hiding [2], reversible watermarking [3], enables media objects to be authenticated and restored to their original versions, making them quite suitable for sensitive purposes, e.g., the US army is interested in such kind of technology for authentication of reconnaissance images[1]. We investigate fragile invertible embedding, meaning that, when the marked object was tampered, one will find it is not authentic. Invertible embedding approaches can be evaluated by the rate-distortion behavior. Namely, for a payload, it is required to reduce the distortion as much as possible. In other words, we hope to embed as many message bits as possible for a fixed distortion.

A number of invertible embedding algorithms have been developed in the past years such as lossless compression (LC) [4], difference expansion (DE) [5], and histogram shifting (HS) [2]. It could be said to a certain extent that, to achieve superior rate-distortion performance, current state-of-the-arts are based on prediction errors (PEs) and HS (or its variants), e.g., [3], [6], [7], [8]. Though LC and DE intuitively are different from HS during data embedding, both can be considered as a variant of HS. In detail, after the cover elements were losslessly compressed, the data embedding procedure in the reserved (empty) space can be considered as shifting between "0" and "1". And, the DE operation can be considered as "jump" between different values, e.g., d can be jumped (shifted) to either $2d$ or $2d+1$ to carry a message bit. Therefore, by building analytic models for HS, we may be able to extend them to LC and DE. And, it can be inferred that, the HS operation is actually not unique, meaning that, from the viewpoint of rate-distortion optimization, we are expecting to find the optimal HS operation, which is the main interest of this paper.

Recalling the key steps of HS based approaches, we first use a predictor to predict the cover elements and then construct a PE sequence (PES), whose generation involves a local complexity evaluation function allowing us to preferentially use smooth elements for providing better performance. Thereafter, with the PES, a PE histogram (PEH) can be constructed. As a pooled vector of the noise-like PEs, the PEH enables us to effectively embed a secret payload with low distortion as long as the HS parameters are well chosen. Many conventional works use a fixed HS operation, which can be roughly descried as follows. The central PEH bins (if any) are unchanged. The two-side border PEH bins are shifted along the two side-directions to reserve empty positions, and the rest bins are shifted along the corresponding direction to carry the secret bits. For single-layer embedding, one could easily find the optimal HS operation that minimizes the introduced distortion. For multi-layer embedding, using the empirical HS operation will be a good choice, which, however, may be not optimal in terms of rate-distortion optimization. Heuristically tuning the HS parameters by a non-deterministic manner may achieve better performance, but it is not desirable for reproducibility of experiments. It therefore motivates us to introduce a deterministic rate-distortion optimization algorithm for HS embedding. We point that, both LC and DE can be generalized by proposed work.

[1]`https://en.wikipedia.org/wiki/Digital_watermarking`

Specifically, we model HS in a bipartite graph. The vertices in the bipartite graph correspond to the PEH bins, and edges show the shifting relationship between vertices. All edges are assigned with a weight to specify the corresponding shifting distortion. We find the minimum weight maximum matching (MWMM) in the bipartite graph such that the optimal HS parameters can be obtained, which minimizes the distortion under given constraint. We incorporate the proposed work into a part of state-of-the-arts, and experiments show that it can significantly improve the performance. The structure of this paper is organized as follows. The problem is first formulated in the next section. Then, we present the proposed graph matching model. The implementation issues and solutions are thereafter provided, followed by experimental results and analysis. Finally, we conclude this paper and provide further research directions.

Problem Formulation

We use image as the cover. Let $\mathbf{x}^{(t)}$, $t \geq 0$, be the image embedded with t times. Obviously, $\mathbf{x}^{(0)}$ is the original image without hidden bits. Let $\mathbf{x}^{(t)} = ({x_1}^{(t)}, {x_2}^{(t)}, ..., {x_n}^{(t)}) \in X = I^n$ be an n-pixel cover image with pixel range I, e.g., $I = \{0, 1, ..., 255\}$ for 8-bit grayscale image. For a payload, we use $\mathbf{x}^{(0)}$ and $\mathbf{x}^{(t)}$ to generate $\mathbf{x}^{(t+1)} (t \geq 0)$ with HS operation. We expect to minimize

$$D(\mathbf{x}^{(0)}, \mathbf{x}^{(t+1)}) = \sum_{i=1}^{n} \rho_i(\mathbf{x}^{(0)}, {x_i}^{(t+1)}), \tag{1}$$

where $\rho_i : X \times I \mapsto R$ exposes the cost of changing ${x_i}^{(0)}$ to ${x_i}^{(t+1)}$. For invertible embedding, we often use squared error to evaluate the distortion. In default, we use mean squared error (MSE), i.e.,

$$D(\mathbf{x}^{(0)}, \mathbf{x}^{(t+1)}) = \frac{1}{n} \cdot \sum_{i=1}^{n} ({x_i}^{(0)} - {x_i}^{(t+1)})^2. \tag{2}$$

We predict elements in $\mathbf{x}^{(t)}$, and generate a PEH. Let $h(v)$ be the occurrence of the PEH bin with value v, where $|*|$ is the size of a set. We use HS for data embedding. Mathematically, let A and B denote a set including all PEH bins and that contains all non-zero occurrence bins, i.e., $A = \{v | v \in (-|I|, |I|)\}$, $B = \{v | h(v) > 0\} \subset A$. For a peak-bin set $P = \{p_1, p_2, ..., p_m\} \subset B$, we first find such two injective functions g_0 and g_1 that, $G_0 = \{g_0(p_1), g_0(p_2), ..., g_0(p_m)\} \subset A$ and $G_1 = \{g_1(p_1), g_1(p_2), ..., g_1(p_m)\} \subset A$, where $G_0 \cap G_1 = \emptyset$. Another injective function $f : B \setminus P \mapsto A \setminus (G_0 \cup G_1)$ is also required. Suppose the bit-size of message is no more than $\sum_{i=1}^{m} h(p_i)$, for data embedding, according to f, we first shift all PEH bins in $B \setminus P$ into some bin-positions of $A \setminus (G_0 \cup G_1)$. Thereafter, since the bin-positions of $(G_0 \cup G_1) \setminus P$ are empty (i.e., with zero occurrence), one can easily embed the secret bits by shifting the bins in P into the bin-positions of $(G_0 \cup G_1)$.

Let $\mathbf{c}^{(t)} = ({c_1}^{(t)}, {c_2}^{(t)}, ..., {c_{n_t}}^{(t)}), n_t \leq n$, represent all the cover pixels to be embedded. For compactness, we sometimes consider $\mathbf{c}^{(t)}$ and ${c_i}^{(t)}$ as a set including all the pixels to be embedded and the i-th pixel with a value of ${c_i}^{(t)}$. Similarly, we denote the prediction of $\mathbf{c}^{(t)}$ and its marked version by $\mathbf{z}^{(t)} = ({z_1}^{(t)}, {z_2}^{(t)}, ..., {z_{n_t}}^{(t)})$ and $\mathbf{s}^{(t)} = ({s_1}^{(t)}, {s_2}^{(t)}, ..., {s_{n_t}}^{(t)})$. The PEs $\mathbf{e}^{(t)} = ({e_1}^{(t)}, {e_2}^{(t)}, ..., {e_{n_t}}^{(t)})$ between $\mathbf{c}^{(t)}$ and $\mathbf{z}^{(t)}$ is therefore denoted by

$${e_i}^{(t)} = {c_i}^{(t)} - {z_i}^{(t)}, (1 \leq i \leq n_t). \tag{3}$$

The relationship of $\mathbf{c}^{(t)}$ and $\mathbf{s}^{(t)}$ can be found from:

$$\begin{aligned} {s_i}^{(t)} &= {z_i}^{(t)} + \hat{e}_i^{(t)} \\ &= \begin{cases} {z_i}^{(t)} + g_{b_k}({e_i}^{(t)}), & \text{if } {e_i}^{(t)} \in P; \\ {z_i}^{(t)} + f({e_i}^{(t)}), & \text{if } {e_i}^{(t)} \in B \setminus P. \end{cases} \end{aligned} \tag{4}$$

Here, $b_k \in \{0, 1\}$ is the k-th (present) bit to be embedded.

