Lossless image data sequence compression using optimal context quantization

Forchhammer, Søren; WU, Xiaolin; Andersen, Jakob Dahl

Published in:
Data Compression Conference. Proceedings

Link to article, DOI:
10.1109/DCC.2001.917136

Publication date:
2001

Document Version
Publisher's PDF, also known as Version of record

Lossless image data sequence compression using optimal context quantization

Søren Forchhammer[1], Xiaolin Wu[2]*and Jakob Dahl Andersen[1]
[1]Dept. of Telecom. 371, Technical University of Denmark
[2]Dept. of Comp. Science, U. of West. Ontario, London, ON. Canada
e-mail: sf@tele.dtu.dk, wu@csd.uwo.ca, jda@tele.dtu.dk

Abstract

Context based entropy coding often faces the conflict of a desire for large templates and the problem of context dilution. We consider the problem of finding the quantizer Q that quantizes the K-dimensional causal context $C_i = (X_{i-t_1}, X_{i-t_2}, \ldots, X_{i-t_K})$ of a source symbol X_i into one of M conditioning states. A solution giving the minimum adaptive code length for a given data set is presented (when the cost of the context quantizer is neglected). The resulting context quantizers can be used for sequential coding of the sequence $X_0, X_1, X_2, \ldots$. A coding scheme based on binary decomposition and context quantization for coding the binary decisions is presented and applied to digital maps and α-plane sequences. The optimal context quantization is also used to evaluate existing heuristic context quantizations.

1 Introduction

A key problem in sequential source coding of a discrete random sequence $X_0, X_1, X_2, \cdots$ is modeling the underlying conditional distribution of the source

$$P(X_i|X^{i-1}), \tag{1}$$

where X^{i-1} denotes $X_0, X_1, \cdots, X_{i-1}$, the prefix of X_i. Given a model class, the model order or the number of parameters must be carefully selected. Algorithm *Context* [1] dynamically selects the variable-order finite-state Markov (FSMX) model with the optimal complexity. The algorithm organizes the contexts in a tree and it can be shown to be universal for the class of FSMX sources [1]. These sources are tree structured. Many practical source coders choose *a priori* a model with fixed complexity, based on domain knowledge such as correlation structure and typical data length, and estimate only the model parameters. E.g. for binary images the JBIG standard uses the contexts of a fixed size template. Estimating $P(X_i|C_i)$ directly using count statistics from past samples can face severe context dilution problems if the number of symbols

*This work was carried out while X. Wu was a visiting professor at the Tech. University of Denmark

1068-0314/01 $10.00 © 2001 IEEE

in the context is large, or if the symbol alphabet is large, resulting in large estimation errors. To avoid this problem, the state-of-the-art lossless image compression algorithm CALIC [3] and the JPEG 2000 verification algorithm EBCOT [2] quantize the modeling context into a relatively small number of conditioning states, and estimate $P(X_i|Q(C_i))$ instead, where Q is a context quantizer. This approach has produced some of the best performing signal compression algorithms, despite the fact that they are not strictly universal. A pivotal issue for these source coders, which impacts their rate-distortion performance, is the design of the context quantizer Q. Off-line context quantizer design algorithms for optimizing Q for minimum code length were considered in [3, 4]. These algorithms attempt to find the Q with a given number of conditioning states that minimizes the conditional entropy $H(Y_i|Q(C_i))$, for a particular bit Y_i in the binary representation of X_i. Only quantizers Q with a linear structure are considered in that work. The globally optimal context quantizer for minimum conditional entropy in the original high-dimensional context space was introduced in [5] given the conditional probability density functions. It turns out that this problem is one of optimal vector quantization design with respect to the Kullback-Leibler distance. We call such a vector quantizer a minimum conditional entropy context quantizer (MCECQ). In this paper we consider the issues of estimating the conditional probabilities from a given set of data, leading to a modification of the expression being optimized. Motivated by the success of the 'heuristic quantizers' we focus on this more general class of context quantizers, though they are not universal as eg. the tree structured algorithm [1]. In Section 2, the MCECQ theory [5] is introduced. Section 3 considers the problems involved when using real data and presents a solution based on adaptive code length calculation. Sections 4 and 5 present CQ design algorithms and a CQ-based compression scheme. Some experimental results are given in Section 6.

