A Quadsection Algorithm for Grammar-Based Image Compression

Morihiro Hayashida, Peiying Ruan, and Tatsuya Akutsu

Bioinformatics Center, Institute for Chemical Research, Kyoto University,
Uji, Kyoto 611-0011, Japan
{morihiro,ruan,takutsu}@kuicr.kyoto-u.ac.jp

Abstract. Grammar-based compression is to find a small grammar that generates a given data and has been well-studied in text compression. In this paper, we apply this methodology to compression of rectangular image data. We first define a context-free rectangular image grammar (CFRIG) by extending the context-free grammar. Then we propose a quadsection type algorithm by extending a bisection type algorithm for grammar-based compression of text data. We show that our proposed algorithm approximates in polynomial time the smallest CFRIG within a factor of $O(n^{4/3})$, where an input image data is of size $O(n) \times O(n)$. We also present results on computational experiments on the proposed algorithm.

Keywords: Bisection, Context-free Rectangular Image Grammar.

1 Introduction

Image compression is one of well-studied problems in data compression and image processing. Extensive studies have been done on image compression, and several methods and/or formats such as JPEG, GIF, PNG have been widely used.

Various techniques are employed in these widely used methods. JPEG was named after Joint Photographic Experts Group, and is usually lossy compression for photographic still images. Each block of size 8×8 pixels is transformed using two-dimensional DCT (Discrete Cosine Transform). The higher frequency components are more coarsely reduced by quantization. Finally, the image is compressed using Huffman coding [4]. GIF stands for Graphics Interchange Format, and is lossless compression for images with less than or equal to 256 distinct colors, based on the Lempel-Ziv algorithm [10], which is a dictionary coder that reads a sequence, constructs a dictionary dynamically, and replaces the sequence with words of the dictionary. PNG stands for Portable Network Graphics, and has been developed to replace GIF. PNG uses filtering and Deflate algorithm that is the combination of the Lempel-Ziv algorithm [10] based method and Huffman coding [4]. The compression rate of PNG is often higher than that of GIF.

T.-h. Kim et al. (Eds.): FGIT 2010, LNCS 6485, pp. 234–248, 2010.

Though compression ratios of these methods are very high for most image data, there are cases where some of these methods fail to achieve high compression ratios. Furthermore, in many existing methods, compressed data are difficult to interpret. That is, it is difficult to extract some patterns, which exist in the original image, from compressed data.

On the other hand, in text compression, extensive studies have been done on grammar-based compression [2,6,8], which is to find a small grammar generating a given string. It is useful not only for data compression but also for extraction of repetitive patterns. Therefore, it is reasonable to try to study grammar-based compression for image data. Various grammars have been proposed for producing image data [3,9]. However, to our knowledge, there was no grammar-based image compression algorithm with a guaranteed approximation ratio. Therefore, in this paper, we extend grammar-based compression for text data to image data compression. In particular, we present QUADSECTION algorithm that is obtained by extending BISECTION algorithm for text data compression [2,6]. Furthermore, we show that QUADSECTION computes in polynomial time a grammar of size $O(g^* n^{4/3})$ for a given image of size $O(n) \times O(n)$, where g^* is the size of a minimum grammar generating the given image.

The organization of the paper is as follows. In Section 2, we define a context-free rectangular image grammar by extending the context-free grammar, formalize the smallest grammar problem for image data, and prove the NP-hardness of the problem. Next, we present QUADSECTION in Section 3, analyze its approximation ratio in Section 4, and extend it to higher dimensions in Section 5. Then, we present results on some computational experiments in Section 6. Finally, we conclude with future work.

2 Context-Free Rectangular Image Grammar

Here, we define *CFRIG* (Context-Free Rectangular Image Grammar). A CFRIG is defined by a 4-tuple $(\Sigma, \Gamma, S, \Delta)$ where Σ, Γ, $S \in \Gamma$ and Δ are a set of terminal symbols, a set of nonterminal symbols, the start symbol and a set of production rules, respectively. Each terminal symbol corresponds to a label of a pixel, and is denoted by a lower-case letter. Each nonterminal symbol corresponds to a rectangular region, and is denoted by an upper-case letter. Since each nonterminal symbol is associated with a rectangular region, each nonterminal symbol is represented as $A_{n,m}$, which means that this symbol generates an image with $n \times m$ pixels (i.e., an image composed of n rows and m columns). Then, we consider the following two types of production rules

(R1) $A_{1,1} \to a$,
(R2) $A_{n,m} \to [B_{n_1,m_1}, C_{n_1,m_2}; D_{n_2,m_1}, E_{n_2,m_2}]$,
where $n_1 + n_2 = n$ and $m_1 + m_2 = m$.