We use $\mathbf{o}^{(t)} = ({o_1}^{(t)}, {o_2}^{(t)}, ..., {o_{n_t}}^{(t)})$ to denote the original values of $\mathbf{c}^{(t)}$ in $\mathbf{x}^{(0)}$. For the pixels not in $\mathbf{c}^{(t)}$, the distortion can be roughly considered as fixed since we will not embed data into these pixels (though we may empty some LSBs to store the secret key). To generate $\mathbf{x}^{(t+1)}$, our task is

$$D(\mathbf{x}^{(0)}, \mathbf{x}^{(t+1)}) = \min_{P, g_0, g_1, f} \frac{1}{n} \cdot \sum_{i=1}^{n_t} ({s_i}^{(t)} - {o_i}^{(t)})^2 + E_c, \tag{5}$$

where E_c is a constant (that can be ignored for optimization) and

$$\begin{aligned} \sum_{i=1}^{n_t} ({s_i}^{(t)} - {o_i}^{(t)})^2 = &\sum_{{e_i}^{(t)} \in P} (g_{b_k}({e_i}^{(t)}) + {z_i}^{(t)} - {o_i}^{(t)})^2 \\ &+ \sum_{{e_i}^{(t)} \in B \setminus P} (f({e_i}^{(t)}) + {z_i}^{(t)} - {o_i}^{(t)})^2. \end{aligned} \tag{6}$$

Obviously, for a fixed P, it is required to determine such optimal (g_0, g_1, f) that $\sum_{i=1}^{n_t} ({s_i}^{(t)} - {o_i}^{(t)})^2$ is minimized. If this subproblem is solved, one can enumerate P to find the globally optimal (P, g_0, g_1, f) since $|P|$ is often small. We study along this direction. With optimal (P, g_0, g_1, f), one can embed data in $\mathbf{x}^{(t)}$.

Graph Matching Model

An important task is to find optimal (g_0, g_1, f) for a fixed P so that the corresponding distortion could be minimized. Let F denote the resultant set by applying f to $B \setminus P$. Figure 1 shows the relationship between the three injective functions. It is required that, $|P| = |G_0| = |G_1|$, $|B \setminus P| = |F|$, $G_0 \cap G_1 = \emptyset$, $G_0 \cap F = \emptyset$, and $G_1 \cap F = \emptyset$. Notice that, $B \subset A$. If any $v \in B$ is mapped to $u \in A$ by f (or g_0, g_1), we say u (or v) is matched by v (or u) according to f. Therefore, a total of $|B| + |P|$ elements in A will be matched by exactly one element in B according to f, g_0 or g_1. It is possible that $u = v$, e.g., $f(v) = v$. Also, there are $|P|$ triples (u, v, w) such that $u < w$ and $g_b^{-1}(u) = g_{1-b}^{-1}(w) = v, b \in \{0, 1\}$.

According to Eqs. (5, 6), our optimization task is:

$$\begin{aligned} L(\mathbf{x}^{(0)}, \mathbf{x}^{(t+1)}; P) &= \min_{g_0, g_1, f} \sum_{i=1}^{n_t} ({s_i}^{(t)} - {o_i}^{(t)})^2 \\ &= \min_{g_0, g_1, f} \sum_{{e_i}^{(t)} \in P} (g_{b_k}({e_i}^{(t)}) + {z_i}^{(t)} - {o_i}^{(t)})^2 \\ &\quad + \sum_{{e_i}^{(t)} \in B \setminus P} (f({e_i}^{(t)}) + {z_i}^{(t)} - {o_i}^{(t)})^2. \end{aligned} \tag{7}$$

We address this problem by modelling it in a weighted bipartite graph. A bipartite graph (typically also called bigraph), is a graph $G(V, E)$ whose vertex-set V can be divided into such two subsets V_1 and V_2 that all edges (in E) connect a vertex in V_1 and a vertex in V_2. It is required that, $V_1 \cup V_2 = V$ and $V_1 \cap V_2 = \emptyset$. If all the edges are associated with a weight, it is called a weighted bipartite graph (or called weighted bigraph).

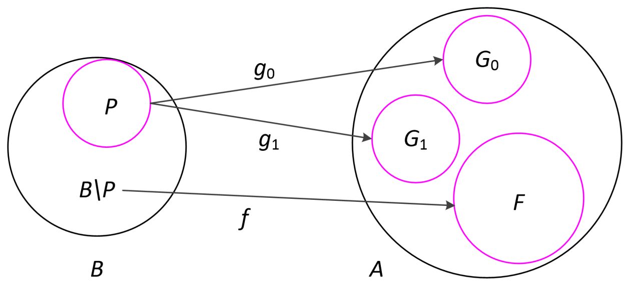

Figure 1. *The relationship between three injective functions g_0, g_1 and f.*

The secret bits can be orderly embedded into $\mathbf{c}^{(t)}$. When P, g_0 and g_1 are all fixed, with the secret message, one can consider $\sum_{e_i^{(t)}\in P}(g_{b_k}(e_i^{(t)})+z_i^{(t)}-o_i^{(t)})^2$ as constant. Then,

$$L(\mathbf{x}^{(0)},\mathbf{x}^{(t+1)};P,g_0,g_1) = \min_f \sum_{e_i^{(t)}\in B\setminus P}(f(e_i^{(t)})+z_i^{(t)}-o_i^{(t)})^2. \quad (8)$$

Below, we first model Eq. (8) in a weighted bipartite graph to find optimal f, where P,g_0,g_1 are fixed. Then, we find optimal (f,g_0,g_1) in Eq. (7) with a newly built weighted bipartite graph.

Model Derivation

Without the loss of generality, we rewrite f as:

$$f(x) = x + \triangle_f(x), \quad (9)$$

where $\triangle_f(x)$ has no need to be injective. Therefore,

$$L(\mathbf{x}^{(0)},\mathbf{x}^{(t+1)};P,g_0,g_1) = \min_f \sum_{e_i^{(t)}\in B\setminus P}(\triangle_f(e_i^{(t)})+c_i^{(t)}-o_i^{(t)})^2. \quad (10)$$

To avoid underflow/overflow problem, we should adjust the boundary pixels to reliable range in advance. The preprocessed pixels are recorded to constitute a location map, which is self-embedded. Thus, $\triangle_f(x)$ should be bounded:

$$-T \leq \triangle_f(x) \leq T,$$

where T is a positive integer, e.g., $T=1$. $g_0(x)$ and $g_1(x)$ should be bounded as well. For simplicity, we always have $|x-g_0(x)|\leq T$ and $|x-g_1(x)|\leq T$.