2 Context Quantization - MCECQ

This section briefly presents MCECQ as introduced in [5]. Let Y be a discrete random variable, and let C be a jointly distributed random vector, possibly real. Given a positive integer M, we wish to find the quantizer $Q : C \to \{1, 2, \ldots, M\}$ such that $H(Y|Q(C))$ is minimized. Clearly, $H(Y|Q(C)) \geq H(Y|C)$ by the convexity of H. However, we wish to make $H(Y|Q(C))$ as close to $H(Y|C)$ as possible. Equivalently, we wish to minimize the non-negative "distortion" of Q

$$D(Q) \;=\; H(Y|Q(C)) - H(Y|C). \tag{2}$$

The quantization regions $A_m = \{\mathbf{c} : Q(\mathbf{c}) = m\}$, $m = 1, \ldots, M$, of an (optimal) minimum conditional entropy context quantizer are generally quite complex in shape, and may not even be convex or connected. However, their associated sets of pmfs $B_m = \{P_{Y|C}(\cdot|\mathbf{c}) : \mathbf{c} \in A_m\}$ are simple convex sets in the probability simplex for Y, owing to the necessary condition for optimal Q [5].
If Y is a binary random variable, then its probability simplex is one-dimensional. In this case, the quantization regions B_m are simple intervals. If the random variable Z is defined as $P_{Y|C}(1|C)$ (the posterior probability that $Y = 1$ as a function of C), then the conditional entropy $H(Y|Q(C))$ of the optimal context quantizer can be expressed by

$$H(Y|Q(C)) = \sum_{m=1}^{M} P\{Z \in [q_{m-1}, q_m)\} H(Y|Z \in [q_{m-1}, q_m)) \tag{3}$$

for some set of thresholds $\{q_m\}$ specifying the quantization regions B_m. Therefore the optimal MCECQ can be found by searching over $\{q_m\}$. This is a scalar quantization

problem, which can be solved exactly using dynamic programming. In this way, the problem of optimal MCECQ design is reduced to one of scalar quantization, regardless of the dimensionality of the context space. Once the scalar problem is solved, the optimal MCECQ cells A_m are given by

$$A_m = \{\mathbf{c} : P_{Y|C}(1|\mathbf{c}) \in [q_{m-1}, q_m)\}. \tag{4}$$

In [6] the minimum conditional entropy criterion above was applied to compute optimal context quantizer for the application of bi-level image compression (tested using the leave-one-out approach). The authors showed that the loss function (2) satisfies a so-called 'concave Monge property'. As a result, the optimization of $\{q_m\}$ in (3) may be solved in $O(kM)$, where k is the number of raw contexts before quantization.

3 Empirical Context Quantization - Binary Case

Context quantization (CQ) was introduced in the previous section based on probability density functions, $P(Y|C)$. In this section, we discuss how to estimate these functions and how the CQ algorithm works with the estimates. Specifically, the issues of having finite size data sets for estimation and coding are addressed. In the following development our scope is restricted to CQ for coding binary decisions. Non-binary variables may be decomposed into a sequence of binary decisions, and coded using the proposed method.

Let 0 and 1 be the labels of the binary variable, y. Let n_0 and n_1 denote the counts of 0 and 1, respectively, in a given context $\mathbf{c}$. A simple estimator of the probability that the next occurrence is 0 in context $\mathbf{c}$ is given by

$$\hat{p}(0|\mathbf{c}) = \frac{n_0 + \delta}{n_0 + n_1 + 2\delta} \tag{5}$$

where δ is a parameter of the estimator. This estimator is often the basis of the adaptive probability estimation for coding binary data with $\delta \in [0, 1]$.

One approach to CQ design is to estimate the probabilities of $P_{Y|C}$ using (5) for a large training data set, and apply the dynamic programming MCECQ algorithm to compute the context quantizer that minimizes (3). For even better performance, one can optimize the context quantizer for a given finite-length binary sequence as follows.

