DATA COMPRESSION OF DISCRETE SEQUENCE: A TREE BASED APPROACH USING DYNAMIC PROGRAMMING

Guruprasad Shivaram, Guna Seetharaman and T. R. N. Rao

The Center for Advanced Computer Studies
University of Southwestern Louisiana
Lafayette, LA 70504-4330
{gps,guna,rao}@cacs.usl.edu
Phone: (318) 482-6875
Fax: (318) 482-5791

EXTENDED ABSTRACT†

1. ABSTRACT

A dynamic programming based approach for data compression of a 1D sequence is presented. The compression of an input sequence of size N to that of a smaller size k is achieved by dividing the input sequence into k subsequences and replacing the subsequences by their respective average values. The partitioning of the input sequence is carried with the intention of reducing the mean squared error in the reconstructed sequence. The complexity involved in finding the partitions which would result in such an optimal compressed sequence is reduced by using the dynamic programming approach, which is presented.

2. INTRODUCTION

Problem Definition. The problem is defined as follows. Given a 1D input sequence of size N, and a compression parameter k, we need to divide the input sequence into k subsequences and replace each of the subsequence by its average value. The goal is to find an optimal partitioning that minimizes the error due to this approximation process. If the input sequence is partitioned at points $n_i = n_0, n_1, \cdots, n_k$, with $n_0 = 1$ and $n_k = N$, the net error due to approximation is

$$E = \sum_{i=1}^{k} l_i \sigma_i^2$$

† This work was partly supported by a grant from the National Science Foundation NSF 9210926 and by NASA under the grant NAG-W-4013. **Submitted to the SIAM - ACM Eighth International Symposium on Discrete Algorithms, Jan 1997.**

where l_i is the length of the subsequence i. The parameter l_i, and the parameters resulting due to approximation, namely variance, σ_i, and mean, μ_i, are given by

$$l_i = n_i - n_{i-1}$$

$$l_i \sigma_i^2 = \sum_{i=n_{i-1}}^{n_i} [f(n) - \mu_i]^2$$

$$\mu_i = \frac{\sum_{i=n_{i-1}}^{n_i} f(n)}{n_i - n_{i-1}}$$

Motivation Image data compression involves minimization the number of bits required to represent an image. Data compression is used extensively in applications involving data transmission and data storage. Data transmission applications include remote sensing via satellite, radar and sonar, teleconferencing, computer communications, facsimile transmission. Image storage is required for medical images, magnetic resonance imaging, digital radiology, weather maps, geological surveys, finger print storage etc.

Data compression methods can be broadly divided into two categories, called lossless and lossy compression techniques. From an algorithmic stand point, they are classified into: 1) spatial domain methods; 2) transform domain methods; 3) statistical methods; and 4) hybrid methods. There exists another classification, in information theory, where these algorithms are divided as: channel based methods, and source based methods.

The approach described in this paper follows a lossy compression strategy. We solve the problem by mapping the input sequence onto the leaves *i.e.*, , external nodes, of a complete binary tree, as shown in Figure 1. We adopt a compact binary tree representation. The internal nodes of the binary tree have a value equal to the mean of that of their respective children. A closer look reveals that the problem at hand is equivalent of one of finding a cutset of size k in the binary tree, as shown in Figure 2. The complexity involved in such an effort is reduced by using the proposed dynamic programming approach.

3. PRELIMINARY CONCEPTS

For an input sequence of size N, the height, h, of the binary tree will be equal to $log_2 N$. Assuming that the node j is at level d, the mean and variance of the $l = 2^{h-d-1}$ leaf nodes below it, will be equal to

