Efficient Provably Algorithms for Boundary Bin-Pair Selection Problem*

CHUN-LIANG JHONG AND HSIN-LUNG WU

Department of Computer Science and Information Engineering
National Taipei University
No.151, Daxue Road, Sanxia District, New Taipei City 237, Taiwan
E-mail: liang851015@gmail.com; hsinlung@mail.ntpu.edu.tw

Given a non-negative vector, a boundary bin pair for this vector consists of two different vector entry indices and the boundary bin pair determines a set of bins whose indices are between the boundary bin indices. The sum of these bins is called the bounded area for this given non-negative vector. The boundary bin-pair selection problem (BBPS problem) is an optimization problem in which, \( n \) non-negative vectors of length \( m \) and a capacity value \( T \) are given, the goal is to determine \( n \) bin pairs for these \( n \) histogram vectors such that the sum of corresponding \( n \) bounded areas is maximized under the constraint that the sum of 2\( n \) boundary bins is greater than or equal to \( T \). The BBPS problem is shown to be equivalent to the modified multiple-histogram-modification problem which is a well-known research topic in reversible data hiding.

A brute-force way to solve the BBPS problem requires exponential time. In this paper, we construct several efficient provably algorithms toward solving the BBPS problem. Our proposed algorithms are constructed in a dynamic-programming way. These proposed algorithms achieves the same time complexity \( O(m^2T) \). In order to compare their efficiency, we generate experimental data by constructing histogram vectors from the prediction errors of some test images. Experimental results give a clear performance comparison between our proposed algorithms for the BBPS problem.

Keywords: Boundary bin-pair selection, histogram vectors, optimization, dynamic programming, multiple histogram modification

1. INTRODUCTION

Given a non-negative vector \( V = (V_1, \ldots, V_m) \in \mathbb{R}^m \) (called \( V \) a histogram vector) and two indices \((\ell, r)\) with \( 1 \leq \ell < r \leq m\), the two bins \( V_\ell \) and \( V_r \) are called the boundary bins for \( V \). The boundary bin pair determines a set of bins whose indices are greater than \( \ell \) and smaller than \( r \) and thus determines the sum \( A_{\ell, r} = \sum_{\ell < k < r} V_k \) (called \( A_{\ell, r} \) the bounded area). Given \( n \) histogram vectors \( V^1, V^2, \cdots, V^n \in \mathbb{R}^m \) and a natural number \( T \), the goal of the boundary bin-pair selection problem (BBPS problem) is to find \( n \) boundary bin pairs \(<(V^i_\ell, V^i_r) : 1 \leq i \leq n>\) such that the sum of bounded area \( A^1_{\ell_1, r_1}, \cdots, A^n_{\ell_n, r_n} \) is maximized under the constraints that the sum of boundary bins is at least \( T \).

Received xxx xx, 2020; revised XXX XX, XXXX; accepted xxx xx, XXXX.
Communicated by the editor.
The BBPS problem has found applications in some research areas such as reversible data hiding [8]. One of invertible ways to embed secret message into an image is based on the method of multiple histograms modification (MHM) proposed by Li et al. [4] where the method for finding good pixel values to embed message is modeled as an optimization problem called the general MHM problem. Up to our knowledge, there is no efficient algorithm which can solve the general MHM problem. Instead of the general MHM problem, the modified MHM problem is suggested in [1, 7]. The BBPS problem captures the modified MHM problem. Thus, an optimal solution of the BBPS problem is an approximation of the optimal solution of the general MHM problem. Therefore, one can use it to obtain reversible data hiding with small distortion.

In [1], a dynamic-programming algorithm is proposed to solve the BBPS problem. However, the time complexity of this algorithm is $O(nT^2 + nm^2T)$. In [7], Qi et al. consider a special case of the modified MHM problem in which the selected boundary bin indices satisfy a specific constraint. The considered case can be seen as the special case of the BBPS problem. Qi et al. proposed a heuristic algorithm of time complexity $O(nmT)$ toward solving this special case. However, there is no correctness proof for their algorithm.

In this paper, we propose several provably efficient dynamic programming algorithms for solving the BBPS problem. The time complexity of the proposed algorithms is $O(nm^2T)$ which improves the time complexity of the algorithm proposed in [1]. Moreover, the correctness proof for these proposed algorithms is also given in order to guarantee the optimality. In addition, our algorithms also solves the BBPS problem with specific boundary bin-pair constraints within time $O(nmT)$. The correctness of the algorithms is also guaranteed. As an application, our proposed algorithm immediately solves the modified MHM problem and provides an improved MHM-based reversible data hiding scheme. Although our proposed algorithms have the same time complexity order in a theoretical view, we also use experimental data to give a performance comparison for these algorithms in a practical view. Experimental results show that two of our proposed algorithms called Quick-BBPS and fQuick-BBPS have much better performance than other proposed algorithms.

2. Boundary Bin-Pair Selection Problem

In this section, we define the boundary bin-pair selection problem (BBPS problem) and show an efficient dynamic programming algorithm toward solving it. A vector $V = (V_1, \cdots, V_n) \in \mathbb{R}_m^m$ is called a histogram vector if $V_i \geq 0$ for each $1 \leq i \leq m$. Let $\mathbb{R}_m^m$ denote the set of histogram vectors of length $m$.

**Problem.** (Boundary Bin-Pair Selection Problem) Given $n$ vectors $V^1, V^2, \cdots, V^n \in \mathbb{R}_m^m$ and a natural number $T$, find $n$ index pairs $< (\ell_i, r_i) : 1 \leq i \leq n >$ with $1 \leq \ell_i < r_i \leq m$ that maximizes the sum of bounded areas $\sum_{i=1}^n \sum_{j=\ell_i+1}^{r_i-1} V^i_j$ subject to $\sum_{i=1}^n V^i_{\ell_i} + V^i_{r_i} \geq T$.

A brute-force way to solve the boundary bin-pair selection problem is to try all possible $n$ pairs $< (\ell_i, r_i) : 1 \leq i \leq n >$ with $\sum_{i=1}^n V^i_{\ell_i} + V^i_{r_i} \geq T$ and find the maximum value of $\sum_{i=1}^n \sum_{j=\ell_i+1}^{r_i-1} V^i_j$. However, this requires $O(\binom{m}{2}^n)$ time which is extremely inefficient for
large \( n \).