Let $\{y_1,y_2,...,y_{|B\setminus P|}\}$ denote all elements in $B\setminus P$. Eq. (10) can be therefore rewritten as:

$$L(\mathbf{x}^{(0)},\mathbf{x}^{(t+1)};P,g_0,g_1) = \min_f \sum_{j=1}^{|B\setminus P|} J(y_j;f), \quad (11)$$

where

$$\begin{aligned} J(y_j;f) &= \sum_{e_i^{(t)}=y_j}(\triangle_f(y_j)+c_i^{(t)}-o_i^{(t)})^2 \\ &= \sum_{e_i^{(t)}=y_j}\sum_{k=-T}^{T}\delta(\triangle_f(y_j),k)\cdot(k+c_i^{(t)}-o_i^{(t)})^2 \\ &= \sum_{k=-T}^{T}\delta(\triangle_f(y_j),k)\sum_{e_i^{(t)}=y_j}(k+c_i^{(t)}-o_i^{(t)})^2 \\ &= \sum_{k=-T}^{T}\delta(f(y_j)-y_j,k)\cdot C_k(y_j), \end{aligned}$$

where $\delta(x,y)=1$ if $x=y$, otherwise $\delta(x,y)=0$; and,

$$C_k(y_j) = \sum_{e_i^{(t)}=y_j}(k+c_i^{(t)}-o_i^{(t)})^2. \quad (12)$$

Weighted Bipartite Graph Matching

Each element in $B\setminus P$ should be matched by exactly one element in $A\setminus(G_0\cup G_1)$ according to f. Notice that, we often have $|B\setminus P|\leq|A\setminus(G_0\cup G_1)|$. We expect to find such optimal f_{opt} that Eq. (11) is minimized. Namely,

$$f_{\text{opt}} = \arg\min_f \sum_{j=1}^{|B\setminus P|}\sum_{k=-T}^{T}\delta(f(y_j)-y_j,k)\cdot C_k(y_j), \quad (13)$$

subject to

$$\sum_{k=-T}^{T}\delta(f(y_j)-y_j,k)=1, \text{ for all } 1\leq j\leq|B\setminus P|. \quad (14)$$

With Eq. (12), all possible $C_k(y_j)$ can be determined in advance. We use $\{q_1,\ q_2,\ ...,\ q_{|A\setminus(G_0\cup G_1)|}\}$ to represent elements in $A\setminus(G_0\cup G_1)$. We model Eq. (13) in a weighted bipartite graph. To build the bipartite graph, we first denote two disjoint sets by $V_1=B\setminus P$ and $V_2=A\setminus(G_0\cup G_1)$. With Eq. (14), for every possible (i,j), if $|y_j-q_i|\leq T$, we assign an edge between y_j and q_i in the bipartite graph. It indicates that, it is possible that $f_{\text{opt}}(y_j)=q_i$. Meanwhile, all edges will be assigned with the corresponding weights. Specifically, if there exists an edge between y_j and q_i, the assigned weight should be $C_{q_i-y_j}(y_j)$, meaning that, if $f_{\text{opt}}(y_j)=q_i$, the corresponding distortion is $C_{q_i-y_j}(y_j)$.

In a bipartite graph, there may be many candidates for maximum matching. A maximum matching ensures that, there have no two edges sharing the same vertex, and the number of edges in the matching is maximum. For any maximum matching $M_{\max}$ of a bipartite graph, there must be $|M_{\max}|\leq\min\{|V_1|,|V_2|\}$. As f_{opt} is injective, we can infer that, f_{opt} corresponds to such a matching M_{opt} that $|M_{\text{opt}}|=|V_1|$, where M_{opt} is also a maximum matching since $|M_{\text{opt}}|=|V_1|\geq\min\{|V_1|,|V_2|\}\geq|M_{\max}|$, i.e.,

Property 1. f_{opt} corresponds to such a maximum matching M_{opt} that $|M_{\text{opt}}|=|V_1|$, where $V_1=B\setminus P$.

Moreover, it can be seen from Eq. (13) that, f_{opt} requires that, the sum of edge-weights in M_{opt} should be the minimum. Therefore, to find f_{opt}, we have to determine

$$M_{\text{opt}} = \arg\min_{|M|=|V_1|}\sum_{(y_j,q_i)\in M,y_j\in V_1,q_i\in V_2} C_{q_i-y_j}(y_j). \quad (15)$$

Namely,

Property 2. M_{opt} has the minimum sum of weights.

In graph theory, for a weighted bipartite graph, a minimum weight maximum matching (MWMM) is defined as a maximum matching where the sum of weights associated to the edges in the matching is minimum. We can determine M_{opt} in the weighted bipartite graph with Hungarian algorithm, and then easily construct f_{opt} with M_{opt}. Since the Hungarian algorithm is a classical algorithm, we will not introduce it. We refer a reader to Ref. [9].

Extended Weighted Bipartite Graph Matching

We have introduced the method to find optimal f for fixed P, g_0 and g_1. To find the globally optimal strategy, one could further enumerate all possible combinations between P, g_0 and g_1. Since in applications, $|P|$ is often small, one can easily find all possible P. For example, if $|P| = 2$, the time complexity is $O(|B|^2)$, where $|B| << |A|$, e.g., $|B| = 40$ and $|A| = 511$. Actually, as $\sum_{p\in P} h(p)$ should be no less than the size of required payload, the number of usable P could be significantly reduced during enumeration.

However, for a fixed P, one has to enumerate all possible g_0 and g_1. As $|x - g_0(x)| \leq T$ and $|x - g_1(x)| \leq T$, the time complexity to enumerate g_0 and g_1 is $O(\binom{2T+1}{2}^{|P|})$. This requires us to choose small $|P|$ or T since the time complexity has the exponential form. From an empirical viewpoint, one can set $G_0 = P$, i.e., $g_0(x) = x$, which has been used in traditional HS strategy. Thus, the time complexity is reduced as $O((2T+1)^{|P|})$, which is lower than the original one, yet still high.

It has motivated us to propose a new bipartite graph structure so that, for a fixed P, we can find the optimal (f, g_0, g_1) simultaneously by solving the MWMM task in the newly built bipartite graph. In this way, our task is only to enumerate all possible P. As mentioned above, $|P|$ can be small, indicating that, the globally optimal (P, f, g_0, g_1) can be determined effectively.

In the newly built bipartite graph, we have $V_1 = B \cup P_c$ and $V_2 = A$, where P_c is a "copy" of P. It does not mean $P = P_c$, but rather a newly inserted set. It means that, though P and P_c have the same values, they correspond to different sets in the bipartite graph. We rewrite V_1 as:

$$V_1 = B \cup P_c = (B \setminus P) \cup P \cup P_c. \tag{16}$$

Therefore, there are a total of $|A| + |B| + |P_c|$ vertices. All edges in the bipartite graph connect a vertex in V_1 and another one in V_2. Since V_1 can be divided into three disjoint subsets as shown in Eq. (16), we can also divide all the edges into three types. Namely, for the new bipartite graph $G(V, E)$,

$$E = \{(u, v)\ |u \in V_1, v \in V_2\} = E_1 \cup E_2 \cup E_3, \tag{17}$$

where $E_1 = \{(u, v) \mid u \in B \setminus P, v \in V_2\}$, $E_2 = \{(u, v) \mid u \in P, v \in V_2\}$ and $E_3 = \{(u, v) \mid u \in P_c, v \in V_2\}$.