The meanings of these rules are clear from Fig. 1. For a rule of type (R2), we allow subcase (R2') of $n_2 = 0$ (i.e., D_{n_2,m_1} and E_{n_2,m_2} are empty) and subcase (R2") of $m_2 = 0$ (i.e., C_{n_1,m_2} and E_{n_2,m_2} are empty). We write $A_{n,m} \to [B_{n,m_1}, C_{n,m_2}]$

and $A_{n,m} \rightarrow [B_{n_1,m}; D_{n_2,m}]$ for the former case and latter case, respectively (see also Fig. 1 (R2') and (R2")).

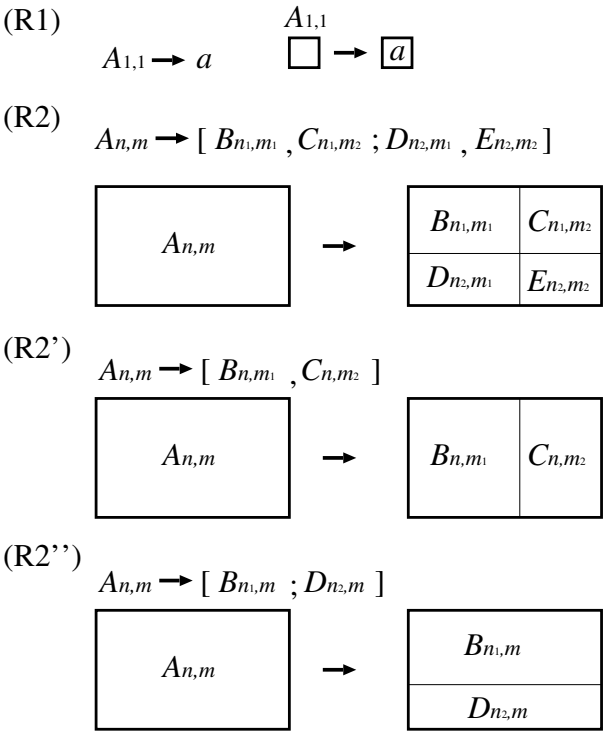

Fig. 1. Production rules for CFRIG

The *size of a grammar* is defined as the total number of symbols appearing in the right hand sides (RHSs) of production rules. From the definition of CFRIG, it is seen that only acyclic grammars are allowed in CFRIG. Furthermore, when we discuss compression algorithms, as in [2], we only consider *non-ambiguous* CFRIGs, that is, each nonterminal symbol appears in the left hand side (LHS) of exactly one rule.

Though we restricted the form of rules to (R1) and (R2), more general rules can be represented by using multiple rules of type (R1) and (R2), as shown in Fig. 2. We can show that such a transformation increases the size of the grammar only by a constant factor though we omit the proof here.

Based on the above definitions, we define the smallest grammar problem for image data is to find a smallest CFRIG which uniquely generates a given image of size $n \times m$. We can show that the smallest grammar problem for image data is NP-hard.

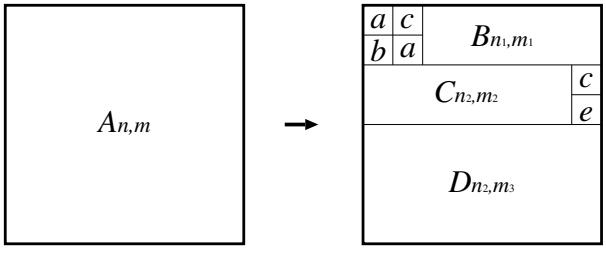

Fig. 2. This kind of rules can be transformed into CFRIG with a constant factor increase of the size

Theorem 1. *Finding the smallest CFRIG for a given image data is NP-hard.*

Proof. Since a string of length m is regarded as an image of size $1 \times m$, we can use almost the same reduction as in [2]. However, since we can only use rules of types of (R1) and (R2), we need to slightly modify the reduction.

Let $G(V, E)$ be an instance of the vertex cover problem. Let $N = |V|$ and $M = |E|$. Recall that the vertex cover problem asks whether or not there exists $W \subseteq V$ of size k such that for any edge $\{u, v\} \in E$, $u \in W$ or $v \in W$ holds.