Next, a special subclass of BBPS problem called the restricted boundary bin-pair selection problem is defined as follows.

**Problem.** (Restricted Boundary Bin-Pair Selection Problem) Given \( n \) vectors \( \mathbf{V}^1, \mathbf{V}^2, \ldots, \mathbf{V}^n \in \mathbb{R}^{m_0}_+ \), a natural number \( T \), and \( n \) constraint functions \( f_1, \ldots, f_n \), find \( n \) index pairs \( \langle (t_i, f_i(t_i)) : 1 \leq i \leq n \rangle \) with \( 1 \leq t_i \leq m \) that maximizes the sum of bounded subproblem at \( t_i \) \( \sum_{j=i}^{t_i-1} V^j \) subject to \( \sum_{j=1}^{n} V^j_{t_i} + V^j_{f_i(t_i)} \geq T \).

It is easy to see that the time complexity of the brute-force way to solve the restricted BBPS problem is \( O(m^n) \) time. In the next section, we propose an efficient dynamic programming algorithm to solve the general BBPS problem.

### 3. A Primitive Algorithm for the BBPS Problem

The proposed algorithm is designed in a dynamic programming way. Given \( n \) vectors \( \mathbf{V}^1, \mathbf{V}^2, \ldots, \mathbf{V}^n \in \mathbb{R}^{m_0}_0 \) and \( u \in \mathbb{Z} \), we define

\[
B[s,t,u] = \max_{<(t_i,r_i): s \leq i \leq t >} \sum_{i=s}^{t} \sum_{j=i+1}^{t_i-1} V^j_i
\]

subject to

\[
\sum_{i=s}^{t} V^j_i + V^j_{r_i} \geq u.
\]

If there is no \( <(t_i,r_i) : s \leq i \leq t > \) satisfying \( \sum_{i=s}^{t} V^j_i + V^j_{r_i} \geq u \), then we define \( B[s,t,u] = -\infty \). In addition, we define \( B[s,t,u] = B[s,t,0] \) for every \( u < 0 \).

Actually, \( B[s,t,u] \) is the subproblem of the BBPS problem. In particular, \( B[1,n,T] \) is exactly the BBPS problem.

Next, we show some properties of \( B[s,t,u] \) and use them to construct an efficient algorithm for the BBPS problem. First of all, we give a recursive relation of \( B[s,t,u] \) in the following theorem.

**Theorem 1.**

\[
B[s,t,u] = \begin{cases} 
\max_{0 \leq k \leq u} \left( B[s,t-1,k] + B[t,t,u-k] \right) & \text{if } s < t \\
\max_{0 \leq k \leq u} \left( B[s,t-1,k] + B[t,t,u-k] \right) & \text{if } s = t
\end{cases}
\]

**Proof.** We give a recursive relation for \( B[s,t,u] \). Note that, for \( 1 \leq s \leq n \) and any \( 0 \leq u \leq T \), \( B[s,s,u] \) can be computed within time \( O(m^2) \) by finding one of \( \binom{m}{2} \) pairs \( (t_i, r_i) \) with \( V^j_{t_i} + V^j_{r_i} \geq u \) such that \( \sum_{j=t_i+1}^{t_i-1} V^j_j \) is maximum. Thus, \( B[s,s,u] = \max_{0 \leq k \leq u} \left( B[s,s-1,k] + B[s,s,u-k] \right) \). We will give a more efficient algorithm to compute \( B[s,s,u] \) later.

Next, we consider the case to compute \( B[s,t,u] \) when \( s < t \). By induction on \( t - s \), we assume that it is efficient to find an optimal solution for \( B[s',t',u'] \) for \( s \leq s' \leq t' \leq t \).
$t$ with $t' - s < t - s$ and $0 \leq u' \leq T$. Let $c$ be any integer with $s \leq c < t$. Suppose that $<(l_i, r_i) : s \leq i \leq t>$ is an optimal solution for $B[s, t, u]$. Let $u_c$ be the value with $u_c = \sum_{i=s}^{t} V_{l_i}^{i} + V_{r_i}^{i}$. Note that $<(l_i, r_i) : s \leq i \leq c >$ and $<(l_i, r_i) : c + 1 \leq i \leq t >$ must be optimal solutions that witness $B[s, c, u_c]$ and $B[c + 1, t, u - u_c]$, respectively. Otherwise, we can find another solution $<(l_i', r_i') : s \leq i \leq t >$ such that $\sum_{i=s}^{t} \sum_{j=l_i'+1}^{r_i'-1} V_j^i > \sum_{i=s}^{t} \sum_{j=l_i+1}^{r_i-1} V_j^i$ and $\sum_{i=s}^{t} V_{l_i}^{i} + V_{r_i}^{i} \geq u$. However, this contradicts the fact that $<(l_i, r_i) : s \leq i \leq t >$ is an optimal solution for $B[s, t, u]$. Therefore, we conclude that $B[s, t, u] = B[s, c, u_c] + B[c + 1, t, u - u_c]$. Since $u_c$ is an integer between 0 and $T$, we obtain the following recurrence relation: $B[s, t, u] = \max_{0 \leq k \leq u} \{B[s, c, k] + B[c + 1, t, u - k]\}$. There are many ways to set the parameter $c$. In our algorithm, we set $c = t - 1$. Thus, the $B[s, t, u]$ can be recursively defined by $B[s, t, u] = \max_{0 \leq k \leq u} \{B[s, t - 1, k] + B[t, t, u - k]\}$.

Based on Theorem 1, we immediately obtain the following algorithm called Primitive-BBPS which computes an optimal solution that witnesses $B[1, n, T]$.

**Primitive-BBPS** with input $V^1, \cdots, V^n \in \mathbb{R}_{\geq 0}^m$

\[
\arg \max_{<(l_i, r_i) : 1 \leq i \leq n, \sum_{i=1}^{n} \sum_{j=l_i+1}^{r_i-1} V_j^i \geq T, \sum_{i=1}^{n} V_{l_i}^{i} + V_{r_i}^{i} \geq T}} \text{subject to } \sum_{i=1}^{n} V_{l_i}^{i} + V_{r_i}^{i} \geq T \]

is obtained as follows:

1. For $1 \leq s \leq n$ and for $0 \leq u \leq T$, compute $B[s, s, u] = \max_{(l_i, r_i) : V_{l_i}^{i} + V_{r_i}^{i} \geq u } \sum_{i=1}^{n} \sum_{j=l_i+1}^{r_i-1} V_j^i$ and $P[s, s, u] = \arg \max_{(l_i, r_i) : V_{l_i}^{i} + V_{r_i}^{i} \geq s } \sum_{i=1}^{n} \sum_{j=l_i+1}^{r_i-1} V_j^i$.