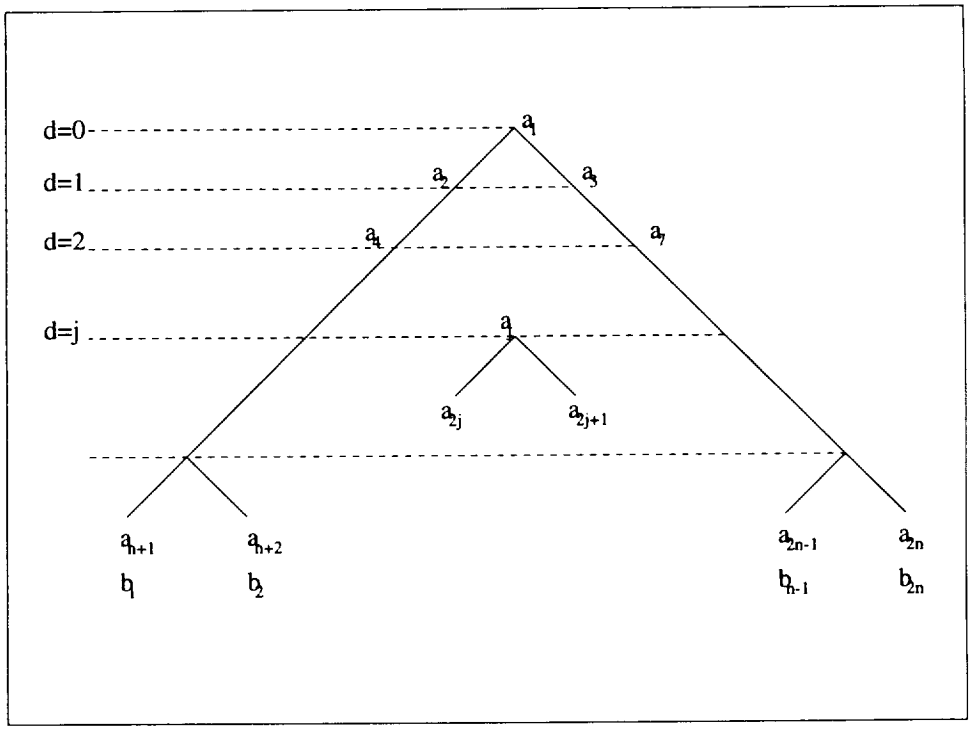


Figure 1. The figure illustrates the building of the binary compression tree.

$$\mu_j = \sum_{i=lj}^{l(j+1)-1} a_i$$

$$\sigma_j^2 = \sum_{i=lj}^{l(j+1)-1} (a_i - \mu_j)^2$$

$$C_j = n_j \sigma_j^2$$

Each internal node j in the tree stores the mean value μ_j and a cost function C_j, where $C_j = n_j \sigma_j^2$, σ_j being the variance of the leaves that are the descendants of j. The cost function would thus represent the net error that would be introduced if the values of these leaves were replaced by μ_j. The cost function, at j can be recursively obtained from the mean and cost functions of its two immediate children using the relationship

$$\mu_j = \frac{\mu_{2j} + \mu_{2j+1}}{2}$$

$$n_j \sigma_j^2 = n_{2j} \sigma_{2j}^2 + n_{2j+1} \sigma_{2j+1}^2 + \frac{n_{2j}}{2} (\mu_{2j} - \mu_{2j+1})$$

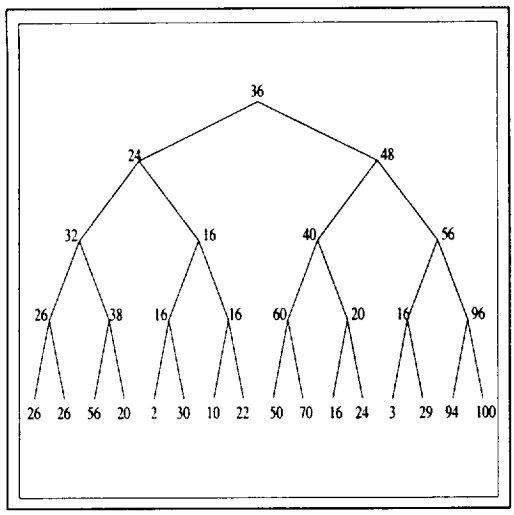
Given the compression parameter k, the problem of data compression is one of finding a node cutset, which is a set of k nodes, that seperates the root from all its leaves. For example the dashed line in Figure 2 constitutes a cutset of size $k = 11$. Thus, the problem of finding a sequence of length k is equivalent to one of finding a cutset, containing k elements in the binary tree. The input signal can be reconstructed, with some error, from the cutset by propogating the mean value stored at each node in the cutset to the leaves that are its descendants in the binary tree, as shown in the Figure 2. Also, the total mean square error in the reconstructed sequence can be computed by simply adding the cost functions stored at each node in the cutset.

Thus, the problem of finding an optimal compressed sequence becomes one of finding a cutset in the compression tree from which if we reconstruct the signal of length N, it would be best represent the original signal. A cutset, containing k elements ,would contain $k-1$ internal nodes in the binary tree . Thus one of the approaches towards finding an optimal solution would be to construct all the binary trees containing $2k-1$ nodes , with the limitation that the height of the tree does not exceed $log_2 2k-1$, and find the tree that would best reproduce the original signal. Without the limitation the problem has an exponential complexity [1]. Thus, this direct approach toward solving the problem is not feasible. However, the proposed dynamic programming approach solves the problem with a complexity of $O(n^2 logn)$.

