An Efficient Decoding Algorithm for WFA

Raghavendra Udupa U   Vinayaka D Pandit   Tanveer A Faruquie
IBM India Research Lab.
New Delhi 110016
India
{uraghave,pvinayak,ftanveer}@in.ibm.com

Abstract

Several promising image/video data compression techniques that explicitly exploit self-similarity in images and videos have been proposed in the recent past. While most of these fractal techniques are variants and/or enhancements of Jacquin's Iterated Function Systems (IFS), the Weighted Finite Automata (WFA) techniques (introduced by Culik and Kari) are based on automata theory and are known to yield better compression rates than the IFS techniques in practice. The algorithm proposed by Culik and Kari for decoding WFA is simple but inefficient. It is very slow and has very high memory requirements. In this paper we give a simple decoding algorithm which is not only faster but also requires considerably less memory.

1 Introduction

The emergence of Internet and Multimedia technologies has led to the development of data intensive multimedia applications. Uncompressed multimedia data requires considerable storage capacity and transmission bandwidth which may not be available in many legacy networks or may not be available in network of limited bandwith. There is a need for efficient compression techniques which yield good compression ratios. Several image compression paradigms have been proposed and studied [9]. In the past few years, fractal image compression has shown a lot of promise. Since the publication of Jacquin's path-breaking work on automated fractal image coding a decade ago [5], numerous variants and enhancements of the Iterated Functions Systems (*IFS*) have been proposed [8], [7]. The IFS based techniques are all based on the representation of an image by a finite set of contractive mappings on the space of images [10]. The fixed point of these contractive mappings is an approximation to the original image. Culik and Kari have proposed a fractal coding technique which is based on automata theory [1]. In contrast to the IFS based techniques, their technique finds a weighted finite automaton (*WFA*) representation of the image. Such a representation is more compact than the original image, and hence very high compression ratios can be achieved. The process of computing the WFA representation of an image is called *encoding* the image. The conjugate of this is the *decoding* of the WFA where the image is generated from the WFA. In practice, an image is usually encoded once but decoded many times. High performance processors can be used for encoding, while decoding should be possible even with the simplest equipment having small memory and limited processing power. This scenario is particularly relevant when more and more people want to view images and video on small hand-held devices which have memory and processing power constraints. Culik and Kari have given a decoding algorithm for WFA in [1] which is very simple but requires a lot of memory and is slow. In this paper we present a new algorithm for WFA decoding. Our algorithm requires significantly less memory and is faster than the algorithm proposed by Culik and Kari while retaining the simplicity of original algorithm.

2 Weighted Finite Automata

A WFA is a 5-tuple $A = (Q, \Sigma, W, I, F)$ ([1]) where

- Q is a finite set of *states*,
- Σ is a finite alphabet,
- $W_a : Q \times Q \rightarrow \Re$, for each $a \in \Sigma$, is the matrix of weights of transitions with label a,
- $I : Q \rightarrow \Re$ is the *Initial Distribution*, and
- $F : Q \rightarrow \Re$ is the *Final Distribution*.

A WFA A computes a function $f_A : \Sigma^* \rightarrow \Re$ as follows:

$$f_A(a_1 \dots a_k) = IW_{a_1} \dots W_{a_k} F$$

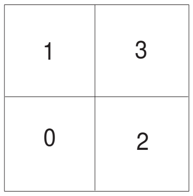

Figure 1: Quadrant Labelling

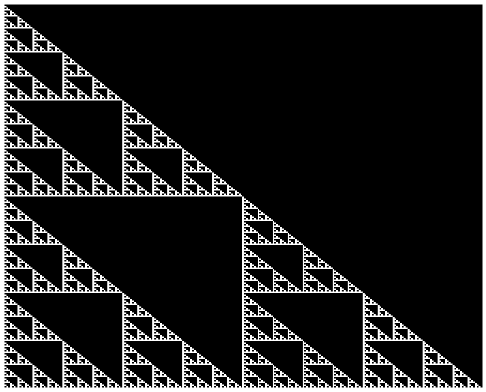

Figure 2: An image computed by a WFA

for each $k \geq 0$ and $a_1 \ldots a_k \in \Sigma^*$.