3.1 Context Quantization by Adaptive Code Length

Lets consider adaptive context based coding of a binary sequence y^I sequentially using the probability estimate (5) in each context. The probability estimate is updated on the fly for each of the I binary input symbols. For a given binary sequence y^I, the total code length, $L(y^I|Q(\mathbf{c}))$ out of adaptive context-based arithmetic coding may be computed based on the set of counts (n_0, n_1) for all contexts, because the order of 0 and 1 appearrance does not change the adaptive code length. Let l_m denote the adaptive code length of all symbols whose contexts fall into CQ cell A_m. The context quantizer $Q(\mathbf{c})$ minimizing the total code length is determined by

$$min_{M,Q(\mathbf{c})} \sum_{m=1}^{M} l_m = min_{M,Q(\mathbf{c})} L(y^I|Q(\mathbf{c})). \tag{6}$$

This yields the optimal context quantizer $(M, Q(\mathbf{c}))$ for the given data set, y^I, with respect to the adaptive code length, (when the cost of the context quantizer itself is neglected).

In our implementation, we restrict ourselves to quantizer cells specified by $A_m = \{\mathbf{c} : \hat{P}_{Y|C}(1|\mathbf{c}) \in [q_{m-1}, q_m)\}$. Under this restriction the solution may be found by dynamic programming as for the MCECQ. The probability estimates $\hat{P}(Y|C)$ that are used for *ordering* the contexts prior to the dynamic programming procedure are obtained from the final counts on the data set, using (5) with $\delta = 0$.

In calculating the adaptive code length for potential quantization classes, probability estimator (5) is used with δ set to the value used in the coding procedure that we are optimizing for. The adaptive code lengths used in the dynamic programming are calculated based on the counts, (n_0, n_1), for each possible context cell (quantizer interval). Since the dynamic programming algorithm uses the actual adaptive code length for a given finite sequence and a fixed δ as the cost function, it can automatically decide the optimum number of coding contexts M. This is simply done by increasing the number of context quantizer cells in the bottom-up dynamic programming process, until reaching the point when the actual code length starts to increase.

Given a quantizer interval and the associated 0 and 1 counts, the corresponding adaptive code lengths can be computed in $O(1)$ time independent of the interval length by a fast algorithm proposed by [8]. The idea is to use table look-up to compute the adaptive code length for small values of the counts, and use Stirlings approximation for large values when such an approximation yields high precision. With the fast adaptive code length computation technique, one can precompute and store the adaptive code lengths for all possible quantizer intervals. This preprocess takes $O(N^2)$ time, where N is the number of distinct unquantized raw contexts. Aided by the intermediate results of the preprocess (adaptive code lengths of all possible quantizer intervals), the dynamic programming algorithm can be completed in $O(MN^2)$ time. Note that the cost function of adaptive code length does not satisfy the "concave Monge property" (also called quadratic inequality). Therefore, the faster linear-time algorithm of [6] cannot be used to solve our problem.

Another technique to speed up the dynamic programming algorithm is to merge all the raw contexts that have the same counts. This can significantly reduce the number of initial contexts subject to quantization. This will not affect the optimal solution because those contexts would be merged anyways by the CQ scheme above.

The estimator (5) is optimal if the events in a context are independent and the prior distribution initially is beta distributed with nuissance parameter δ. In this view all the contexts of the same counts have the same distribution of the parameter $\hat{p}(0|\mathbf{c})$, which also suggests that they should be quantized into the same context cell.

We refer to the procedure described above as minimum (adaptive) code length context quantization (MCLCQ). The MCLCQ procedure may also be used for on-line context quantization. In this case δ is set to a small value in the initial sorting of $\hat{P}(Y|C)$.

4 CQ-based Coding Process

The context quantization may be calculated based on the counts obtained from a training set and thereafter fixed, or it may adaptively be calculated based on counts from a causal part of the data set. In the latter case, the encoder and decoder must apply the same context quantization algorithm to the causal data. Finally CQ may be used in a two-pass manner: collecting statistics, performing CQ and coding the context mapping as a preamble before coding the data set itself. The two-pass coding

raises the question of coding the context mapping which we do not consider here instead focussing on the first solutions.
We shall only apply context quantization to binary random variables. Non-binary variables, X_i are decomposed into a sequence of binary decisions when being coded. A simple choice is to code the bits of the binary representation of each value x_i.