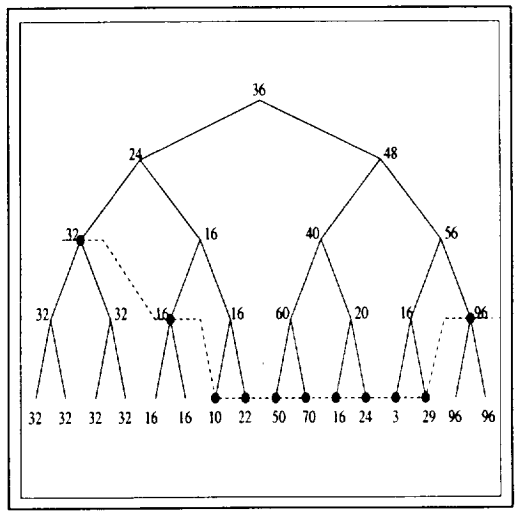
4. DYNAMIC PROGRAMMING SOLUTION

The problem of data compression has two key ingredients that make a dynamic programming solution feasible: optimal substructure and overlapping subproblems [1].

A problem is said to have an optimal substructure when it obeys the Principle of Optimality [2]. The principle states that an optimal sequence of decisions has the property that whatever the initial state and decision are, the remaining decisions must constitute an optimal decision sequence with regard to the state resulting from the first decision. To explain the suitability of applying the dynamic programming technique it is necessary to describe the process of finding the node cutset.



An example illustrating the building of the binary compression tree for an input sequence of size 16



The figure illustrates the reconstruction of the sequence for the given cutset

Figure 2. The figures illustrate the compression and the reconstruction of the input sequence of size 16. A cutset containing k elements would give us a compressed sequence of size k. The figure on the right illustrates the reconstruction of the sequence of size 16 by propogation of the mean at each node of the cutset to its leaves.

A cutset CS, having k nodes can be viewed as being a result of the concatenation of two cutsets, LCS and RCS, that belong to the left and right subtrees of the root of the binary tree, respectively. If the number of nodes in LCS is p then RCS would contain $k - p$ nodes. Thus the cost of transmitting CS is equal to the sum of costs of transmitting LCS and RCS. We thus need to find a $p, 1 \leq p < N$ that would give us a CS of minimum cost. The key to obtaining an optimal solution is that given a value of p, LCS and RCS should by themselves be the results of an optimal merger. If this were not obeyed it would be possible to find a LCS and/or RCS that would be of a lesser cost and hence would result in a final cutset having a lower cost. Thus, the optimal solution to the problem is a result of optimal solutions for the subproblems, which indicates that the problem exhibits an optimal substructure.

To find the optimal solution we modify our representation of the cost function slightly. The cost function, represented by $C(j,k)$, would represent the total error that would result if we were to include a cutset of size k, from a subtree with root at node j, in the final cutset. Note that $C(j,1)$ would represent the total error if the mean value stored at node j is the cutset. $C(j,1)$ can be computed by using the formula.

$$C(j,1) = C(2j,1) + C(2j+1,1) + \frac{n_{2j}}{2}(\mu_{2j} - \mu_{2j+1})$$

Thus, the total error that would result if we transmit a sequence of size k for an input sequence of size n will be $C(0,k)$. Equivalently, the optimal solution to the compression problem can be obtained if we define $C(j,k)$ as

$$C(j,k) = \left\{ \begin{array}{ll} \mu_j & \text{for } k = 1 \\ \min_{1 \leq p < n} C(2j,p) + C(2j+1, k-p) & \text{for } k > 1 \end{array} \right\} \quad (1).$$

One method of solving the above problem would be to adopt divide and conquer approach and solve the problem top-down. But the problem of data compression has a "small" space of subproblems and is of a polynomial input size. A divide and conquer approach solves the same subproblems over and over again thus increasing the complexity.