2. For $s = 2$ to $n$ and for $0 \leq u \leq T$, compute

\[
B[1, s, u] = \max_{0 \leq k \leq u} \{B[1, s - 1, k] + B[s, s, u - k]\}
\]

and

\[
P[1, s, u] = \arg \max_{0 \leq k \leq u} \{B[1, s - 1, k] + B[s, s, u - k]\}.
\]

3. Set $u_j = 0$ if $j = 0$, $u_j = P[1, j + 1, u_{j+1}]$ if $0 < j < n$, and $u_j = T$ if $j = n$.

4. Output $\{P[i, i, u_i - u_{i-1}] : 1 \leq i \leq n\}$.

**Primitive-BBPS** is proposed previously in [1]. It is an efficient algorithm since its running time is $O(nT + nT^{2})$ which is much lower than $O(T^{n})$ for large $n$. However, for applications, the time complexity of **Primitive-BBPS** is still high in terms of $T$. We will give two improved versions of **Primitive-BBPS** in the next section.

### 4. Improved Versions of Primitive-BBPS

In this section, we improve the time complexity of **Primitive-BBPS** algorithm to be a linear-time algorithm in terms of $T$. First of all, we show more properties of $B[s, t, u]$.

**Lemma 1.** For any fixed $s, t$ with $s \leq t$, $B[s, t, q] \leq B[s, t, p]$ for any $0 \leq p < q \leq T$. 

**Proof.** The proof is by induction on \(d = t - s\). First, we prove the lemma for the base case that \(d = t - s = 0\). By way of contradiction, suppose that there are two integers \(p, q\) with \(p < q\) such that \(B[t, t, p] < B[t, t, q]\). Let \((\ell_p, r_p)\) and \((\ell_q, r_q)\) denote the optimal selected bin pairs for \(B[t, t, p]\) and \(B[t, t, q]\), respectively. By definition, \(\sum_{j=\ell_q+1}^{r_q-1} V_j^q\) is maximum subject to \(V_j^q \geq q\). Since \(p < q\) and \(B[t, t, p] < B[t, t, q]\), we have

\[
V_{\ell_q}^q + V_{r_q}^q \geq q > p
\]

and

\[
\sum_{j=\ell_q+1}^{r_q-1} V_j^q > \sum_{j=\ell_p+1}^{r_p-1} V_j^p = B[t, t, p].
\]

Thus \((\ell_q, r_q)\) is a better feasible selection than \((\ell_p, r_p)\) for the sub-problem \(B[t, t, p]\). We get a contradiction.

Now suppose that \(B[s', t', u + 1] \leq B[s', t', u]\) holds for any \(t' - s' \leq d\). Let \(s, t\) be two integers such that \(t - s = d + 1\). We claim that, for any \(0 \leq u < T\), \(B[s, t, u + 1] \leq B[s, t, u]\). Let \(k_0\) and \(k_1\) be two integers such that

\[
B[s, t, u] = \max_{0 \leq k \leq u} (B[s, t - 1, k] + B[t, t, u - k])
\]

and

\[
B[s, t, u + 1] = \max_{0 \leq k \leq u} (B[s, t - 1, k] + B[t, t, u + 1 - k])
\]

We consider the following two cases according to whether \(k_0\) equals \(k_1\). For the first case that \(k_0 = k_1\), we have

\[
B[t, t, u + 1 - k_1] \leq B[t, t, u - k_0]
\]

since \(B[t, t, u]\) is non-increasing with respect to \(u\). Hence \(B[s, t, u + 1] \leq B[s, t, u]\) in this case. Next we consider the second case that \(k_0 \neq k_1\). If \(k_1 = u + 1\), then

\[
B[s, t, u + 1] = B[s, t - 1, u + 1] + B[t, t, 0]
\]

\[
\leq B[s, t - 1, u] + B[t, t, 0]
\]

\[
\leq \max_{0 \leq k \leq u} (B[s, t - 1, k] + B[t, t, u - k])
\]

\[
= B[s, t, u]
\]

where the first inequality holds since, by induction, \(B[s, t - 1, u]\) is non-increasing with respect to \(u\). If \(k_1 \neq u + 1\), then

\[
B[s, t, u + 1] = B[s, t - 1, k_1] + B[t, t, u + 1 - k_1]
\]

\[
\leq B[s, t - 1, k_1] + B[t, t, u - k_1]
\]

\[
\leq \max_{0 \leq k \leq u} (B[s, t - 1, k] + B[t, t, u - k])
\]

\[
= B[s, t, u]
\]
where the first inequality holds since, by induction, $B[t, t, u]$ is non-increasing with respect to $u$. Therefore, for any fixed $s < t$, $B[s, t, u]$ is non-increasing with respect to $u$.

**Lemma 2.** Let $V$ be a histogram vector. Let $p_t$ be a peak index such that $V_{p_t} = \max\{V_1, V_2, \ldots, V_m\}$. Then $B[t, t, u] = \max_{(r, t) \leq p_t \leq r \leq t} \sum_{i=r+1}^{\ell} V_i$ subject to $V_t + V_r \geq u$.

**Proof.** Without loss of generality, suppose that $\ell, r'$ are two different indices such that $p_t < \ell' < r'$, $V_{\ell'} + V_{r'} \geq u$, and $B[t, t, u] = \sum_{i=\ell'+1}^{r'-1} V_i$. In this case, we set $\ell'' = p_t$. Then we have $\ell'' \leq p_t < r'$, $V_{\ell''} + V_{r'} \geq V_{\ell'} + V_{r'} \geq u$, and $\sum_{i=\ell'+1}^{r'-1} V_i \geq \sum_{i=\ell''+1}^{r'-1} V_i = B[t, t, u]$. Therefore, the lemma holds.

Based on Lemma 1 and Lemma 2, we are able to design a more efficient algorithm called **Improved-BBPS** which improves the time complexity of **Primitive-BBPS**.