In the newly built bipartite graph, the edges in E_1 show all possible shifting-relationship for the PEH bins belonging to $B \setminus P$. It means that, f will be corresponding to a subset of E_1. For any PEH bin v belonging to P, if the current bit to be embedded is "0", we will modify it as $g_0(v)$. Otherwise, $g_1(v)$ is used to replace v since the secret bit will be "1". Therefore, for all bins in P, there are two possible shifting operations, which depend on the value of the message bit to be embedded. To model the two shifting operations in the newly built bipartite graph, we use E_2 to indicate the shifting-relationship corresponding to g_0, and E_3 to indicate the shifting-relationship corresponding to g_1. Both g_0 and g_1 rely on P. That is why we propose to use a copy of P here to split the dependency, which is a highlight of this paper.

Suppose $V_1 = B \cup P_c = \{u_1, u_2, ..., u_{|V_1|}\}$ and $V_2 = A = \{v_1, v_2, ..., v_{|V_2|}\}$, for every possible index-pair (i, j), if $u_i \in B \setminus P$ and $|u_i - v_j| \leq T$, we add an edge $(u_i, v_j) \in E_1$, and the weight is determined as $C_{v_j - u_i}(u_i)$ according to Eq. (12). If $u_i \in P \subset V_1$ and $|u_i - v_j| \leq T$, we add an edge $(u_i, v_j) \in E_2$, and the weight is:

$$C_{v_j - u_i}(u_i) = \sum_{e_l{}^{(t)} = u_i} \delta(b_k, 0) \cdot (v_j - u_i + c_l{}^{(t)} - o_l{}^{(t)})^2, \tag{18}$$

where $b_k \in \{0, 1\}$ means the k-th (current) bit to be embedded. Otherwise, if $u_i \in P_c \subset V_1$ and $|u_i - v_j| \leq T$, we add an edge between u_i and v_j ($\in E_3$), and the weight is:

$$C_{v_j - u_i}(u_i) = \sum_{e_l{}^{(t)} = u_i} \delta(b_k, 1) \cdot (v_j - u_i + c_l{}^{(t)} - o_l{}^{(t)})^2. \tag{19}$$

All edges are assigned with a weight to specify the corresponding distortion. Accordingly, a weighted bipartite graph can be finally constructed. We find the MWMM in the newly built bipartite graph, by which we can compute the optimal (f, g_0, g_1) for a fixed P. One can enumerate P and repeatedly call the proposed optimization method to find the final (P, f, g_0, g_1).

Example of the MWMM

An example is provided here for better explanation. Suppose that, $A = \{-6, -5, ..., 6\}$, $B = \{-5, -3, -2, -1, 0, 1, 2, 3, 4\}$, $P = \{-1, 1\}$. Our task is to find the optimal (f, g_0, g_1) such that a total of $h(-1) + h(1)$ bits can be embedded while the introduced distortion is minimized. Notice that, we do not assume here that the histogram is Gaussian-like, but rather that the histogram can be arbitrary. Obviously, we have $B \setminus P = \{-5, -3, -2, 0, 2, 3, 4\}$ and $P_c = \{-1_c, 1_c\}$. Here, we use a mark "c" to distinguish the elements in P and that in P_c. It means that, "$-1_c \in P_c$" and "$-1 \in P$" are corresponding to two different vertices in the bipartite graph though they have the same value. In addition, we set $T = 2$. Figure 2 (a) shows the newly built bipartite graph, and (b) gives the MWMM. The optimal f, g_0 and g_1 are corresponding to black edges, blue edges and red edges shown in Figure 2 (b), respectively. For example, "1_c" is matched by "2", meaning that, $g_1(1) = 2$. And, "2" is matched by "3", i.e., $f(2) = 3$. In Figure 2 (a), $C_2(4)$ shows the distortion by shifting "4" to "'6'. $C_{-1}(-5)$ represents the distortion by shifting "-5" to "-6", here, $-1 = -6 - (-5)$.

Implementation Issues and Solutions

In this section, we address two key implementation issues.

Reversibility

A data hider has to self-embed P, g_0, g_1 and f. Since $|P|$ is small, the space to store P, g_0 and g_1 will be small. Self-embedding g_0, g_1 and f means to embed integer-pairs, e.g., $g_0(3) = 4$ tells us to embed ("3","4"). To embed $f : B \setminus P \mapsto V_2$, one can first sort elements in $B \setminus P$ in an increasing order, where the difference between two adjacent elements is often small (e.g., "-1" in most cases). We can use run-length encoding (RLE) or

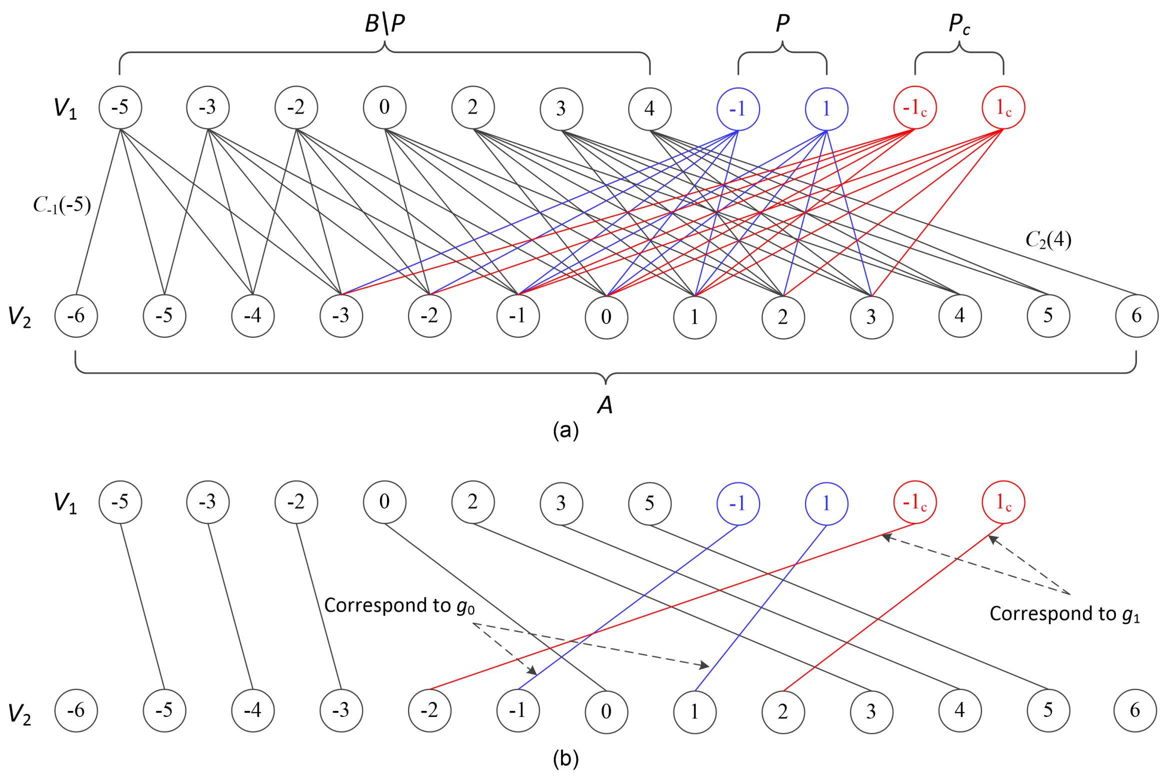

Figure 2. *An example to find optimal (f, g_0, g_1) for a fixed P: (a) the newly built weighted bipartite graph, (b) the MWMM. (Notice that, the MWMM is not unique.)*