4.1 Binary Decomposition

In general, any binary decomposition of the values x_i may be used. We shall use the approach of ordering the possible values and let each decision code whether the value is the next in order until the actual value has been coded. We consider two ways of ordering the values: 1) Based on an order of individual pixel values within the context. 2) Dynamic ordering based on the adaptively estimated likelihood of the possible values. As an example of the first type of ordering, PWC [7] starts by coding the binary decision whether the current pixel, x_i is equal to the previous pixel, x_{i-1}. This is done by coding a so called edge-map. The dynamic ordering is elaborated below.

4.2 Two-level Context Based Prediction

The current non-binary value, x_i may be decomposed in order of decreasing estimated conditional binary probabilities. Such a binary decomposition is context specific and data dependent. The decomposition is used both in CQ design and in actual coding. Let z^I denote the training data and z_i the i'th sample of z^I. The training data is traversed once, and statistics is gathered for the unquantized contexts ($\mathbf{c}$). Based on the statistics for the causal part z^{i-1} the values are ordered by $\hat{p}(z_i|\mathbf{c})$, which are estimated using (5). Thereafter the values are considered one by one in order of decreasing probability until the actual value is found. Statistics is gathered for each of these binary decisions. After the statistics for the training data has been gathered, CQ is applied to the contexts, $\mathbf{c}$ for each of these binary decisions. In our tests on image sequences the previous image is used as the training data, z^I for the current image, x^I. The conditional probabilities $\hat{p}(x_i|\mathbf{c})$ used for the binary decomposition in the actual coding are based on the statistics of the previous image, z^I and the causal part of the current image, x^{i-1}.
Very likely (new) contexts or values in a given context will appear in the data being coded. These cases are handled seperately using an escape technique.

5 CQ Coding Scheme

Based on a combination of the techniques in the previous sections, we present a coding scheme for image sequence data. The coding scheme is specified by 1) binary decomposition, 2) context quantization, and 3) adaptive coding. In the actual coding the statistic counts are reset to zero before each image. Thus the probability estimates of the binary variables $\hat{p}(y_i|Q(\mathbf{c}))$ is based on the statistics of the causal part, x^{i-1} of the current image. Based on these estimates the data may be coded using arithmetic coding.
The scheme is aimed at sequences of computer generated image data as maps, graphics, α-planes etc. Single images as individual maps may be tiled and the scheme applied to a sequence of these tiles.
The coding of each pixel value is decomposed in binary decisions, supplemented by M-ary decisions for escapes, in the following order:

Is the current pixel value, x_i equal to

1. the west pixel value? (B)
2. the most probable value appearing in the context (besides the west pixel)? (B)
3. one of the remaining values appearing in the context, $\mathbf{c}$?
4. the most probable (remaining) symbol values? (B)

where (B) marks binary decisions.
The last question may be repeated (three times in our case). If the value has not been coded by affirmative answer to one of the questions above, the actual value among the remaining colors is coded using the full context, $\mathbf{c}$.
Binary CQ based on (5 - 6) is applied directly to the binary decisions above. For questions 2 and 4, binary CQ based on (5 - 6) with two-level contexts is applied as described in Section 4.2. An individual CQ is carried out for each of the binary questions.
A few comments to this scheme may be in place. The first question is always asked. Once the value has been determined no more questions are asked. One or more of the questions after the first may be void depending on the unquantized context and which values have appeared in it. Contexts not present in the data set, which the CQ is based on, are included without quantization when they appear. In the non-binary question 3 only one context is used. The decomposition scheme above is partly inspired by PWC [7], which starts with the same question and RAPP [10] which codes if the value is equal to one of the values in the (4 pixel) template.