The improved algorithm consists of two parts. The first part is an algorithm which computes each $B[t, t, u]$. The second part is an algorithm for computing general term $B[s, t, u]$ for $s < t$. First, we define some notations. For a histogram vector $V$ and a peak index $p_t$, we define

$$\Delta_t = \{(\ell, r) : \ell \leq p_t \leq r, \ell \neq r\}.$$  

In addition, we define the multi-set

$$\Gamma_t = \{V_{\ell} + V_r : (\ell, r) \in \Delta_t \} \cup \{0\}.$$  

Note that $|\Gamma_t| - 1 = |\Delta_t| = (2m)^t / 2$. Let $0 = u_0 \leq u_1 \leq u_2 \leq \ldots \leq u_k$ be the sorted elements of $\Gamma_t$, where $k_t = |\Gamma_t|$. For each $u_i$, let $(\ell_i, r_i) \in \Delta_t$ be its corresponding indices such that $V_{\ell_i} + V_{r_i} = u_i$. For each $(\ell_i, r_i)$, we define

$$\xi_i = \sum_{\ell_j < r_j} V_{\ell_j} \quad \text{and} \quad \xi_0 = \sum_{j=1}^{m} V_j.$$  

Lemma 2 states that, if $u_{i-1} < u \leq u_i$ for some $i$, then $B[t, t, u] = \max\{\xi_i, \xi_{i+1}, \ldots, \xi_k\}$. By Lemma 1, this implies that $B[t, t, u]$ is a non-increasing step function. Now we introduce the way to compute $B[t, t, u]$ for $0 \leq u \leq T$. First, we set

$$B[t, t, T] = \begin{cases} -\infty & \text{if } u_k \leq T \\ \max\{\xi_i, \xi_{i+1}, \ldots, \xi_k\} & \text{if } u_{i-1} \leq T \leq u_i \text{ for some } i. \end{cases}$$  

Moreover, we can compute $B[t, t, u]$ for $0 \leq u < T$ recursively as follows.

$$B[t, t, u] = \begin{cases} \max\{B[t, t, u+1], \xi_i, \ldots, \xi_k\} & \text{if } u_{i-1} < u \leq u_i \leq u_{i+1} \leq u_{i+2}, \\
B[t, t, u+1] & \text{if } u_{i-1} < u < u_i + 1 \leq u_i, \\
\xi_0 & \text{if } u = u_0. \end{cases}$$  

**Remark 1.** The time complexity of the above procedure for computing $B[t, t, u]$ for $0 \leq u \leq T$ is $O(|\Delta_t| + T)$ which improves the time complexity $O(2^t T)$ of the primitive algorithm. Thus, the time complexity of computing all terms $B[t, t, u]$ is $O(\sum_{i=1}^{n} (|\Delta_t| + T)) = O(nm^2 + nT)$. 
Lemma 3. For any \( s \neq t \),
\[
B[s, t, u] = \max_{q \in \Gamma_t} (B[s, t-1, u-q] + B[t, t, q]).
\]

Proof. Based on Lemma 1, \( B[s, t, u] \) is non-increasing for any \( s, t \). Note that, there exists some \( q' \) such that
\[
B[s, t, u] = \max_{0 \leq q \leq u} (B[s, t-1, u-q] + B[t, t, q])
= B[s, t-1, u-q'] + B[t, t, q'].
\]

Let \( u_1 \leq u_2 \leq \ldots \leq u_k \) be the sorted elements of \( \Gamma_t \). We consider two cases as follows. In the first case, we assume that \( u_{i-1} < q' \leq u_i \). In this case, we have
\[
B[s, t-1, u-u_i] + B[t, t, u_i] \leq B[s, t-1, u-q'] + B[t, t, q']
= B[s, t-1, u-q'] + B[t, t, u_i]
\leq B[s, t-1, u-u_i] + B[t, t, u_i]
\]
where the first inequality holds since \( q' \) is the optimal selection, the first equality holds since \( B[t, t, u] \) is a step function, and the second inequality holds since \( B[s, t-1, u] \) is non-increasing. In this case, we obtain that
\[
B[s, t, u] = B[s, t-1, u-u_i] + B[t, t, u_i] = \max_{q \in \Gamma_t} (B[s, t-1, u-q] + B[t, t, q]).
\]

In the second case, we assume that \( u_{i-1} < q' \leq u < u_i \). In this case, we have
\[
B[s, t-1, 0] + B[t, t, u] = B[s, t-1, 0] + B[t, t, u]
\leq B[s, t-1, u-q'] + B[t, t, q']
= B[s, t-1, u-q'] + B[t, t, u_i]
\leq B[s, t-1, 0] + B[t, t, u_i]
\]
where the first and second equalities hold since \( B[t, t, u] \) is a step function, the first inequality holds since \( q' \) is the optimal selection, and the second inequality holds since \( B[s, t-1, u] \) is non-increasing. So we have \( B[s, t, u] = B[s, t-1, 0] + B[t, t, u_i] \). Since \( u - u_i < 0 \), we have \( B[s, t, u-u_i] = B[s, t, 0] \). Therefore we obtain that
\[
B[s, t, u] = B[s, t-1, u-u_i] + B[t, t, u_i] = \max_{q \in \Gamma_t} (B[s, t-1, u-q] + B[t, t, q]).
\]
\( \square \)
By Lemma 3, we can efficiently compute $B[s, t, u]$ by searching those elements $q \in \Gamma^u_{t}$ instead of the whole elements from 0 to $u$. Now based on the above discussion, the improved version of Primitive BBPS called Improved-BBPS is presented as follows.

**Improved-BBPS** with input $V^1, \cdots, V^n \in \mathbb{R}_{\geq 0}$

arg $\max_{(t_i, t_j) \leq t} \sum_{i=1}^{n} \sum_{j=t_i+1}^{t_j-1} V_j$ subject to $\sum_{i=1}^{n} V_{i}^n + V_{i}^0 \geq T$ is obtained as follows:

1. For each histogram vector $V^t$, compute a peak index $p_t$ for $V^t$ and generate two sets $\Delta_t$ and $\Gamma_t$.
2. For $1 \leq t \leq n$, do the following:
   
   (a) List the sorted elements $u_0, u_1, \ldots, u_k$ in $\Gamma_t$ and compute their corresponding values $\xi_0, \xi_1, \xi_2, \ldots, \xi_k$.
   