In case of images, the alphabet Σ is the set $\{0, 1, 2, 3\}$, where each symbol corresponds to a quadrant of a sub-square (see Figure 1). Every sub-square of the image can be addressed by a unique string of these symbols. If w is the label of a sub-square, then the labels of its quadrants are $w0, w1, w2, w3$. Thus, the label of the entire image is ϵ, i.e., the null symbol and the labels of the four quadrants of the image are $0, 1, 2, 3$. If the image is of resolution $2^n \times 2^n$, then each pixel of the image can be addressed by a unique string of length n. Figure 2 shows an example of an image computed by WFA. Notice that the quadrants $0, 1, 2$ of the image are scaled copies of the image. The WFA that computes this image is the following: $W_0 = W_1 = W_2 = [1], W_3 = [0], I = [1], F = [1]$. More examples of images computed by WFA with very few states can be found in [1]. The same paper also describes an encoding algorithm which can generate a WFA that approximately computes a given image.

A WFA is an *average preserving* WFA, if for every $w \in \Sigma^*$, $f_A(w) = \frac{1}{|\Sigma|}\sum_{a \in \Sigma} f(wa)$. The WFA returned by the encoding algorithm of Culik and Kari [1] is an *average preserving* WFA. It can be easily seen, from the definition of WFA, that the function computed by it is *multiresolutional*. In other words, once we obtain a WFA representation of an image, we can decode the image at any desired resolution.

2.1 Efficient Decoding of WFA

We now describe a very efficient decoding algorithm for WFA. Culik and Kari observed that a WFA A defines a multiresolutional function ψ_p for every state p of the WFA:

$$\psi_p(\epsilon) = F(p),$$

$$\psi_p(aw) = \sum_{q \in Q} W_a(p, q)\psi_q(w), \ a \in \Sigma, \ w \in \Sigma^*$$

or equivalently,

$$\psi_p(a_1 \ldots a_k) = (W_{a_1} \ldots W_{a_k} F)_p \, , \ a_1 \ldots a_k \in \Sigma^*.$$

The function computed by A is a linear combination of these functions:

$$f_A(w) = \sum_{q \in Q} I(q)\psi_q(w).$$

To compute the image f_A at any given resolution, we could use these mutually recursive relationships among the ψ_ps. This is the gist of the Culik and Kari algorithm. It can be easily shown that this algorithm computes $f_A(w)$ for all $w \in \Sigma^n$, $n > 0$, in $O(m^2 t^n + m t^n) = O(m^2 t^n)$ time at the expense of $O(m t^n)$ space. Here $t = |\Sigma|$ and $m = |Q|$.

We observe that a WFA A defines another multiresolutional function, Φ_p, for every state p of the WFA:

$$\Phi_p(\epsilon) = I(p),$$

$$\Phi_p(wa) = \sum_{q \in Q} W_a(q, p)\Phi_q(w), \ a \in \Sigma, \ w \in \Sigma^*.$$

or equivalently,

$$\Phi_p(a_1 \ldots a_k) = (I W_{a_1} \ldots W_{a_k})_p \, , \ a_1 \ldots a_k \in \Sigma^*.$$

It is easily seen that the function computed by A is a linear combination of these functions:

$$f_A(w) = \sum_{q \in Q} \Phi_q(w)F(q).$$

A more important observation is the following. Suppose we know $\Phi_p(u)$ and $\psi_p(v)$, for all $p \in Q$, where $u, v \in \Sigma^*$. We can compute $f_A(uv)$ as

$$f_A(uv) = \sum_{p \in Q} \Phi_p(u)\psi_p(v).$$

Our algorithm makes use of this observation to reduce space and time.

Algorithm DECODE_WFA
Input: A WFA A, and $n > 0$.
Output: $f_A(w)$ for each w in Σ^n.

- For all $p \in Q$ let $\Phi_p = I(p)$ and $\psi_p = F(p)$.
- For $i = 1, 2, \ldots, n_1$ do the next step.
- For all $p \in Q,\ w \in \Sigma^{i-1}$ and $a \in \Sigma$ compute
$$\Phi_p(wa) = \sum_{q \in Q} W_a(q,p)\Phi_q(w).$$
- For $i = 1, 2, \ldots, n_2$ do the next step.
- For all $p \in Q,\ w \in \Sigma^{i-1}$ and $a \in \Sigma$ compute
$$\psi_p(aw) = \sum_{q \in Q} W_a(p,q)\psi_q(w).$$
- For each $u \in \Sigma^{n_1},\ v \in \Sigma^{n_2}$ compute
$$f_A(uv) = \sum_{q \in Q} \Phi_q(u)\psi_q(v).$$

In the above algorithm, $n_1 + n_2 = n$. Notice that the Culik and Kari algorithm is a special case of our algorithm with $n_1 = 0$ and $n_2 = n$.