From G and k, we construct an instance of CFRIG as follows. We map G to an image I of size $1 \times m$ by

$$I = \prod_{v_i \in V} (\#v_i|v_i\#|)^2 \prod_{\{v_i,v_j\} \in E} (\#v_i\#v_j\#|),$$

where v_i denotes a distinct terminal corresponding to each vertex, each '|' denotes a distinct terminal (delimiter), and xy means a concatenation of x and y. Let W be a vertex cover of size k. Then, we will have the following rules.

$$\begin{aligned}
D_j &\to |_j, \\
H &\to \#, \\
A_i^0 &\to v_i, \\
A_i^L &\to A_i^0 H, \\
A_i^R &\to H A_i^0, \\
A_i &\to A_i^R H, \qquad \text{if } v_i \in W
\end{aligned}$$

where $|_j$'s are introduced since each '|' denotes a distinct delimiter. It is straightforward to verify that the total size of these production rules is

$$4N \ + \ M \ + \ 1 \ + \ N \ + \ 2N \ + \ 2N \ + \ 2k \ = \ 9N + M + 2k + 1.$$

If a production rule with long RHS were allowed, we would have such a rule as

$$\begin{aligned}
S \to\ & A_1^R D_1 A_1^L D_2 A_1^R D_3 A_1^L D_4 \cdots \\
& A_1 A_2^L D_{4N+1} A_1 A_3^L D_{4N+2} \cdots,
\end{aligned}$$

for the start symbol S. It is to be noted that for each edge $\{v_i, v_j\} \in E$, a subsequence '$A_i A_j^L D_{4N+l}$' or '$A_i^R A_j D_{4N+l}$' appears in this rule. If $v_i \in W$, the former appears. Otherwise, the latter appears. It is straight-forward to see that the size of this rule is

$$8N + 3M.$$

In order to represent this rule by CFRIG, we need

$$8N + 3M - 1$$

rules of type (R2'), where each rule is of size 2. Summing up all, the total size of a grammar corresponding to W is

$$25N + 7M + 2k - 1.$$

Therefore, there exists a grammar of size $25N + 7M + 2k - 1$ that generates image I if there exists a vertex cover of size k.

On the other hand, suppose that there exists a grammar of size at most $25N+7M+2k-1$ which generates image I. Then, as in the proof of Theorem 1 in [2], we need only consider grammars having the above mentioned form. Then, we can construct a vertex cover of size k from the set of nonterminals each of which has an expansion of the form $\#v_i\#$. Therefore, the theorem holds. □

We can prove that CFRIG remains NP-hard even if $n \times n$ images are given, where the details are omitted in this paper.

3 Compression Algorithm

Our compression algorithm for image data is based on BISECTION [2,6] and is denoted by QUADSECTION here. BISECTION takes a string, and recursively decomposes the string into two smaller substrings. QUADSECTION recursively decomposes a given rectangular image $I_{n,m}$ into smaller rectangular images until each image consists of one pixel, where the same nonterminal symbol is assigned to identical rectangular images. Let $h(i)$ be 2^j for the largest integer j such that $2^j < i$, where we let $h(1) = 1$. For example, $h(2^i) = 2^{i-1}, h(2^i + 1) = 2^i$. For a rectangular image $I_{n,m}$ of size $n \times m$, $I_{[i_1:i_2],[j_1:j_2]}$ denotes the sub-rectangular image composed of i_1th - i_2th rows and j_1th - j_2th columns. The following is a pseudocode of QUADSECTION, where it is invoked with the input image $I_{n,m}$ and an empty grammar $\mathcal{G}$. QUADSECTION returns the start symbol that generates $I_{n,m}$.

```
procedure QUADSECTION(I_{n,m})
  var
    I_{n,m}: an image of size n × m;
    A_{n,m}: a nonterminal symbol uniquely assigned to I_{n,m};
    n, m, h, n_1, n_2, m_1, m_2: Integer;
```

begin
  if the same image $I'_{n,m}$ as $I_{n,m}$ has already appeared then return $A'_{n,m}$;
  if $n = 1$ and $m = 1$ then
    add $A_{1,1} \to a$ to $\mathcal{G}$ where $I_{n,m} = a$;
    return $A_{1,1}$;
  endif;
  $h := \max\{h(n), h(m)\}$; $n_1 := \min\{n, h\}$; $m_1 := \min\{m, h\}$;
  $n_2 := n - n_1$; $m_2 := m - m_1$;
  if $n_1 = n$ then
    $B_{n,m_1} :=$ QUADSECTION$(I_{[1:n],[1:m_1]})$;
    $C_{n,m_2} :=$ QUADSECTION$(I_{[1:n],[m_1+1:m]})$;
    add $A_{n,m} \to [B_{n,m_1}, C_{n,m_2}]$ to $\mathcal{G}$;
  else if $m_1 = m$ then
    $B_{n_1,m} :=$ QUADSECTION$(I_{[1:n_1],[1:m]})$;
    $D_{n_2,m} :=$ QUADSECTION$(I_{[n_1+1:n],[1:m]})$;
    add $A_{n,m} \to [B_{n_1,m}; D_{n_2,m}]$ to $\mathcal{G}$;
  else
    $B_{n_1,m_1} :=$ QUADSECTION$(I_{[1:n_1],[1:m_1]})$;
    $C_{n_1,m_2} :=$ QUADSECTION$(I_{[1:n_1],[m_1+1:m]})$;
    $D_{n_2,m_1} :=$ QUADSECTION$(I_{[n_1+1:n],[1:m_1]})$;
    $E_{n_2,m_2} :=$ QUADSECTION$(I_{[n_1+1:n],[m_1+1:m]})$;
    add $A_{n,m} \to [B_{n_1,m_1}, C_{n_1,m_2}; D_{n_2,m_1}, E_{n_2,m_2}]$ to $\mathcal{G}$;
  endif;
  return $A_{n,m}$;
end.