   (b) Compute
   \[
   B[t, t, T] = \begin{cases} 
   -\infty & \text{if } u_k < T \\
   \max\{\xi_i, \xi_i+1, \ldots, \xi_k\} & \text{if } u_{i-1} < T \leq u_i \text{ for some } i.
   \end{cases}
   \]
   
   (c) For $0 \leq u < T$, compute
   \[
   B[t, t, u] = \begin{cases} 
   \max\{B[t, t, u+1], \xi_i, \ldots, \xi_{i-1}\} & \text{if } u_{i-1} < u \leq u_i = u_T < u + 1 \leq u_{i+1} \\
   B[t, t, u+1] & \text{if } u_{i-1} < u + 1 \leq u_i, \\
   \xi_0 & \text{if } u = u_0.
   \end{cases}
   \]
   
   and
   \[
   P[t, t, u] = \arg \max_{(t_i, t_j) \in \Delta_t} \sum_{j=t_i+1}^{t_j-1} V_j.
   \]

3. For $t = 2$ to $n$ and for $0 \leq u \leq T$, compute
   \[
   B[1, t, u] = \max_{k \in \Gamma_t^u} (B[1, t-1, u-k] + B[t, t, k])
   \]
   
   and
   \[
   P[1, t, u] = \arg \max_{k \in \Gamma_t^u} (B[1, t-1, u-k] + B[t, t, k]).
   \]

4. Set $u_j = 0$ if $j = 0$, $u_j = u_{j+1} - P[1, j+1, u_{j+1}]$ if $0 < j < n$, and $u_j = T$ if $j = n$.
5. Output $\{P[i, i, u_i - u_{i-1}] : 1 \leq i \leq n\}$.

**Remark 2.** The time complexity of Improved-BBPS for computing $B[1, n, u]$ for $0 \leq u \leq T$ is $O((\sum_{i=1}^{n} (|\Delta_i| + T)) + T \sum_{i=2}^{n} (|\Delta_i|)) = O(nm^2 + nt + nmT)$.

**Remark 3.** For the restricted boundary bin-pair selection problem, one can also apply Improved-BBPS on this problem. Since the sizes of $\Delta_t$ and $\Gamma_t$ is $O(m)$, the time complexity of Improved-BBPS for computing restricted BBPS problem is $O((\sum_{i=1}^{n} (|\Delta_i| + T)) + T \sum_{i=2}^{n} (|\Delta_i|)) = O(nm + nt + nmT)$. 

4.1 Another improved algorithm for BBPS problem

In this subsection, we provide a slightly better algorithm which improves the performance of Improved-BBPS. The main bottleneck of Improved-BBPS is the procedure for computing \(B[s,t,u]\). Recall that \(B[s,t,u] = \max_{q \in \Gamma_t^u} (B[s,t-1,u-q] + B[t,t,q])\). Improved-BBPS needs to check those \(q \in \Gamma_t^u\) for computing the maximum value. However, this procedure takes \(\Omega(|\Gamma_t|)\) steps for most \(u\). We design a new approach to improve this drawback. To obtain this goal, our key observation is that \(B[s,t,u]\) is a non-increasing step function as stated in the following lemma. First of all, we define some notations.

**Definition 2.** Assume that \(\tilde{u}^T\) is the minimum discontinuous input of the function \(B[s,t,\cdot]\) which is larger than or equal to \(T\). Let \(\Gamma_{t,j}^T\) be the subset of discontinuous inputs of \(B[s,t,\cdot]\) defined by \(\Gamma_{t,j}^T = \{u : u \leq T \text{ and } u \text{ is a discontinuous input of } B[s,t,\cdot] \} \cup \{\tilde{u}^T\}\).

**Definition 3.** Given two sets \(A,B \subseteq \mathbb{R}\), let \(A+B\) be the subset of \(A\) and \(B\) defined by \(A+B = \{u+u' : u \in A \text{ and } u' \in B\}\).

The following lemma states that \(\Gamma_{t,j}^T\) is a finite set and \(\Gamma_{t,j}^T\) is computable inductively.

**Lemma 4.** For any \(s \neq t\), \(B[s,t,u]\) is a non-increasing step function with respect to \(u\). In particular, \(\Gamma_{s,j}^T \subseteq \Gamma_{s,j-1}^T + \Gamma_{t,j}^T\).

**Proof.** The non-increasing property has been shown in Lemma 1. In addition, we have shown that \(B[t,t,u]\) is a non-increasing step function with respect to \(u\) previously. Now we prove that \(B[s,t,u]\) is a step function by induction on \(d = |t-s|\). Note that, for each function \(B[t,t,u]\), \(\Gamma_{t,j}^T\) is contained in the support set of \(\Gamma_t\) and is computable within time \(O(|\Delta_t|)\). Next, suppose that \(\Gamma_{s,j-1}^T\) and \(\Gamma_{t,j}^T\) are found. We claim that

\[
\Gamma_{s,j}^T \subseteq \Gamma_{s,j-1}^T + \Gamma_{t,j}^T = \{u' + u'' : u' \in \Gamma_{s,j-1}^T \text{ and } u'' \in \Gamma_{t,j}^T\}.
\]

Let \(0 = u_0 < u_1 < \cdots < u_k\) be the sorted elements of \(\Gamma_{s,j-1}^T + \Gamma_{t,j}^T\). If \(u = 0\), then \(B[s,t,0] = B[s,t-1,0] + B[t,t,0]\). Suppose that \(w_0 < w_1 < \cdots < w_k\) and \(0 < z_1 < \cdots < z_k\) are sorted elements of \(\Gamma_{s,j-1}^T + \Gamma_{t,j}^T\), respectively. Now, suppose that \(u_i < u \leq u_{i+1}\) for some \(i\). By Lemma 3 and the non-increasing property of \(B[s,t,u]\), there exists some \(z_p \in \Gamma_{t,j}^T\) and some \(w_j, w_{j+1} \in \Gamma_{s,j-1}^T\) such that \(w_j < u - z_p \leq w_{j+1}\) and