arithmetic coding (AC) to compress the differences. The corresponding elements in V_2 should be recorded as well. Let $B\setminus P$ be $\{u_1, u_2, ..., u_{|B\setminus P|}\}$, where $u_1 < ... < u_{|B\setminus P|}$. we can compress $\{f(u_1)-u_1,\ f(u_2)-u_2, ..., f(u_{|B\setminus P|})-u_{|B\setminus P|}\}$ by RLE or AC since they are bounded by using a small T.

There are different methods to achieve self-embedding. For example, the data hider can choose a part of pixels (not in $\mathbf{c}^{(t)}$) to store the above-mentioned auxiliary data. The LSBs of these pixels will be kept unchanged throughout the specified-layer embedding such that one can successfully extract the hidden bits and recover the image content. The original LSBs will be considered as a part of the secret data.

Weight Determination

According to Eqs. (12, 18, 19), the computational complexity of weight determination is $O(n_t \times |V_1| \times (2T+1))$. One can determine the weights by an exhaustive way as long as the computational resource is enough. However, it would be more suitable to reduce the computational cost from the viewpoint of practical use. We aim to reduce the complexity. We rewrite Eq. (6) as:

$$\begin{aligned}\sum_{i=1}^{n_t}({s_i}^{(t)}-{o_i}^{(t)})^2 &= \sum_{i=1}^{n_t}({z_i}^{(t)}+\hat{e}_i^{(t)}-{o_i}^{(t)})^2\\ &= \sum_{i=1}^{n_t}({c_i}^{(t)}-{o_i}^{(t)}+\hat{e}_i^{(t)}-{e_i}^{(t)})^2 \qquad (20)\\ &= \sum_{i=1}^{n_t}({\alpha_i}^{(t)}+{\beta_i}^{(t)})^2,\end{aligned}$$

where ${\alpha_i}^{(t)} = {c_i}^{(t)}$-${o_i}^{(t)}$ and ${\beta_i}^{(t)} = \hat{e}_i^{(t)}$-${e_i}^{(t)} = \Delta({e_i}^{(t)})$.

Obviously, all $\alpha_i^{(t)}$ $(1\leq i\leq n_t)$ are constants before data embedding, meaning that, they can be determined in advance. And, $|{\beta_i}^{(t)}|\leq T$ for all $1\leq i\leq n_t$. We use $\mathbf{H}=\{H_{u,v}|-|I|<u,v<|I|\}$ to record the occurrence of every possible pair (u,v), where u represents the possible value of ${\alpha_i}^{(t)}$ and v shows the possible value of ${e_i}^{(t)}$. Then, we have:

$$\begin{aligned}\sum_{i=1}^{n_t}({\alpha_i}^{(t)}+{\beta_i}^{(t)})^2 &= \sum_u\sum_v H_{u,v}\cdot(u+\Delta(v))^2\\ &= \sum_u\sum_{v\in P} H_{u,v}\cdot(u+\Delta(v))^2 \qquad (21)\\ &\quad+\sum_u\sum_{v\in B\setminus P} H_{u,v}\cdot(u+\Delta(v))^2,\end{aligned}$$

where

$$\begin{aligned}\sum_{v\in P}H_{u,v}\cdot(u+\Delta(v))^2 &= \sum_{v\in P}\frac{H_{u,v}}{2}\cdot(u+g_0(v)-v)^2\\ &\quad+\sum_{v\in P}\frac{H_{u,v}}{2}\cdot(u+g_1(v)-v)^2,\end{aligned}$$

and

$$\sum_{v\in B\setminus P}H_{u,v}\cdot(u+\Delta(v))^2 = \sum_{v\in B\setminus P}H_{u,v}\cdot(u+f(v)-v)^2.$$

We take Eq. (12) as the task to be addressed. Eqs. (18, 19) can be processed according to a similar way. Based on Eq. (21), we rewrite Eq. (12) as:

$$\begin{aligned} C_k(y_j) &= \sum_{e_i^{(t)}=y_j} (k + c_i{}^{(t)} - o_i{}^{(t)})^2 \\ &= \sum_u \sum_{v=y_j} H_{u,v} \cdot (u+k)^2, \end{aligned} \tag{22}$$

which is corresponding to a time complexity of $O(|I|^2)$.

Therefore, once we find $\mathbf{H}$ with a complexity of $O(n_t)$, for each edge, we can obtain its weight with a complexity of $O(|I|^2)$, resulting in a whole complexity of $O(n_t + |V_1| \times (2T+1) \times |I|^2)$, which significantly outperforms the enumeration strategy.

Simulation Results and Analysis

In this section, we conduct experiments to evaluate the proposed work. Since our work was originally designed for HS operation, it is fair and necessary to compare it with HS based works. We incorporate the proposed optimization algorithm into four HS based invertible embedding algorithms, i.e., PC_HS [10], RW_SP [3], GF_HS A1 [8] and DCSPF [7], to evaluate the rate-distortion performance. In our experiments, for an embedding algorithm, we only optimize the HS operation, meaning that, the others such as pixel prediction, pixel selection and local-complexity function are all the same as the original ones.

Experimental Setup

Since the time complexity of enumerating all P has the factorial form, for simplicity, we use $|P| = 2$ for all embedding algorithms and their optimized versions so that searching the optimal P will not be time consuming. The optimized versions of the above-mentioned algorithms are denoted by "PC_HS opt", "RW_SP opt", "GF_HS A1 opt" and "DCSPF opt", respectively. It is pointed that, for the original embedding algorithms, one can actually use multiple PEH bins to carry the secret data. Therefore, it can be said that, we actually simplify the embedding operation.

In PC_HS and GF_HS A1, one has to use non-overlapped pixel-blocks to carry secret bits. The block-size is set to be 3×3 for both methods, which is the same as described in the two methods. In GF_HS A1, before data embedding, the authors use a pixel selection parameter s to take advantage of smooth pixels. In our simulation, when to use the proposed optimization method, there has no need to determine s directly since we can sort the local-complexities in an increasing order such that smooth pixels can be utilized for data embedding, which is equivalent to using s. It means that, we will not determine and store s for GF_HS A1 opt. The other content-dependent operations for PC_HS opt and GF_HS A1 opt are the same as PC_HS and GF_HS A1.