3 Analysis

Let $t = |\Sigma|$. For the sake of simplicity we assume that $n = 2l$. It is easy to see that to compute $\Phi_p(u)$ for all $u \in \Sigma^{n_1}$, our algorithm requires $O(m^2 t^{n_1})$ time and $O(mt^{n_1})$ space. Similarly, the computation of $\psi_p(v)$ for all $v \in \Sigma^{n_2}$ requires $O(m^2 t^{n_2})$ time and $O(mt^{n_2})$ space. Finally, the computation of $f_A(uv)$ for all $uv = w \in \Sigma^n$ in the last step requires $O(mt^n)$ time. Therefore, the time taken by the algorithm is $O(m^2 t^{n_1} + m^2 t^{n_2} + mt^n)$ and the space taken is $O(mt^{n_1} + mt^{n_2})$. To minimize both time and space, we should choose $n_1 = n_2 = n/2 = l$. Thus the time complexity of our algorithm is $O(m^2 t^l + m^2 t^l + mt^{2l}) = O(m^2 t^l + mt^{2l})$ and the space complexity is $O(mt^l + mt^l) = O(mt^l)$ while the corresponding figures for the Culik and Kari algorithm are $O(m^2 t^{2l})$ and $O(mt^{2l})$ respectively. For each pixel our algorithm takes $O(m^2/t^l + m)$ time and $O(m/t^l)$ space while the Culik and Kari algorithm takes $O(m^2)$ time and $O(m)$ space. In a typical image compression application, $t = 4$, $m > 25$, and $l \geq 4$. Therefore, it is clear that our algorithm reduces drastically both time and space needed for decoding the WFA.

Figure 3: Face

Figure 4: Tree and building

4 Experiments and Results

We generated WFA for several images (of different sizes) and decoded them at different resolutions. For each of the test images, we generated several W-FAs that approximate that image. Some images used in our experiments are shown in figure 3 and 4. We carried out our experiments on IBM's RS-6000 (single processor RISC) machine with 256 Megabytes of memory.

The images used were of size 128 X 128. Figure 5 shows the time required to decode a particular WFA (of 63 states) computed on image shown in figure 3 at different resolutions. In the plots, time and memory are plotted as functions of n, instead of the resolution $2^n \times 2^n$. Figure 6 shows the amount of memory required for decoding the same WFA at different resolutions. We see in figure 6 that the system could not allocate the memory required by Culik and Kari algorithm for resolutions 512 x 512 and higher since the requirement exceeded the total available memory on the system (256 Mb). This also explains why we could not measure the time needed for decoding the WFA using Culik and Kari algorithm at those resolutions. In figure 5 both the plots have two major bumps. The first bump occurs when the data (i.e., the images representing different states of the WFA) does not fit into the cache of the system. The second bump occurs when the data does not fit into the main memory. From these plots it is evident that the proposed algorithm gives a better performance in terms of both speed and memory.

We encoded the image shown in figure 4 with WFA of different sizes (by the size of a WFA we mean the number of states in the WFA). Figure 7 shows the time required to decode these WFA at resolution 256 x 256. The corresponding memory requirements are shown in figure 8. Again we see that our algorithm requires less time and memory than Culik and Kari algorithm.

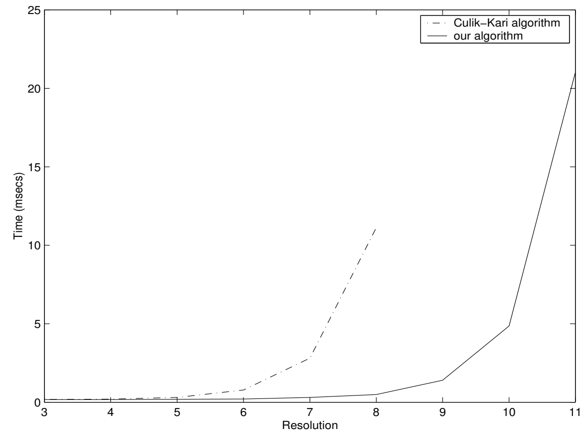

Figure 5: Decoding time as a function of resolution

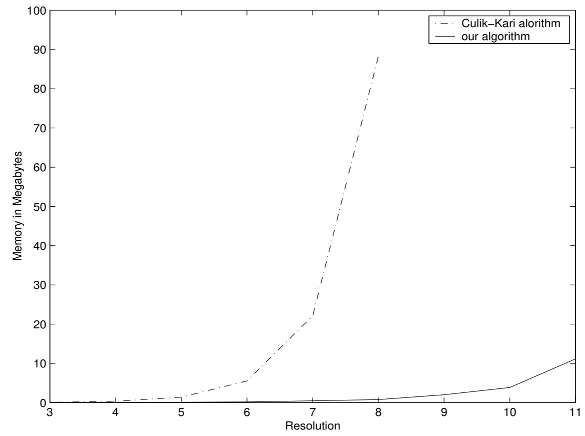

Figure 6: Memory required as a function of resolution

4.1 Further Improvements

In practical applications the WFA may have limited amount of nondeterminism, i.e., many of the values of $I(p)$ and $W_a(p,q)$ may be zero. In such cases further improvement to our basic algorithm is possible. We need not compute many of the functions at some