\[
B[s,t,u] \leq B[s,t,u_0]
= \max_{q \in \Gamma_t^u} (B[s,t-1,u-q] + B[t,t,q]) \text{ by Lemma 3}
= \max_{q \in \Gamma_t^u} (B[s,t-1,u-z_p] + B[t,t,z_p]) \text{ by Lemma 3}
= \max_{0 \leq p \leq w_{j+1} + z_p} (B[s,t-1,w_{j+1} + z_p - q] + B[t,t,q]) \text{ since } w_{j+1} \text{ is the minimum discontinuous input larger than } u - z_p
\leq \max_{0 \leq q \leq w_{j+1} + z_p} (B[s,t-1,w_{j+1} + z_p - q] + B[t,t,q])
= B[s,t,w_{j+1} + z_p] \text{ where we define } w_{j+1} + z_p = \tilde{u}' \in \Gamma_{s,j-1}^T + \Gamma_{t,j}^T
= B[s,t,\tilde{u}']\]
Since \( \tilde{u}_i < u \leq \tilde{u}_j \), we have \( \tilde{u}_{i+1} \leq \tilde{u}_j \). Since \( B[s,t,\cdot] \) is non-increasing, \( B[s,t,\tilde{u}_j] \leq B[s,t,\tilde{u}_{i+1}] \). Therefore, \( B[s,t,u] = B[s,t,\tilde{u}_{i+1}] \) for any \( \tilde{u}_i < u \leq \tilde{u}_{i+1} \). Thus, the subset \( \Gamma_{1,j}^T \) is a subset of the sumset \( \Gamma_{1,j-1}^T + \Gamma_{1,j}^T \), that is \( \Gamma_{1,j}^T \subseteq \Gamma_{1,j-1}^T + \Gamma_{1,j}^T \).

Based on Lemma 4, we have the following algorithm called Quick-BBPS for computing \( B[1,n,T] \).

**Quick-BBPS** with \( V^1, \cdots, V^n \in \mathbb{R}^m_{\geq 0} \)

\[
\arg\max_{(t_1,r_1), \ldots, (t_n,r_n) : 1 \leq i \leq n} \sum_{j=1}^{n-1} \sum_{i=1}^{r_j-1} V^j_i \] subject to \( \sum_{i=1}^{n} V^j_i + V^n_i \geq T \) is obtained as follows:

1. For \( 1 \leq t \leq n \), do the following:
   a. Compute the set \( \Gamma_{1,t}^T \) by examining the discontinuity of \( B[t,t,u] \) for those \( u \in \Gamma_r \).
   b. Keep \( B[t,t,u] \) and \( P[t,t,u] \) for \( u \in \Gamma_{1,t}^T \).
2. For \( t = 2 \) to \( n \) do the following:
   a. Compute the sumset \( \Gamma_{1,t-1}^T + \Gamma_{1,t}^T \).
   b. Find the subset \( \Gamma_{1,t}^T \) of discontinuous inputs by examining function values of inputs in the sumset \( \Gamma_{1,t-1}^T + \Gamma_{1,t}^T \).
   c. Keep \( B[1,t,u] \) and \( P[1,t,u] \) for \( u \in \Gamma_{1,t}^T \).
3. Set
   \[
u_j = \begin{cases} \min\{q \in \Gamma_{1,t}^T : q \geq T\} & \text{if } j = n \\ P[1,j+1,u_{j+1}] & \text{if } 0 < j < n \\ 0 & \text{if } j = 0 \end{cases} \]
4. Output \( \langle P[i,i,\mu_i - \nu_{i-1}] : 1 \leq i \leq n \rangle \).

**Remark 4.** The time complexity of Quick-BBPS for computing \( B[1,n,T] \) is

\[
O \left( \sum_{i=1}^{n} |\Gamma_i| + \sum_{i=1}^{n-1} (|\Gamma_{1,i}^T| \times |\Gamma_{1,i+1}^T|) \right) = O \left( nm^2 + nm^2 T \right). \]

Note that the order of the time complexity of Quick-BBPS is the same as the one of Improved-BBPS. However, in practice, the running time of Quick-BBPS is better than Improved-BBPS. We will show some experimental results to support it in the next section.

The main procedure of Quick-BBPS is to find the set \( \Gamma_{1,t}^T \) of discontinuous inputs from the sumset \( \Gamma_{1,t-1}^T + \Gamma_{1,t}^T \). However, it is not necessary to check all values in the sumset \( \Gamma_{1,t-1}^T + \Gamma_{1,t}^T \). Given \( u' \in \Gamma_{1,t-1}^T \), let us define \( q_{u',u'} = \min_{q \in \Gamma_{1,t}^T} \{ q : q \geq T - u' \} \). Instead of searching values in the whole sumset \( \Gamma_{1,t-1}^T + \Gamma_{1,t}^T \), we only need to check its subset

\[
\text{Sub} \{ \Gamma_{1,t-1}^T + \Gamma_{1,t}^T \} = \{ u' + u'' : u' \in \Gamma_{1,t-1}^T, u'' \in \Gamma_{1,t}^T, u'' \leq q_{u',u'} \}.
\]
For convenience, we call the updated algorithm $f\text{Quick-BBPS}$ that means faster $\text{Quick-BBPS}$. In the next section, we demonstrate the efficiency of $\text{Quick-BBPS}$ and $f\text{Quick-BBPS}$ by using some experimental test data.

![Histogram vectors constructed from prediction errors of shadow pixels of the image Lena.](image)

Fig. 1. The histogram vectors constructed from the prediction errors of the shadow pixels of the image Lena.

5. CONCLUDING

In this paper, we study the so-called boundary-bin-pair selection problem (BBPS problem) which is arose from the multiple-histogram-modification problem in reversible data hiding. We propose several efficient provably algorithms toward solving the BBPS problem with correctness proof as a guarantee. Since these algorithms achieve the same order of time complexity in a theoretical view, we provide some experimental result to demonstrate the efficiency of some proposed algorithms.
ACKNOWLEDGMENT

This work was supported in part by the Ministry of Science and Technology of Taiwan under contract MOST-108-2221-E-305-006.

REFERENCES


A. Generating Experimental Data from Reversible data hiding

A.1 Application of the BBPS problem: Reversible data hiding schemes based multiple histogram modification

The goal of a reversible data hiding (RDH) is to embed a secret message into an image such that the watermarked image is similar to the original one and one can losslessly recover the original image after extracting the secret message. One of methods to obtain this task is the multiple-histogram-modification (MHM) method proposed in [4]. In this section, we show an application of the BBPS problem to obtain the task of MHM-based reversible data hiding.