In DCSPF, one has to set two important system parameters, i.e., the pixel-blocking rate and the number of pixel-selection layers. We enumerate the pixel-blocking rate from 10% to 90% with a step of 10%, and vary the number of pixel-selection layers from 3 to 6 with a step of 3 for both DCSPF and DCSPF opt. Other operations for DCSPF opt are the same as DCSPF.

For each non-optimized embedding algorithm, with an image, boundary pixels are adjusted into the reliable range in advance according to T. The resulting location map will be losslessly compressed by AC and self-embedded into the preprocessed image together with the secret message. The HS parameters and secret key are embedded into the LSBs of pixels in the border of an image. The original LSBs are collected and self-embedded together with the secret message. For each optimized algorithm, we use the LSBs of pixels in the border of an image to store (f, g_0, g_1) and secret key, the original LSBs are collected and self-embedded.

Table 1. The averaged size (bits, rounded to integer) of losslessly compressed codes.

Algorithms	T = 1	T = 2	T = 3
PC_HS opt	357	471	481
GF_HS A1 opt	140	152	154
DCSPF opt	149	160	169
RW_SP opt	188	257	264

Multiple-pass embedding strategy [8] was used for PC_HS, GF_HS A1 and their optimized versions. In RW_SP, a data hider can first use the dot set for data embedding and cross set for prediction. Then, the cross set can be used for data embedding, and the dot set for prediction, which is also a kind of multiple-pass strategy. For multi-layer embedding, since it is free to set the payload size of each layer, in default, we will embed additional bits into a specified layer as much as possible controlled by a payload-step until it cannot carry additional bits, and the higher-layer embedding is then applied if the entire payload is not fully carried. Notice that, this strategy may be not optimal. We use a random bit-string to represent the to-be-embedded data. For PC_HS, RW_SP, and GF_HS A1, the payload-step is set as 5000. For DCSPF, the payload-step is set as 1000. Notice that, it is always free to set the payload-step. The payload-steps for the optimized systems are the same as their non-optimized versions.

Auxiliary Data and Runtime Analysis

With the proposed method, one can obtain the optimized P, f, g_0, and g_1. Comparing with an original system, the optimized system may carry more side information, which is mainly affected by (f, g_0, g_1). Therefore, we have to analyze the size of losslessly compressed auxiliary data used for storing (f, g_0, g_1).

We take *Airplane*, *Lena*, *Baboon* and *Sailboat* (from smooth to complex) sized $512 \times 512 \times 8$ for experiments. With the aforementioned payload-step, we orderly increase the entire payload-size started from 1×10^4. For each image, during data embedding, when the required (f, g_0, g_1) is determined, we losslessly encode them and compute the size of compressed code. For each image, 50 compressed codes corresponding to different (f, g_0, g_1) are orderly collected. The mean size is computed. The mean sizes for the four test images are further averaged as the final result.

Table 1 shows the results due to different T. It is seen that, different algorithms have different sizes of the compressed code. A larger T usually has a relatively larger size of the compressed code. In cases $T = 2$ and $T = 3$, the sizes are close to each other, meaning that, when T increases, the size will not significantly increase. Overall, the sizes are kept relatively low. It is also admitted that, when the size of entire payload is quite small, e.g., 1000 bits, the ratio of such auxiliary data would be relatively large. Therefore, it is not recommended to embed a quite small payload in practice when to use the optimized model. Furthermore, PC_HS opt has a relatively larger size of compressed code. The reason is, unlike other methods, in PC_HS, the prediction of

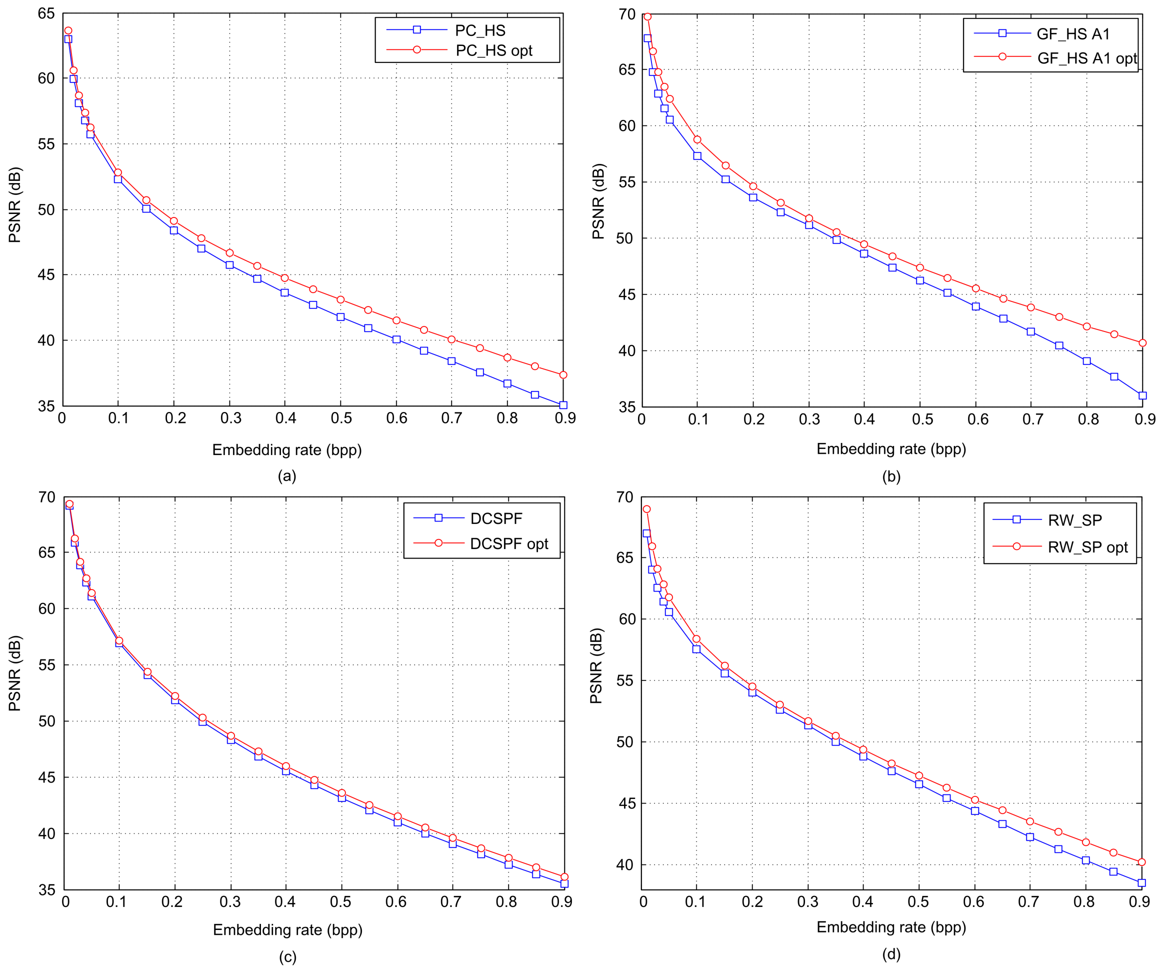

Figure 3. *The rate-distortion performance comparison with a set of 200 images (bpp: bits per pixel). The embedded payloads include the side information.*

Table 2. The averaged running time (in seconds) of executing the proposed optimization procedure.