6 Experimental Results

The CQ coding scheme of the previous section was applied to digital maps and α-plane sequences. The CQ scheme was also applied as an analysis of the performance of some existing context quantization schemes. The results of the CQ-algorithms are measured by ideal code lengths, i.e. calculating the code length.

6.1 Maps

The coding scheme was applied to street maps. The first test was to measure the performance of coding the first question (the west pixel) and comparing with existing methods applying a heuristic context quantization. A street map of Copenhagen of 723 by 546 pixels with 12 colors was coded [9]. Table 1 gives the results for (optimal template sizes) coding with a 5 pixel template directly compared with context quantization by relative pixel patterns [9], [10] (9 pixels), edge patterns (as in PWC, 10 pixels), and a combination of 4 unquantized pixels and 3 pixels quantized by relative pixel patterns. In comparison, Table 2 gives the optimal adaptive code length using CQ on the same data set. MCLCQ and MCECQ gave the same code lengths within the accuracy given in the table. MCLCQ automatically finds the number of contexts to use. For MCECQ we searched for the number of contexts yielding the shortest code length. These code lengths can not be realized as they require that the decoder knows the exact context quantization. They serve as a lower bound for context based adaptive coding using the pixels of the template. They suggest that improvement is possible over the heuristic methods of Table 1. For this map the first question accounts for most of the code length (0.305 bpp out of 0.381 bpp for the 7 pixel combination of unquantized and relative pixels patterns).

Template	Rel. patt.	Edge pat.	Combined
0.327	0.322	0.331	0.305

Table 1: Copenhagen map. Code lengths for coding the first (west pixel) question in bits per pixel using different coding methods.

Temp. size	West pix (bpp)	Contexts	Count CQ	MCLCQ	MCECQ
1	0.539	12	12	10	10
2	0.431	100	94	22	22
3	0.397	388	261	24	25
4	0.352	1064	488	27	27
5	0.316	2446	744	29	28
6	0.290	5025	1000	31	30
7	0.266	8279	1111	32	31
8	0.248	12505	1199	31	31
9	0.230	17728	1244	32	30
10	0.210	23751	1200	32	31
11	0.192	30257	1178	30	30
12	0.175	37114	1117	31	30

Table 2: Copenhagen map. Ideal code length for coding the first (west pixel) question (bits per pixel) using MCLCQ or MCECQ as a function of the template size. (MCLCQ or MCECQ yield the same code lengths.) The number of different contexts appearing in all and the number of context classes after CQ by counts, MCLCQ and MCECQ are also given.

Browsing maps on the internet, the maps may be tiled and requested sequentially. The tiles of maps already received may be used for coding the next tile. We have conducted initial tests coding the current tile using a context quantizer designed for the previous tile. We conducted a test taking four maps of 1280 by 897 pixels (Table 3). These maps were tiled in 3 by 3 tiles. The tiles were coded in column by column order. For each tile the CQ algorithm was run on the previous tile. For the first tile of each map, a tile from one of the other maps was used as reference. The total over the 9 tiles is given in Table 3. Each map was also coded as 1 tile with a context quantizer designed for one of the other maps. This simulates the case were different maps are requested in sequence or that there is a default quantizer designed on a test set. For three of the maps both MCLCQ set-ups outperform PWC and the 1 tile version is better than the 9 tile version. For one of the images the results are basically the same.

Map	PWC	MCLCQ, 1 tile	MCLCQ, 9 tiles
038r	88.185	81.620	83.435
117r	166.382	149.738	154.615
144r	46.784	46.467	46.862
148r	255.843	199.031	204.252

Table 3: KRAK maps, reduced size, 1280 by 897 pixels. Code lengths (bytes) for PWC and MCLCQ coding the map as 1 and 9 tiles.