It is straight-forward to see that QUADSECTION works in polynomial time.

4 Analysis

In the following, we assume without loss of generality (w.l.o.g.) that $n \geq m$. If we consider images with $n \times 1$ pixels, CFRIG corresponds to CFG and thus the lower bounds on the approximation ratio on compression in [2] holds for CFRIG. In the same way, the lower bound for BISECTION (Theorem 5 in [2]) holds also for QUADSECTION.

Proposition 1. *The approximation ratio of* QUADSECTION *is* $\Omega(\sqrt{n}/\log n)$.

Similarly, we obtain the following proposition.

Proposition 2. *The smallest CFRIG that generates an image of size* $n \times m$ *has size* $\Omega(\log n)$.

In order to analyze the upper bound of QUADSECTION, we first establish mk Lemma [2] for CFRIG, where we use g instead of m to denote the size of a grammar here.

Lemma 1. *If the input image data $I_{n,m}$ is generated by an CFRIG of size g, $I_{n,m}$ contains at most $2ngk$ distinct sub-images of size $k \times h$, where we assume w.l.o.g. that $k \geq h$.*

Proof. Let $H_{k,h}$ be a sub-image of size $k \times h$ of $I_{n,m}$. If $k = h = 1$, $H_{1,1}$ is represented by RHS of a rule of type (R1). Otherwise, since CFRIG is an acyclic grammar, there exists a rule of type (R2), $A_{n',m'} \rightarrow [B_{n_1,m_1}, C_{n_1,m_2}; D_{n_2,m_1}, E_{n_2,m_2}]$, that $I_{A_{n',m'}}$ contains $H_{k,h}$ and none of $I_{B_{n_1,m_1}}$, $I_{C_{n_1,m_2}}$, $I_{D_{n_2,m_1}}$ and $I_{E_{n_2,m_2}}$ contains $H_{k,h}$, where I_A for a nonterminal A denotes the expansion image of A. (See Fig. 3.) We assume w.l.o.g that a part of $H_{k,h}$ is included in $I_{B_{n_1,m_1}}$. $H_{k,h}$ is one of at most $km_1 + hn_1 \leq 2nk$ sub-images. Therefore, $I_{n,m}$ contains at most $2ngk$ distinct sub-images of size $k \times h$. □

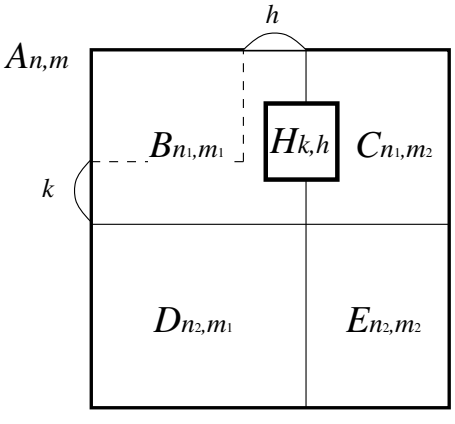

Fig. 3. Proof of Lemma 1

Theorem 2. QUADSECTION *computes in polynomial time an CFRIG of size $O(g^* n^{4/3})$ for a given image $I_{n,m}$ of size $n \times m$ $(n \geq m)$, where g^* is the size of the smallest CFRIG generating $I_{n,m}$.*

Proof. We prove the theorem only for the case that $n = m = 2^l$ holds for some integer l. Modification of the proof for the other cases is straight-forward.

The number of sub-images that are generated by recursive calls of depth at most k is bounded by

$$1 + 4 + 4^2 + \cdots + 4^k.$$

On the other hand, the number of distinct sub-images that are generated by recursive calls of depth at least $k + 1$ is bounded by

$$2 \sum_{i=0}^{h} g^* n 2^i$$

from Lemma 1, where $h = \log n - k$. Therefore, the number of production rules generated by QUADSECTION is

$$O(4^k + g^* n 2^{\log n - k}).$$

By letting $4^k = n2^{\log n - k}$, we have

$$k = \frac{2}{3} \log n.$$

Therefore, the number of production rules generated by QUADSECTION is

$$O(n^{4/3} + g^* n^{4/3}) = O(g^* n^{4/3}). \qquad \square$$

It is to be noted that the grammar may have size $O(n^2)$ in the worst case and thus the above approximation ratio is meaningful.