Here we assume that the cover pixels are listed in a sequence \((x_1, \ldots, x_N)\) where \(N\) is the total number of collected cover pixels. Next, we use a predictor (e.g., the rhombus predictor in [3]) to predict each \(x_i\) according to neighbors of \(x_i\) in the cover image. Let
\( \hat{x}_i \) be the prediction value of \( x_i \). Then, the prediction-error is computed by \( e_i = x_i - \hat{x}_i \) and the prediction-error sequence \( (e_1, \ldots, e_N) \) is obtained. In addition, for each pixel \( x_i \), its complexity measurement \( cm_i \) (e.g., the measurement defined in the equation (21) of [4]) is computed according to its neighbors. Moreover, \( M \) disjoint integer-valued subsets \( S_1, \ldots, S_M \) are selected. Then we obtain a prediction-error histogram sequence \((h_1, \ldots, h_j, \ldots, h_M)\) by defining \( h_j(e) = \# \{ 1 \leq i \leq N : e_i = e \text{ and } cm_i \in S_j \} \) for each \( j \).

Once the embedded data bit \( w \) is given, the expansion-bin pair \((a_j, b_j)\) for each histogram \( h_j \) is determined by the algorithm Quick-BBPS in the following way. Let \( e_{\min} \triangleq \min \{ e_1, \ldots, e_N \} \) and \( e_{\max} \triangleq \max \{ e_1, \ldots, e_N \} \). For each \( j \), we define the non-negative vector \( V^j \) by \( V^j_i = h_j(s + e_{\min} - 1) \) for \( 1 \leq s \leq e_{\max} - e_{\min} + 1 \). Let \( < (\ell_j, r_j) : 1 \leq j \leq M > \) be the output of Quick-BBPS with the input \((V^1, \ldots, V^M)\) and the length of data bits \( \alpha \) where \( \alpha \) will be defined later. Now, for each histogram \( h_j \), the expansion-bin pair \((a_j, b_j)\) is determined by setting \( a_j = \ell_j + e_{\min} - 1 \) and \( b_j = r_j + e_{\min} - 1 \). The information of these expansion-bin pairs \( \{(a_j, b_j) : 1 \leq j \leq M\} \) is encoded as a location map which helps to recover the original image from the watermarked image. Some fixed pixel bits of the original image are replaced by the location map. For image restoration, we should record the removal which is denoted as \( \hat{w} \). Let \( \alpha \) be the data bit string defined by \( \alpha = (w, \hat{w}) \).

For a cover pixel \( x_i \) to be processed, the modified pixel \( \hat{x}_i \) is computed according to the following procedure:

1. Compute the prediction value \( \hat{x}_i \), the prediction-error \( e_i \), and the complexity measurement \( cm_i \) of \( x_i \).

2. If \( cm_i \in S_j \), then modify the prediction-error \( e_i \) by

\[
\hat{e}_i = \begin{cases} 
  e_i & \text{if } a_j < e_i < b_j \\
  e_i + m & \text{if } e_i = b_j \\
  e_i - m & \text{if } e_i = a_j \\
  e_i + 1 & \text{if } e_i > b_j \\
  e_i - 1 & \text{if } e_i < a_j 
\end{cases}
\]

where \( m \) is a data bit of \( \alpha \) to be embedded.

3. Modify \( x_i \) to \( \hat{x}_i = \hat{x}_i + \hat{e}_i \).

The data extraction and image restoration are described as follows. For a pixel \( \hat{x}_i \) to be recovered, the original cover pixel \( x_i \) is computed in the following procedure. Due to the complexity measurement method in [4], we assume that the pixel scanning order of the extraction procedure is exactly inverse to that of the embedding procedure.

1. Compute the prediction value \( \hat{x}_i \), the marked prediction-error \( \hat{e}_i \), and the complexity measurement \( cm_i \) of \( \hat{x}_i \).

2. If \( cm_i \in S_j \), then recover the original prediction-error \( e_i \) by \( e_i = \hat{e}_i \) if \( a_j \leq \hat{e}_i \leq b_j \), \( e_i = \hat{e}_i - 1 \) if \( \hat{e}_i > b_j \), and \( e_i = \hat{e}_i + 1 \) if \( \hat{e}_i < a_j \).

3. Recover \( x_i \) by setting \( x_i = \hat{x}_i + e_i \).

In addition, the embedded data bit \( m \) can be extracted as \( m = 0 \) if \( \hat{e}_i \in \{a_j, b_j\} \), or \( m = 1 \) if \( \hat{e}_i \in \{a_j - 1, b_j + 1\} \).
A.2 Theoretical analysis on image degradation

We assume that \( \alpha \) is the binary string which consists of the secret message and the needed auxiliary information. Furthermore, we assume that \( \alpha \) is generated randomly since \( \alpha \) is usually encrypted before embedding into the image. Let \( I = (x_1,\ldots,x_{N_{\text{end}}}) \) be the sequence of pixels which are needed to be processed in the cover pixels and \( \tilde{I} = (\tilde{x}_1,\ldots,\tilde{x}_{N_{\text{end}}}) \) be the processed sequence after embedding \( \alpha \). Note that \( N_{\text{end}} \leq N \). For each \( j \), define

\[ g_j(e) = \#\{1 \leq i \leq N_{\text{end}} : e_i = e \text{ and } cm_i \in S_j \}. \]

By the analysis in [4], Li et al. showed that the expected embedding distortion of \( I \) and \( \tilde{I} \) in \( L^2 \) norm is equal to

\[ \text{ED} \triangleq \mathbb{E}(|I - \tilde{I}|_2^2) = \sum_{j=1}^{N_{\text{end}}} \mathbb{E}[(x_i - \tilde{x}_i)^2] = \sum_{j=1}^{M} \left( \frac{g_j(a_j) + g_j(b_j)}{2} + \sum_{e < a_j} h_j(e) + \sum_{e > b_j} h_j(e) \right). \]

Note that \( g_j \) is a prediction-error sub-histogram of \( h_j \) since it only counts the frequencies of prediction-errors for the first \( N_{\text{end}} \) pixels. One may expect that the shape of \( g_j \) is very similar to the one of \( h_j \) for each \( j \). Therefore, as in [4], a reasonable assumption is that, for each \( (e,j) \), \( g_j(e) \approx \frac{N_{\text{end}}}{N} h_j(e) \). By this assumption,

\[ \text{ED} \approx \frac{N_{\text{end}}}{N} \sum_{j=1}^{M} \left( \frac{h_j(a_j) + h_j(b_j)}{2} + \sum_{e < a_j} h_j(e) + \sum_{e > b_j} h_j(e) \right). \]