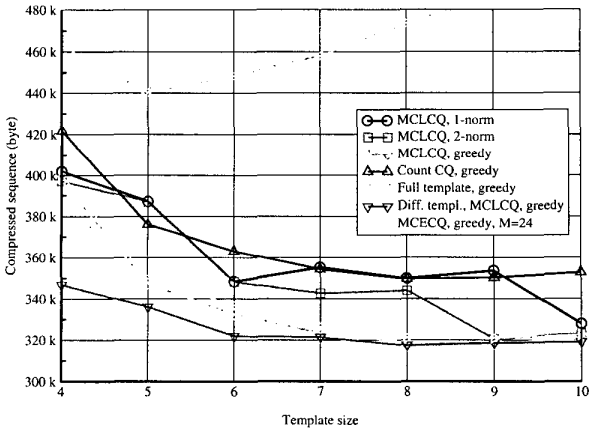

Figure 1: Ideal code length for the Logo sequence as a function of the template size. The template pixels were found by greedy search, 1-norm or 2-norm distances.

6.2 α-planes

In MPEG4 coding using video objects, an α-plane may be associated with each frame of a video object. The α-plane specifies how the video object should be blended with other video objects. The adaptive CQ schemes were applied to three MPEG4 α-plane sequences, designing the context quantizer based on the previous frame. Each frame has 720 by 486 pixels. The results of the Logo sequence is depicted in Figs. 1-3. The first frame is kept out of the results, coding the remaining 299 frames with a total of 104 Mpixels. Fig. 1 gives the total ideal code length as a function of the template size. The best result (diff. templ.) is 317,399 bytes using the MCLCQ algorithm with a 10 pixel template for the three first questions and an 8 pixel template for the rest (escapes). The template pixels are found by a greedy search. In each pass of the data set, neighboring pixels of the template of size n are considered for inclusion in the template of size $n+1$. Several of the other CQ results are close to this for large template sizes. The MCECQ results using a greedy template is almost as good as the MCLCQ results with a greedy template. For MCECQ we searched for the number of contexts ($M = 24$) yielding the shortest code length. The CQ results are better than what is obtained by quantization by counts and much better than the result without context quantization. The result is also significantly better than coding the frames individually using PWC (775,103 bytes) or RAPP (1,067,216 bytes). It may be noticed that the CQ algorithms are robust for increasing template size whereas the full context version deteriorates above 5 template pixels.
Fig. 2 depicts the results frame by frame for the best MCLCQ and MCECQ settings (with fixed template size) compared with PWC and RAPP. The two low plateaus of the CQ results reflect parts with very high frame to frame correlation. But also outside these parts of the sequence, the CQ algorithms significantly outperform the other algorithms. For this sequence the average number of decisions per pixel were only 1.03 binary and 0.01 non-binary decisions.
Fig. 3 shows the code length for the first decision (the west pixel) for the Logo sequence. This decision is fully coded based on binary CQ. For a template size of 9, the average number of different contexts per frame was 2579 over the sequence. Mapping contexts with the same counts reduced the average number of different contexts to 179. The optimal CQ algorithms reduced this to as little as 11. Table 4 gives results for two

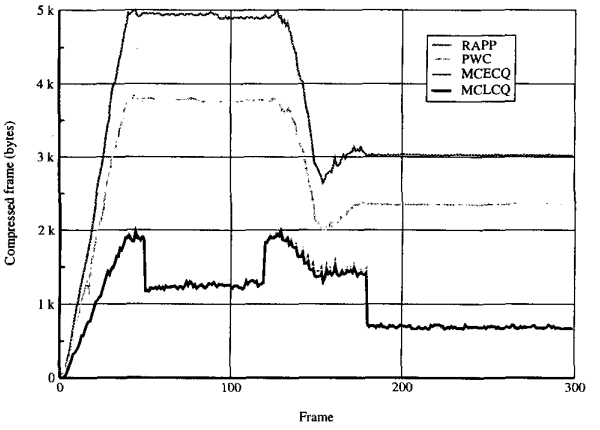

Figure 2: Code length frame by frame for the Logo sequence comparing MCLCQ and MCECQ with RAPP and PWC.