5 Extension to *d*-Dimensional Volume Data

Although we have defined a grammar for two-dimensional image data, we can define another grammar, *d-CFRVG* (Context-Free d-dimensional Rectangular Volume Grammar), for d-dimensional volume data ($d \geq 3$) in a similar way. A d-CFRVG is defined by a 4-tuple $(\Sigma, \Gamma, S, \Delta)$ where Σ, Γ, $S \in \Gamma$ and Δ are a set of terminal symbols, a set of nonterminal symbols, the start symbol and a set of production rules, respectively. Each terminal symbol corresponds to a label of a d-dimensional voxel, and is denoted by a lower-case letter. Each nonterminal symbol corresponds to a d-dimensional rectangular region, and is denoted by an upper-case letter. Since each nonterminal symbol is associated with a d-dimensional rectangular region, each nonterminal symbol is represented as $A_{n^{(1)},\cdots,n^{(d)}}$, which means that this symbol generates a volume with $n^{(1)} \times \cdots \times n^{(d)}$ units. Then, production rules are as follows.

(R1) $A_{\underbrace{1,\cdots,1}_{d}} \to a$,

(R2) $A_{n^{(1)},\cdots,n^{(d)}} \to \left[B_{n_1^{(1)},n_{j_2}^{(2)},\cdots,n_{j_d}^{(d)}}, C_{n_2^{(1)},n_{j_2}^{(2)},\cdots,n_{j_d}^{(d)}} \right]_{(j_2,\cdots,j_d)\in\{1,2\}^{d-1}}$,
where $n_1^{(i)} + n_2^{(i)} = n^{(i)}$ for $i = 1, \cdots, d$.

It should be noted that d-CFRVG is also an acyclic grammar. For d-CFRVG, the following lemma holds as well as CFRIG.

Lemma 2. *If the input volume data $V_{n^{(1)},\cdots,n^{(d)}}$, where $n = n^{(1)} \geq \cdots \geq n^{(d)}$, is generated by an d-CFRVG of size g, $V_{n^{(1)},\cdots,n^{(d)}}$ contains at most $dkgn^{d-1}$ distinct sub-images of size $k^{(1)} \times \cdots \times k^{(d)}$, where we assume w.l.o.g that $k = k^{(1)} \geq \cdots \geq k^{(d)}$.*

QUADSECTION can be extended in a straight-manner to compression of d-dimensional volumes, and is called HYPERSECTION.

Theorem 3. HYPERSECTION *computes in polynomial time a d-CFRVG of size* $O(g^* n^{d^2/(d+1)})$ *for a given volume* $V_{n^{(1)},\cdots,n^{(d)}}$ *of size* $n^{(1)} \times \cdots \times n^{(d)}$*, where* g^* *is the size of the smallest d-CFRVG generating* $V_{n^{(1)},\cdots,n^{(d)}}$*.*

Proof. We prove the theorem only for the case that $n = n^{(1)} = \cdots = n^{(d)} = 2^l$ holds for some integer l as well as Theorem 2.

The number of sub-volumes that are generated by recursive calls of depth at most k is bounded by

$$1 + 2^d + 2^{2d} + \cdots + 2^{kd}.$$

On the other hand,the number of distinct sub-volumes that are generated by recursive calls of depth at least $k+1$ is bounded by

$$d \sum_{i=0}^{h} g^* n^{d-1} 2^i$$

from Lemma 2, where $h = \log n - k$. Therefore, the number of production rules generated by HYPERSECTION is

$$O(2^{kd} + g^* n^{d-1} 2^{\log n - k}).$$

By letting $2^{kd} = n^{d-1} 2^{\log n - k}$, we have

$$k = \frac{d}{d+1} \log n.$$

Therefore, the number of production rules generated by HYPERSECTION is

$$O(n^{d^2/(d+1)} + g^* n^{d^2/(d+1)}) = O(g^* n^{d^2/(d+1)}).$$

□

It is to be noted that the grammar may have size $O(n^d)$ in the worst case and thus the above approximation ratio is meaningful.

6 Computational Experiments

We implemented QUADSECTION and applied it to several images. In our implementation, input raw images are given in PGM (Portable GrayMap) format or PPM (Portable PixMap) format. An image in PGM and PPM format consists of the format type, width, height, maximum pixel value, and pixel values in raster scan order. Each pixel value is represented either by ascii codes or by binary values according to the format type. Since the file size in ascii format depends on the pixel values, we used only binary format. A pixel value in PGM format is stored in 8 bits, and that in PPM format is stored in 24 bits, where each color of red, green and blue is represented in 8 bits.