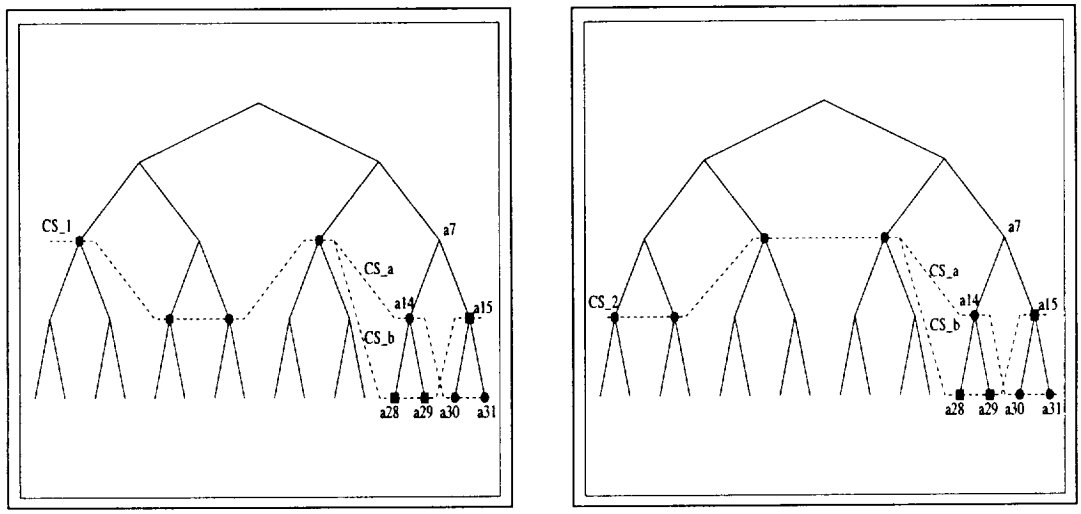


Figure 3. The figure gives an example of a situation where a recursive solution to the data compression problem does not take advantage of the optimal substructure inherent in the problem.

The complexity involved in the top-down approach of solving the problem can be reduced by using a dynamic programming approach. While the recursive algorithm solved the subproblems a number of times, the dynamic programming approach solves each of them only once. The dynamic programming approach obtains an optimal solution to the problem by using a bottom-up approach, where the cost of the optimal cutsets of a subtree are obtained from those of its subtrees. The optimal solution to the various subproblems are stored in a tabular form and is looked up whenever necessary, instead of being solved all over again.

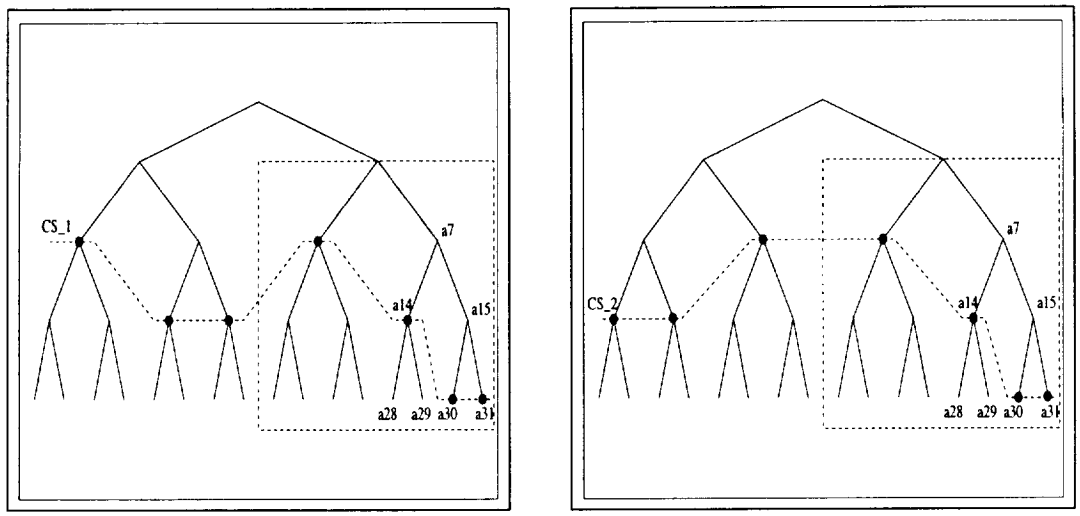


Figure 4. The figures indicate that both, CS_1 and CS_2, have the same subproblem, indicated within the dotted box. A recursive solution would solve the subproblem during the evaluation of both CS_1 and CS_2.