Algorithms	T = 1	T = 2	T = 3
PC_HS opt	5.26	7.74	11.38
GF_HS A1 opt	0.31	0.53	0.82
DCSPF opt	0.83	1.55	2.55
RW_SP opt	0.67	1.37	2.34

a pixel simply equals the central pixel in the corresponding block, which results in a larger $|B|$ that cannot benefit the compression procedure. It can be said that, though our work does not rely on any specific content-dependent procedure, well-designed content-dependent procedure can keep the size of compressed code low.

It is necessary to analyze the running time for use. As mentioned above, we collect compressed codes for each test image. It is true that, during the process, we can simultaneously determine the running time of executing the proposed optimization procedure. The averaged running time can be computed similar to Table 1. We use MATLAB R2012b and Microsoft Visual C++ 2010 (compiler, for mex files) as the simulation platform. The processor (x64-based) information of our used laptop (Windows 8, 64-bit OS) is: Intel(R) Core(TM) i5-4200U CPU 1.60 GHz. And, the installed memory (RAM) is 4.00 GB (3.90 GB usable). Table 2 shows the results due to different T. A larger T implies a higher running time of executing the proposed optimization procedure. Even though different methods have different computational costs, the running time is overall acceptable. As shown in Table 2, the running time for PC_HS opt is significantly higher than other methods, which is due to the different content-dependent operations. It implies that, well-designed content-dependent process has the ability to keep the running time low.

Rate-distortion Performance Evaluation

We use 200 images randomly selected from BOSSBase 1.01 [11] to evaluate the rate-distortion performance. The test images are all grayscale and sized $512 \times 512 \times 8$. For a payload, the mean PSNR is used as the distortion measure. A payload to be embedded is treated as a random bit-string. We optimize T in range [1, 3]. Figure 3 shows the comparison results for different algorithms. It is observed that, different embedding systems equipped with the proposed approach result in different improvement. The reason includes at least two aspects: 1) different algorithms have different content-dependent operations, which allow different al-

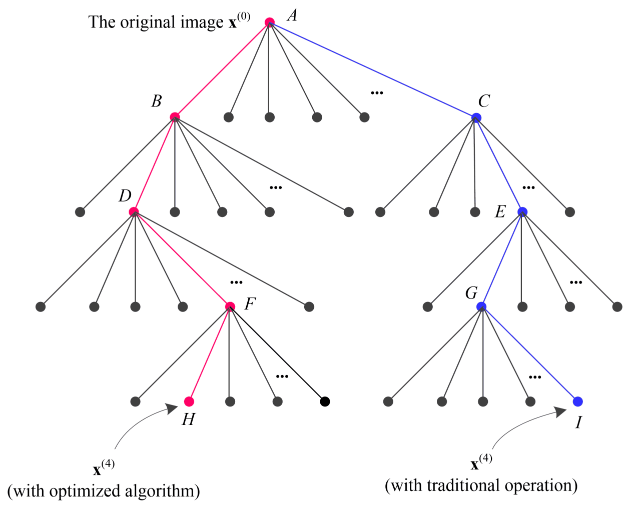
The original image $\mathbf{x}^{(0)}$
A
B
C
D
E
F
G
H
I
$\mathbf{x}^{(4)}$
(with optimized algorithm)
$\mathbf{x}^{(4)}$
(with traditional operation)

Figure 4. *An intuitive explanation to rate-distortion optimization of invertible embedding (multi-layer) from the perspective of a search tree model.*

gorithms to use different PEHs for data embedding; and 2) different images have different statistical characteristics, enabling an algorithm to use different image areas for embedding and thus provide different performance. Though the performance vary for different optimized embedding algorithms, overall, the optimized algorithms could improve their original versions significantly.

For DCSPF opt, the performance improvement is not as significant as other algorithms. Comparing with other algorithms, the rate-distortion performance of DCSPF opt is close to that of D-CSPF. The reason is that, in DCSPF, an efficient dynamic content-dependent approach is proposed to take advantages of the smooth pixels as much as possible. Meanwhile, an enumeration manner for searching near-optimal pixel selection parameters is applied, which allows a hider to select the best PEH from many candidates for data embedding. It makes small room for performance improvement since the proposed model focuses on the combinatorial optimization of the HS operation while the other steps are the same as the original ones. Therefore, it is possible that the performance improvement for a well-designed system equipped with the proposed optimization procedure is not significant.

Example of Optimized HS Operation

We further show an example observed from our experiments that the optimized HS operation was different from the traditional one. We take RW_SP opt for explanation. We set $T = 2$ and embed a total of 10.5×10^4 bits (≈ 0.40 bpp) into *Lena* with two embedding-layers controlled by above-mentioned step-value. Thereafter, in the third layer, we first embed 1.5×10^4 bits into the dot set, and then embed 1.5×10^4 bits into the cross set. We analyze the optimized (f, g_0, g_1) for the cross set. We find that $P = \{-6, 5\}$, $h(-6) = 6738$, $h(-7) = 5011$, $h(5) = 8262$ and $h(6) = 5934$. And, $g_0(-6) = -6$, $g_1(-6) = -8$, $g_0(5) = 5$, $g_1(5) = 7, f(-7) = -7, f(6) = 6$. It is seen that, the optimized g_1 is different from the traditional operation which corresponds to $g_1(-6) = -7$ and $g_1(5) = 6$. To this end, we determine the shifting distortion between different PEs according to Eqs. (12, 18, 19), namely, $C_0(-6) = 13035$, $C_{-1}(-6) = 29586$, $C_{-2}(-6) = 52875$, $C_0(-7) = 19168$, $C_{-1}(-7) = 43639$, $C_0(5) = 15976$, $C_1(5) = 36265$, $C_2(5) = 64816$, $C_0(6) = 22902$, $C_1(6) = 52016$. It is easy to determine the distortion for optimized (g_0, g_1), i.e., $d_0 = C_{-2}(-6) + C_0(-6) + C_0(5) + C_2(5) = 146702$. Notice that, $C_{-2}(-6)$ shows the distortion introduced by modifying a half bins with a value of "-6" as that with a value of "-8". The computation for $C_{-1}(-6)$, $C_0(-6)$, $C_0(5)$, $C_1(5)$ and $C_2(5)$ are similar. The traditional operation corresponds to $d_1 = C_{-1}(-6) + C_0(-6) + C_0(5) + C_1(5) = 94862$. We need to further consider the distortion introduced by f. For simplicity, we here only consider the distortion impact introduced by the PEs "-7" and "6". For optimized f, we have $d_2 = C_0(-7) + C_0(6) = 42070$. While for the traditional operation, we have $d_3 = C_{-1}(-7) + C_1(6) = 95655$. Obviously, though $d_0 > d_1$, we have $d_0 + d_2 = 188772 < d_1 + d_3 = 190517$, indicating that, the overall distortion introduced by the optimized operation will be lower than the traditional operation, which has verified the superiority. Notice that, the overall distortion introduced by optimized (f, g_0, g_1) does not equal $d_0 + d_3$. We actually show the simplified derivation here. In practice, one should further take into account the other PEs for computing the overall distortion.