The implemented version of QUADSECTION generates CFRIG $(\Sigma, \Gamma, S, \Delta)$ from a given image in PGM or PPM format, and the grammar is stored in newly introduced QSN format as follows (see Table 1). The set of rules Δ is divided into four sets of rules, $\Delta^{(\mathrm{R2})}$, $\Delta^{(\mathrm{R2'})}$, $\Delta^{(\mathrm{R2''})}$ and $\Delta^{(\mathrm{R1})}$, corresponding to the types of rules, (R2), (R2'), (R2") and (R1) respectively. Let Δ_i denote the i-th rule of Δ. Then, $\Delta_1 = \Delta_1^{(\mathrm{R2})}$, $\Delta_{|\Delta^{(\mathrm{R2})}|+1} = \Delta_1^{(\mathrm{R2'})}$ and $\Delta_{|\Delta|} = \Delta_{|\Delta^{(\mathrm{R1})}|}^{(\mathrm{R1})}$. In particular, we suppose that LHS of either $\Delta_1^{(\mathrm{R2})}$, $\Delta_1^{(\mathrm{R2'})}$, $\Delta_1^{(\mathrm{R2''})}$ or $\Delta_1^{(\mathrm{R1})}$ is the start symbol S. The nonterminal symbol of LHS of Δ_i is replaced with the number $i-1$. Thus, each nonterminal symbol is represented with $\lceil \log |\Delta| \rceil$ bits number. Each terminal symbol, that is a pixel value, is represented with 8 bits for PGM format and 24 bits for PPM format. In Table 1, $\mathrm{RHS}(\Delta_i)$ denotes the nonterminal and terminal symbols appeared in RHS of Δ_i. In QSN format, the numbers corresponding to symbols contained in $\mathrm{RHS}(\Delta_i)$ are stored sequentially in order. In the case of the black-color image of size 512×512, QUADSECTION generates the following 10 rules: $S = A_{512,512} \to [A_{256,256}, A_{256,256}; A_{256,256}, A_{256,256}]$, $\cdots$, $A_{2,2} \to [A_{1,1}, A_{1,1}; A_{1,1}, A_{1,1}]$, $A_{1,1} \to 0$. Since $\lceil \log |\Delta| \rceil = \lceil \log 10 \rceil = 4$, $|\Delta^{(\mathrm{R2})}| = 9$ and $|\Delta^{(\mathrm{R1})}| = 1$, the compressed file size in QSN format is $8 + 1 + 8 + 2 + \lceil \log 10 \rceil \cdot (3 + 4 \cdot 9) + 8 \cdot 1 = 183$ bits. It should be noted that the actual file size is $\lceil 183/8 \rceil = 23$ bytes because files are created in a storage in terms of bytes.

Table 1. QSN format for CFRIG $(\Sigma, \Gamma, S, \Delta)$

# bits	contents
8	maximum pixel value
1	0 if PGM, 1 if PPM
8	$\lvert\Delta\rvert$
2	rule type including the start symbol
$\lceil \log \lvert\Delta\rvert \rceil$	$\lvert\Delta^{(\mathrm{R2})}\rvert$
$\lceil \log \lvert\Delta\rvert \rceil$	$\lvert\Delta^{(\mathrm{R2'})}\rvert$
$\lceil \log \lvert\Delta\rvert \rceil$	$\lvert\Delta^{(\mathrm{R2''})}\rvert$
$4\lceil \log \lvert\Delta\rvert \rceil$ /rule	$\mathrm{RHS}(\Delta_i^{(\mathrm{R2})})$ $(i = 1, \cdots, \lvert\Delta^{(\mathrm{R2})}\rvert)$
$2\lceil \log \lvert\Delta\rvert \rceil$ /rule	$\mathrm{RHS}(\Delta_i^{(\mathrm{R2'})})$ $(i = 1, \cdots, \lvert\Delta^{(\mathrm{R2'})}\rvert)$
$2\lceil \log \lvert\Delta\rvert \rceil$ /rule	$\mathrm{RHS}(\Delta_i^{(\mathrm{R2''})})$ $(i = 1, \cdots, \lvert\Delta^{(\mathrm{R2''})}\rvert)$
8 (or 24) /rule	$\mathrm{RHS}(\Delta_i^{(\mathrm{R1})})$ $(i = 1, \cdots, \lvert\Delta^{(\mathrm{R1})}\rvert)$

In order to evaluate the compression ability of QUADSECTION, we compared the following image file formats, PNG (Portable Network Graphics), GIF (Graphics Interchange Format), JPEG (Joint Photographic Experts Group) and IFS (Iterated Function Systems) [7]. Both of GIF and PNG use lossless compression algorithms as well as QUADSECTION. It should be noted that GIF is not

able to deal with more than 256 distinct colors. If an image has more than 256 distinct colors, GIF ignores less frequently used colors. In order to enhance compression rates, PNG firstly employs filtering, which replaces the color of each pixel with the difference of colors between adjacent pixels. It makes use of the characteristics that colors of adjacent pixels are often very close in images. IFS is a quadtree-based fractal image coder/decoder, and the software called Mars implemented by Polvere [7] is available from `http://inls.ucsd.edu/~fisher/Fractals/Mars-1.0.tar.gz`.