The table for an input sequence of size eight is shown in Figure 5. As is evident from the figure, the table contains a block for each node in the compression tree. Thus the number of blocks at any level in the table is equal to the number of nodes in the tree at the corresponding level. The number of entries in each block of the table is equal to the maximum number of resources that can be allocated to a subtree with its root at the corresponding node in the tree. The entries in a block j store the value of the cost, $C(j,k), 1 \leq k \leq maximumresources$, of all the optimal cutsets. The values of $C(j,k)$ is found by using the formula (1). Note that the values of $C(2j,*)$ in (1) has already been computed and stored in the table and its value can be obtained by looking up the corresponding entry in the table. The complexity of solving the problem using this technique is $O(n^2 logn)$.

5. EXPERIMENTAL RESULTS

The dynamic programming technique was implemented and was tested on a number of images after representing the image as a 1D sequence. The results of applying the technique on a human face and on an oceanographic image have been presented in Figures 6 and 7 respectively. It is seen that the technique reproduces the sharp edges very well. It is also seen that the technique produces very good results on the oceanographic image as the image is largely flat.

6. CONCLUSION

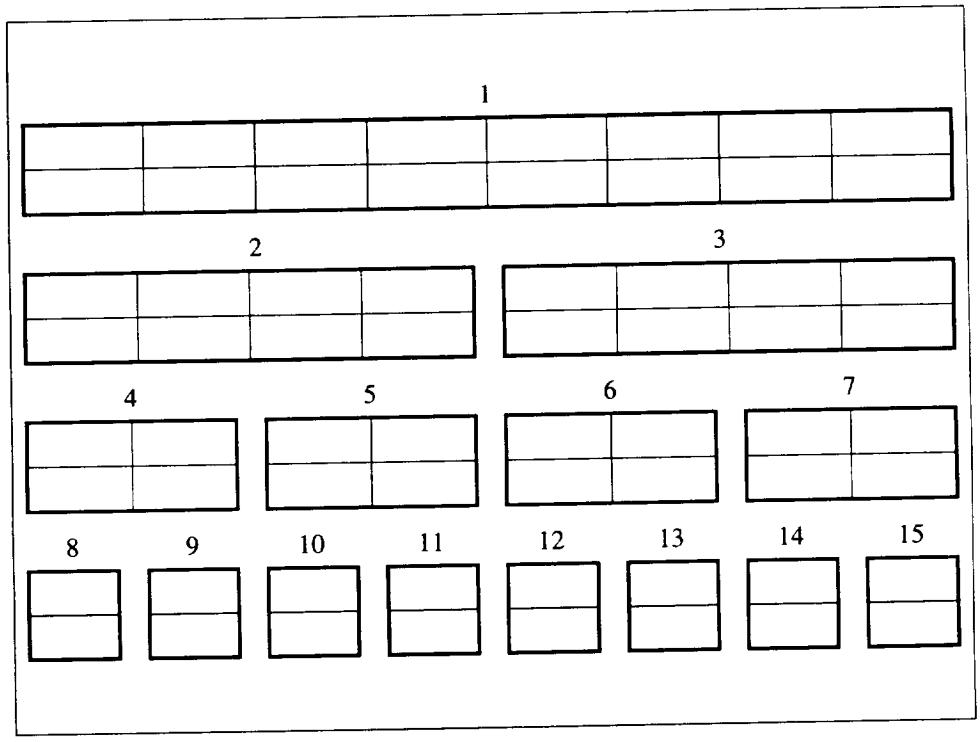


Figure 5. Table for storing the $C(j,k)$ for an input sequence containing eight elements. The number above each box indicates the corresponding node number in the compression tree. Note that the number of entries in each box is equal to the maximum number of nodes that can be allocated to each node.

The proposed method has been tried on several images. The experimental results indicate the compression is efficient in reproducing sharp edges. This method will be equivalent to wavelet in performance. Our method divides the 1D sequence of numbers into several runs whose length is restricted to powers of two. This enables us to solve the identification of optimal cutset in a tractable way. An arbitrary run-length coding is NP. Further results and comparative studies will be presented in the final paper.

7. BIBLIOGRAPHY

[1] Leiserson Cormen and Rivest. *Introduciton to Algorithms.* Mc Graw Hill, 1994.

[2] Richard Bellman. *Dynamic Programming.* Princeton University Press, 1957.

Bit rate=1.49 bits/Pixel

Bit rate=2.25 bits/Pixel

Figure .

Bit rate=3.0 bits/Pixel

Original image

Figure 6. Results of applying the compression technique on a human face.

Bit rate=0.6 bits/Pixel

Bit rate=0.97 bits/Pixel

Figure .

Bit rate=1.33 bits/Pixel

Original image

Figure 7. Results of applying the compression technique on an oceanographic image.