In addition, the embedding capacity is required to be \( |\alpha| \). So we have

\[ |\alpha| \approx \sum_{j=1}^{M} (g_j(a_j) + g_j(b_j)) \approx \frac{N_{\text{end}}}{N} \sum_{j=1}^{M} (h_j(a_j) + h_j(b_j)). \]

Then, we derive that

\[ \text{ED} \approx |\alpha| \left( \frac{\sum_{j=1}^{M} (\sum_{e < a_j} h_j(e) + \sum_{e > b_j} h_j(e))}{\sum_{j=1}^{M} (h_j(a_j) + h_j(b_j))} + \frac{1}{2} \right). \]

Now the problem to minimize the embedding distortion in \( L^2 \) norm with the embedding capacity \( |\alpha| \) can be formulated by the following multiple histograms modification problem (MHM problem) which was introduced previously in [4].

**Problem.** (Multiple Histograms Modification Problem) Given \( M \) prediction-error sub-histograms \( h_1,\ldots,h_M \), minimize

\[ \frac{\sum_{j=1}^{M} (\sum_{e < a_j} h_j(e) + \sum_{e > b_j} h_j(e))}{\sum_{j=1}^{M} (h_j(a_j) + h_j(b_j))}, \]

subject to \( \sum_{j=1}^{M} (h_j(a_j) + h_j(b_j)) \geq |\alpha| \).
Since $\sum_{j=1}^{M} (h_j(a_j) + h_j(b_j))$ is at least $|\alpha|$, we have

$$\frac{\sum_{j=1}^{M} (\sum_{e<a_j} h_j(e) + \sum_{e>b_j} h_j(e))}{\sum_{j=1}^{M} (h_j(a_j) + h_j(b_j))} \leq \frac{\sum_{j=1}^{M} (\sum_{e<a_j} h_j(e) + \sum_{e>b_j} h_j(e))}{|\alpha|}.$$ 

Based on this observation, we try to minimize the value $\sum_{j=1}^{M} (\sum_{e<a_j} h_j(e) + \sum_{e>b_j} h_j(e))$ since the minimum value of this problem provides an upper bound for embedding distortion ED. Thus, instead of the MHM problem, we consider the following modified multiple histograms modification problem (modified MHM problem).

**Problem.** (Modified Multiple Histograms Modification Problem) Given $M$ prediction-error sub-histograms $h_1, \ldots, h_M$, minimize

$$\sum_{j=1}^{M} (\sum_{e<a_j} h_j(e) + \sum_{e>b_j} h_j(e)),$$

subject to $\sum_{j=1}^{M} (h_j(a_j) + h_j(b_j)) \geq |\alpha|$.

Here we remark that the modified MHM problem is a special case of the BBPS problem. Suppose that $< (a_j, b_j) : 1 \leq j \leq M >$ is an optimal solution that witnesses the minimum value of the modified MHM problem. Then it is easily seen that $\sum_{j=1}^{M} \left( \sum_{e\leq a_j} h_j(e) \right)$ must be the maximum value subject to $\sum_{j=1}^{M} (h_j(a_j) + h_j(b_j)) \geq |\alpha|$. As a result, this maximum value and an optimal solution which witnesses it can be computed efficiently by our proposed algorithm Quick-BBPS since the modified MHM problem is a special case of the BBPS problem.

### A.3 Implementation

![Fig. 2. Shadow and blank groups in the cover image. The scanning order for pixels in the shadow (blank) group is from left to right and top to bottom.](image)

As in [4], we use the rhombus predictor and the double-layered embedding technique [3] in our implementation. For the complexity measurement, we use the measurement suggested by Li et al. (Equation (21) in [4]). For completeness, we describe their definitions as follows. The whole cover image is divided into two groups denoted as "shadow"
and "blank" as illustration in Fig. 2. The first (second) half of the secret message will be embedded into the shadow (blank) pixels. Since the embedding procedures in these two layers are the same, we only explain the embedding procedure in the shadow part. First, as shown in Fig. 2, pixels in the shadow part except for the pixels in borders are scanned from left to right and top to bottom to obtain the cover pixel sequence \((x_1, x_2, \ldots, x_N)\). In order to prevent underflow and overflow problems, pixels with value 0 and 255 will be modified to be 1 and 254, respectively. Moreover, we use a location map to record these locations. The location map is compressed losslessly to reduce the size.

Next we use the rhombus predictor to compute the prediction value of a pixel \(x_i\). As shown in Fig. 3, the rhombus prediction value of the pixel \(x_i\) defined by \(\hat{x_i} = \left\lfloor \frac{v_1 + v_2 + v_3 + v_4}{4} \right\rfloor\) where \(v_1, v_2, v_3, v_4\) are its four blank neighboring pixels. In addition, given 12 pixels \(v_2, v_3, v_4, w_1, \ldots, w_9\) as shown in Fig. 3, the complexity measurement \(c_{m_i}\) of \(x_i\) is defined by

\[
c_{m_i} = |v_2 - w_3| + |w_3 - w_6| + |v_3 - w_7| + |v_4 - w_4| + |w_4 - w_8| + |w_1 - w_2| + |w_2 - w_5| + |w_5 - w_9| + |v_4 - w_2| + |w_3 - v_3| + |v_3 - w_4| + |w_4 - w_5| + |w_6 - w_7| + |w_7 - w_8| + |w_8 - w_9|.
\]

Furthermore, the way to generate \(M\) integer-valued subsets \(V_1, \ldots, V_M\) is the same as the one in [4]. Precisely, \(M - 1\) thresholds \(t_1, t_2, \ldots, t_{M-1}\) are defined first by setting \(t_k = \min\{\frac{\#\{1 \leq i \leq N : c_{m_i} \leq q\}}{N} \geq \frac{k}{M}\}\). Then \(M\) disjoint integer-values subsets are defined by \(V_1 = [0, t_1], V_2 = [t_1 + 1, t_2], \ldots, V_{M-1} = [t_{M-2} + 1, t_{M-1}],\) and \(V_M = [t_{M-1} + 1, \infty)\).