We examined the following images, 'black', 'cross' (Fig. 4), 'cross2' (Fig. 5), 'hilbert' (Fig. 6), 'lena' (Fig. 7), and 'lena2' (Fig. 8). The image of black is considered before, consists of black color pixels, and the size is 512×512. The image of cross consists of 3 distinct colors, and the size is 512×512. The image of cross2 is the left-top part of that of cross, and the size is 234×345. The image of hilbert is a 6-th order Hilbert curve. The Hilbert curve is known as one of fractal diagrams, can be formed using the following rules. $S \rightarrow A$, $A \rightarrow LBFRAFARFBL$, $B \rightarrow RAFLBFBLFAR$, where S is the start symbol, L means 'turn left at a right angle', R means 'turn right at a right angle', and F means 'draw forward' [3]. The image of lena was transformed from the full color image file '4.2.04.tiff' provided on the USC-SIPI Image Database (`http://sipi.usc.edu/database/`) to PGM format using a tool of ImageMagick (`http://www.imagemagick.org/`), 'convert', and the size is the same as the original one, 512×512. The image of lena2 was transformed from '4.2.04.tiff' to the binary image in PGM format using '-colors 2' option of the tool 'convert'. We also used the tool 'convert' in order to transform the above images from PGM format to PNG, GIF or JPEG format.

Table 2. Results on the compression sizes (byte) in QSN, PNG, GIF, JPEG and IFS formats for several images

image	QSN	PNG	GIF	JPEG	IFS
black	23	265	828	1185	1544
cross	57	1688	8645	2392	1544
cross2	146	1031	2797	885	1114
hilbert	181	1196	11083	99172	40582
lena	697387	223614	264340	65338	15825
lena2	33524	17859	15501	22888	2868

Table 2 shows the results on the compression sizes (byte) in QSN, PNG, GIF, JPEG, and IFS formats for the images. The uncompressed size of black, cross, hilbert, lena, and lena2, in PGM format is 262159 bytes, and that of cross2 is 80745 bytes, respectively. For the images having symmetric and geometric patterns, that is black, cross, and hilbert, QUADSECTION was able to compress them better than other image compression methods. However, for the image of cross2, the compression size in QSN format was larger than that of cross