We also found that, in a very few cases, the optimized algorithms introduce a relatively slightly higher distortion. We explain this normal phenomenon from the perspective of a search tree model. For both the optimized algorithm and its original version, the embedding procedure corresponds to a path in the search tree. For example, in Figure 4, the node-paths for an optimized algorithm and its traditional version are $A \to B \to D \to F \to H$ and $A \to C \to E \to G \to I$, respectively. For an optimized algorithm, starting from the root node (corresponding to the original host), a suboptimal node (in terms of rate-distortion performance) is selected out at each layer. For the non-optimized version, it selects a child-node that may be not optimal for the corresponding layer. For example, node B outperforms node C in terms of rate-distortion performance. However, when the data-embedding layer becomes deeper, it is possible that in some layer, the traditional operation may be better than the optimized version since the selection of a node also relies on its father-node. For example, in Figure 4, it is possible that nodes $\{B, D, F\}$ outperform $\{C, E, G\}$ while I may provide a lower distortion than H since the generation of I and H rely on $\{A, G\}$ and $\{A, F\}$ respectively, which are two different inputs for distortion optimization.

We would like to point that, when to produce $\mathbf{x}^{(t+1)}$, the traditional HS operation only corresponds to a maximum matching, it may not ensure the minimum distortion. Therefore, the rate-distortion performance of a scheme equipped with the proposed optimization method will not be worse than that with the traditional operation for the identical input. If the traditional HS strategy is optimal, our method will surely find it out.

Conclusion and Discussion

Mainstream invertible embedding operations can be generalized by injective functions (g_0, g_1, f), relying on P. We use graph matching model to optimize the functions. We first fix (g_0, g_1) and model f in a weighted bipartite graph. Then, we relax (g_0, g_1) and model (g_0, g_1, f) in an extended weighted bipartite graph. Experimental results have shown the superior performance. A lot of practical optimization problems can be modeled as a graph problem. The core work is to build equivalence between the target problem and some graph model. Modeling information hiding in graphs is an important and promising topic. There are some related works [12], [13], [14], that have been reported for steganography. To the best knowledge of the authors, few works study the rate-distortion optimization of invertible embedding with graphic optimization models. This work presents a different perspective to invertible embedding. In the future, we are to improve the proposed algorithm. For example, the existing work [15] has shown that one can use multiple PEs as the embedding unit, which may provide superior performance. However, when to model multiple PEs as an embedding unit, the corresponding graph model will be more complex and difficult. We address this problem in future, and investigate steganography using graph models.

Acknowledgement

It was supported by National Natural Science Foundation of China under grant numbers 61902235, U1636206, U1936214, and 61525203. It was also supported by "Chen Guang" project co-funded by the Shanghai Municipal Education Commission and Shanghai Education Development Foundation.

References

[1] I. Cox, M. Miller, J. Bloom, J. Fridrich, and T. Kalker. Digital watermarking and steganography. *Morgan Kaufmann*, Nov. 2007.

[2] Z. Ni, Y. Shi, N. Ansari, and W. Su. Reversible data hiding. *IEEE Trans. Circuits Syst. Video Technol.*, 16(3): 354-362, Mar. 2006.

[3] V. Sachnev, H. Kim, J. Nam, S. Suresh, and Y. Shi. Reversible watermarking algorithm using sorting and prediction. *IEEE Trans. Circuits Syst. Video Technol.*, 19(7): 989-999, Apr. 2009.

[4] J. Fridrich, M. Goljan, and R. Du. Lossless data embedding - new paradigm in digital watermarking. *EURASIP J. Applied Signal Process.*, 2002(2): 185-196, Dec. 2002.

[5] J. Tian. Reversible data embedding using a difference expansion. *IEEE Trans. Circuits Syst. Video Technol.*, 13(8):890-896, Aug. 2003.

[6] H. Wu, Y. Shi, and H. Wang. PPE-based reversible data hiding. In: *Proc. ACM Workshop Inf. Hiding Multimed. Security*, pp. 187-188, Jun. 2016.

[7] H. Wu, Y. Shi, and H. Wang. Dynamic content selection-and-prediction framework applied to reversible data hiding. In: *Proc. IEEE Workshop Inf. Forensics Security*, pp. 1-6, Dec. 2016.

[8] X. Li, B. Li, B. Yang, and T. Zeng. General framework to histogram-shifting-based reversible data hiding. *IEEE Trans. Image Process.*, 22(6): 2181-2191, Jun. 2013.

[9] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. Introduction to algorithms. *The MIT Press*, Jul. 2009.

[10] P. Tsai, Y. Hu and H. Yeh, "Reversible image hiding scheme using predictive coding and histogram shifting," *Signal Process.*, 89(6): 1129-1143, Jun. 2009.

[11] P. Bas, T. Filler, and T. Pevný, "Break our steganographic system - the ins and outs of organizing BOSS," In: *Int. Workshop Inf. Hiding (IH'11)*, vol. 6958, pp. 59-70, May 2011.

[12] H. Wu and H. Wang, "Multibit color-mapping steganography using depth-first search," In: *IEEE Int. Symp. Biometrics and Security Technol. (ISBAST'13)*, pp. 224-229, Jul. 2013.

[13] S. Hetzl and P. Mutzel. "A graph-theoretic approach to steganography," In: *Proc. Int. Conf. Commun. Multimed. Security*, vol. 3677, pp. 119-128, Sept. 2005.

[14] H. Wu, H. Wang, H. Zhao and X. Yu, "Multi-layer assignment steganography using graph-theoretic approach," *Multimed. Tools Appl.*, 74(18): 8171-8196, Sept. 2015.

[15] B. Ou, X. Li, Y. Zhao, R. Ni and Y. Shi, "Pairwise prediction-error expansion for efficient reversible data hiding," *IEEE Trans. Image Process.*, 22(12): 5010-5021, Dec. 2013.

Author Biography

Hanzhou Wu received both B.Sc. and Ph.D from Southwest Jiaotong University, Chengdu, China, in 2011 and 2017. From 2014 to 2016, he was a visiting scholar in New Jersey Institute of Technology, New Jersey, United States. He was a researcher in Institute of Automation, Chinese Academy of Sciences, from 2017 to 2019. Currently, he is an Assistant Professor in Shanghai University, China. His research interests include information hiding, graph theory and game theory. He has published around 20 papers in peer journals and conferences such as IEEE TDSC, IEEE TCSVT, IEEE WIFS, ACM IH&MMSec, and IS&T Electronic Imaging, Media Watermarking, Security and Forensics.

Xinpeng Zhang received B.Sc. from Jilin University, China, in 1995, and the M.S. and Ph.D. from Shanghai University, in 2001 and 2004, respectively. Since 2004, he has been with the faculty of the School of Communication and Information Engineering, Shanghai University, where he is currently a full-time Professor. He is also with the faculty of the School of Computer Science, Fudan University. He was with The State University of New York at Binghamton as a Visiting Scholar from 2010 to 2011, and also with Konstanz University as an experienced Researcher, sponsored by the Alexander von Humboldt Foundation from 2011 to 2012. His research interests include multimedia security, image processing, and digital forensics. He has published over 200 research papers. He served an Associate Editor of IEEE Transactions on Information Forensics and Security.