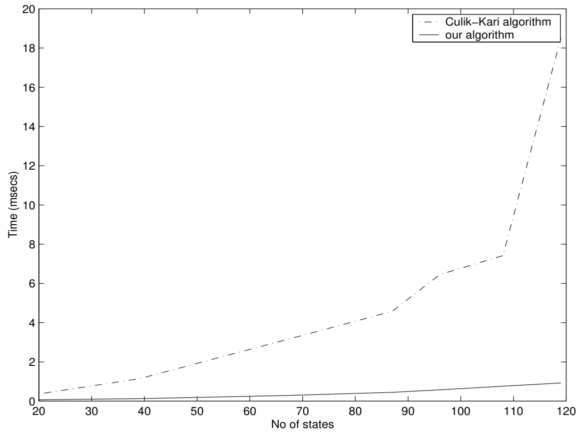

Figure 7: Decoding time as a function of the size of the WFA

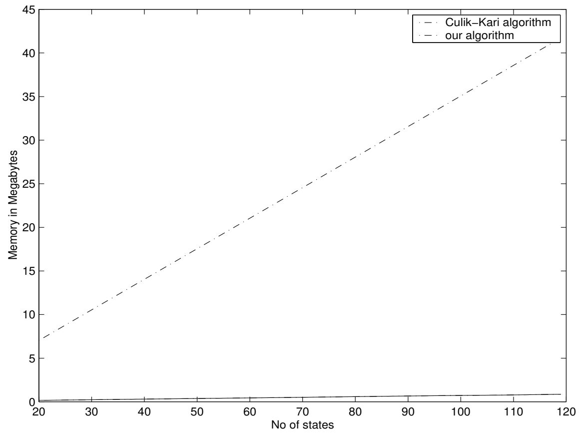

Figure 8: Memory required as a function of the size of the WFA

stages in the computation. We first identify the Φ_ps and ψ_ps that have to be computed at level l in order to compute f_A. We then identify the Φ_ps that have to be computed at level $l-1$ to compute Φ_ps at level l, and so on. We similarly identify the ψ_ps that have to be computed at levels $l-1,\ldots,1$. In the algorithm, instead of computing all Φ_ps and ψ_ps at all levels from 1 to l, we compute only those which are actually needed.

5 Conclusions

In this paper we have proposed a fast and memory efficient decoding algorithm for WFA. We have proved that our algorithm requires less memory and has lower time complexity than the earlier decoding algorithm proposed by Culik and Kari. We have verified our claims through experimentation.

References

[1] K. Culik and J. Kari, "Image Compression Using Weighted Finite Automata", in *Fractal Image Compression: Theory and Techniques*, Ed. Yuval Fisher, Springer Verlag, pp 243-258 1994.

[2] K. Culik and J. Kari, "Finite State Methods for Compression and Manipulation of Images", *Data Compression Conference*, 1995.

[3] K. Culik and S. Dube, "Using Fractal Geometry for Image Compression", *Data Compression Conference*, 1991.

[4] K. Culik, J. Kari, and V. Valenta, "Compression of Silhouette-like images based on WFA", *Data Compression Conference*, 1997.

[5] A. Jacquin, "A Novel Fractal Block-coding Technique for Digital Images", *ICASSP-90*, 1990.

[6] A. Jacquin, "Image Coding based on a Fractal Theory of Iterated Contractive Image Transformations", *IEEE Transactions on Image Processing*, 1(1), pp 18-30, Jan, 1992.

[7] B. Wholberg and G. Jagger, "A Review of the Fractal Image Coding Literature", *IEEE Transactions on Image Processing*, 8(12), pp 1716-1729, Dec, 1999.

[8] A. Jacquin, "Fractal Image Coding: A Review", *Proceedings of the IEEE*, 81(10), pp 1451-1465, Oct, 1993.

[9] D. Salomon, "Data Compression: The Complete Reference", Springer, 1997.

[10] J. C. Hart, "Fractal Image Compressions and Recurrent Iterated Function Systems", *IEEE Computer Graphics and Applications*, 16(4), pp 25-33, Jul, 1996.