Sequence	MCLCQ	PWC	RAPP
Logo	317.399	820.484	1.067.216
Weather	862.042	1.050.212	883.104
Rain	3.513.183	3.863.470	4.209.170

Table 4: Total code lengths (bytes) for α-plane sequences.

other MPEG4 α-sequences coded by the same techniques. For the Weather sequence 299 frames are coded using a three pixel template combined with a one pixel template for escapes. For the Rain sequence 168 frames are coded using a two pixel template. For both sequences the template pixels are chosen by 2-norm distance. The MCLCQ coding scheme uses statistics (but not specific pixel values) from the previous frame, whereas PWC and RAPP just code individual frames. On the other hand PWC resorts to prediction of values which are different from the nearest neighbors. Introducing prediction and motion compensation would most likely improve the results of the MCLCQ. The emphasis so far has been on the CQ design.

7 Conclusions

A new scheme for context quantization (MCLCQ) based on adaptive code length calculation was presented. The scheme was developed for contexts used to code binary variables. Non-binary data is decomposed into a binary representation. The CQ scheme was used to analyze existing heuristic context quantizations applied to digital maps. Introducing a binary decomposition based on (estimated) likelihood of the possible values, a coding scheme was also presented for image (sequence) data. This may also be used for tiles of an image as eg. in browsing digital maps. Good results were obtained on digital maps and α-plane sequences. We believe the scheme could provide a useful tool for developing context quantizers in entropy coding. The coding performance of the presented scheme could be improved by using large training sets or by special processing of contexts and symbols not present in the data used for designing the CQ. Using it as an entropy coder combining it with processing as

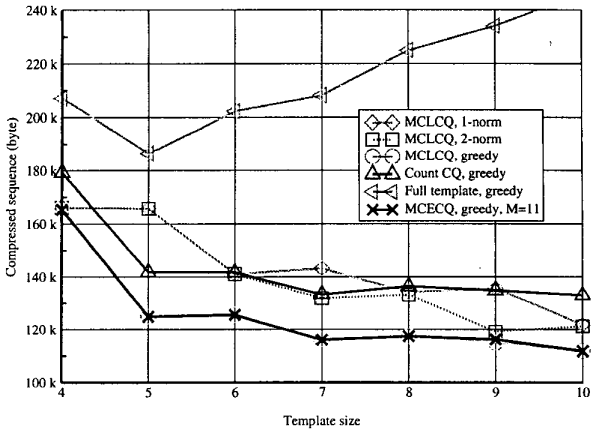

Figure 3: Ideal code lengths for the first question for the Logo sequence.

prediction of values or motion compensation could also improve the results on some of the test data.

References

[1] J. Rissanen, "Universal coding, information, prediction, and estimation", *IEEE Trans. Info. Theory*, vol. 30, pp. 629-636, July 1984.

[2] D. Taubman, "High performance scalable image compression with EBCOT", *IEEE Trans. on Image Proc.*, vol. 9, no. 7, pp. 1158–1170, 2000.

[3] X. Wu, "Lossless compression of continuous-tone images via context selection and quantization", *IEEE Trans. on Image Proc.*, vol. 6, no. 5, pp. 656–664, 1996.

[4] X. Wu, "Context quantization with fisher discriminant for adaptive embedded wavelet image coding", *Proc. Data Comp. Conf.*, pp. 102-111, Mar. 1999.

[5] X. Wu, P. A. Chou and X. Xue, "Minimum conditional entropy context quantization", *Proc. of Int'l. Symp. Inform. Theory, 2000*, pp. 43, 2000.

[6] D. Greene, F. Yao, T. Zhang, "A linear algorithm for optimal context clustering with application to bi-level image coding", *Proc. Int'l. Conf. Image Proc. 1999*,

[7] P. J. Ausbeck Jr., "A streaming piecewice-constant model", *Proceedings Data Compression Conference*, March 1999, pp. 208-217.

[8] B. Martins and S. Forchhammer, "Tree coding of bilevel images", *IEEE Trans. Image Processing*, vol. 7, no. 4, April 1998, pp. 517-528.

[9] S. Forchhammer and O. Riis, "Content layer progressive coding of digital maps", *Proceedings Data Compression Conference*, March 2000, pp. 233-242.

[10] Viresh Ratnaker, "RAPP: Lossless image compression with runs of adaptive pixel patterns", *32nd Asilomar Conf. on Signals, Systems and Comp.*, Nov. 1998.