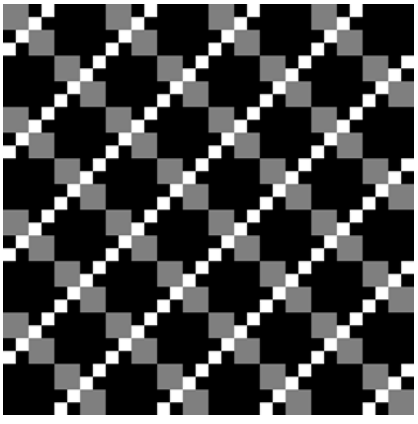

Fig. 4. Image of cross with size 512×512 and 3 distinct colors

Fig. 5. Image of cross2 with size 234×345. cross2 is the left-top part of cross

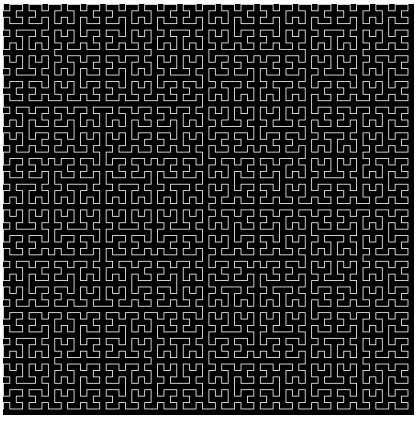

Fig. 6. Binary image of hilbert with size 512×512

Fig. 7. Gray-scale image of lena with size 512×512

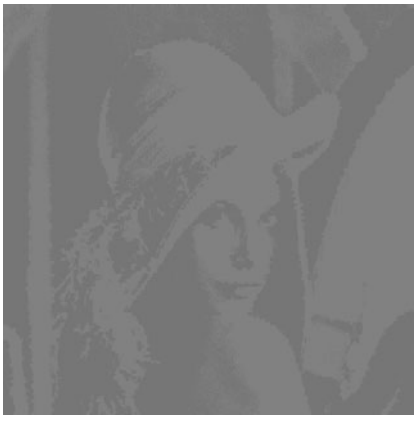

Fig. 8. Binary image of lena2 with size 512×512

although the image size of cross2 is smaller than cross, and the compression sizes of cross2 in other formats were smaller than those of cross. It is considered that the image size of cross2, 234×345, is not a power of two, some sub-images corresponding to nonterminal symbols had various sizes, and it increased the number of rules. It should be noted that IFS did not compress the hilbert image well. It is considered because fractal image compression methods find sub-images whose contraction image is similar to a sub-image. In contrast, the results of IFS for the photographic images, lena and lena2, were good. For the image of lena, which is a gray-scale photographic image, the compression size in QSN format was larger than the size of the raw image and the compression sizes in other image formats. It is considered that QUADSECTION could not compress it well because the image is not symmetric and has many colors. The compression size

of lena2 in QSN format was smaller than that of lena and the size of lena2 in PGM format. The rate of the compression size in QSN format to that in PNG or GIF format decreased from about 3 for lena to about 2 for lena2. This result suggests that QUADSECTION is still useful for compression of non-symmetric binary image data.

7 Conclusion

We have proposed a grammar-based image compression algorithm, QUADSECTION, by extending the BISECTION algorithm for text data compression. For that purpose, we defined CFRIG, which is an extension of the context-free grammar for strings. Since QUADSECTION is quite simple, there may exist the same or similar methods. However, the most important contribution of this paper is that it gives a guaranteed approximation ratio to the smallest grammar, which might stimulate further studies of improvements and extensions of grammar-based image compression.

Our proposed method has some similarity with *fractal image compression* [1,5]. Fractal image compression is based on a fact that parts of an image are often similar to other parts of the same image, and makes extensive use of these similarities. However, fractal image compression is usually computationally expensive. Furthermore, fractal image compression is usually lossy (i.e., it discards some information in the original image data). Different from fractal image compression, our proposed method is lossless and efficient, and has a guaranteed approximation ratio.

In this paper, we proposed a direct approach for grammar-based compression of image data. However, we can consider an indirect approach to compress image data in which a given image is first transformed into a string by means of raster scan and then is compressed using grammar-based compression algorithms for text data. Though it is difficult to extract patterns by such an approach, it might lead to better compression performances or better approximation ratios. Therefore, such an approach should be studied.

As shown in Section 6, for some types of binary or ternary image data, QUADSECTION had better performances than other standard image compression methods. However, in general, it is not better than those methods. In particular, QUADSECTION is not very useful for compression of gray-scale images or color images because QUADSECTION makes use of exactly repetitive patterns. Therefore, development of grammar-based compression methods for gray-scale images and color images is left as future work.

Acknowledgments. This work was partially supported by Grants-in-Aid #19650053 and #21700323 from the Ministry of Education, Culture, Sports, Science and Technology, Japan.

References

1. Bamsley, M.F., Demko, S.: Iterated function systems and the global construction of fractals. Proc. of Royal Society of London A399, 243–275 (1985)
2. Charikar, M., Lehman, E., Liu, D., Panigrahy, R., Prabhakaran, M., Sahai, A., Shelat, A.: The smallest grammar problem. IEEE Transactions on Information Theory 51, 2554–2576 (2005)
3. Drewes, F.: Grammatical picture generation: A tree-based approach. Springer, Heidelberg (2006)
4. Huffman, D.A.: A method for the construction of minimum-redundancy codes. In: Proceedings of the Institute of Radio Engineers, vol. 40, pp. 1098–1101 (1952)
5. Jacquin, A.E.: Image coding based on a fractal theory of iterated contractive image transformations. IEEE Transactions on Image Processing 1, 18–30 (1992)
6. Kieffer, J.C., Yang, E.H.: Grammar-based codes: A new class of universal lossless source codes. IEEE Transactions on Information Theory 46, 737–754 (2000)
7. Polvere, M., Nappi, M.: A feature vector technique for fast fractal image coding. Tech. rep., University of Salerno (1998)
8. Rytter, W.: Application of lempel-ziv factorization to the approximation of grammar-based compression. Theoretical Computer Science 302, 211–222 (2003)
9. Subramanian, K.G., Ali, R.M., Geethalakshmi, M., Nagar, A.K.: Pure 2d picture grammars and languages. Discrete Applied Mathematics 157, 3401–3411 (2009)
10. Ziv, J., Lempel, A.: Compression of individual sequences via variable-rate coding. IEEE Transactions on Information Theory 24, 530–536 (1978)