Distortion-Complexity Optimization of the H.264/MPEG-4 AVC Encoder using the GBFOS Algorithm *

Rahul Vanam[†] Eve A. Riskin[†] Sheila S. Hemami[‡]
Richard E. Ladner[§]
[†] Department of Electrical Engineering, Box 352500,
University of Washington, Seattle, WA 98195-2500.
[‡] School of Electrical and Computer Engineering,
Cornell University, Ithaca, NY 14853.
[§] Department of Computer Science and Engineering, Box 352350,
University of Washington, Seattle, WA 98195-2350.
Email: {rahulv,riskin}@ee.washington.edu,
hemami@ece.cornell.edu, ladner@cs.washington.edu

Abstract

The H.264/ACV standard provides significant improvements in performance over earlier video coding standards at the cost of increased complexity. Our challenge is to determine H.264 parameter settings that have low complexity but still offer high video quality. In this paper, we propose two fast algorithms for finding the H.264 parameter settings that take about 1% and 8%, respectively, of the number of tests required by an exhaustive search. Both the fast algorithms result in a maximum decrease in peak-signal-to-noise ratio of less than 0.71 dB for different data sets and bitrates.

1 Introduction

H.264/MPEG-4 AVC is an international video coding standard that was jointly developed by the ITU-T and the ISO/IEC [1]. Like previous video coding standards, H.264 uses block-based motion compensation prediction. Its large number of parameter choices includes the number of reference frames, variable block size in both motion compensation and residual transform coding, the resolution of motion vectors, and the step sizes for residual quantization. H.264 yields a bitrate savings of about 50% over MPEG-2 for the same video quality [2]. Choosing the right set of encoder parameters results in efficiently coded video while

*This work is supported by the National Science Foundation under grant numbers CCF-0514353 and CCF-0514357.

an inappropriate selection of parameters wastes bits, sacrifices quality, and takes longer to encode. Joint rate-distortion-complexity (R-D-C) analysis of H.264 is complex due to the large number of possible combinations of encoding parameters.

Formalized complexity-distortion analysis (or complexity-rate-distortion analysis) has been proposed for still image coding [3, 4]. In these works, the traditional R-D optimization is augmented with complexity constraints and formulated using the common Lagrange optimization technique. The particular image coding techniques selected (tree-structured vector quantization (TSVQ) in [3] and quadtree-based DCT coding in [4]) are relatively easily cast in this formulation. To provide reasonable complexity, the TSVQ optimization resorts to a greedy algorithm and simpler distortion measures, while the DCT algorithm optimizes for variable complexity in the decoder.

Motion-compensated transform-coded video compression algorithms cannot in general be cast in such a nicely unified approach for either R-D or R-D-C optimizations. The large number of parameter selections for each frame, the many conditional decisions made in encoding, and the dependent coding of multiple frames make exhaustive search techniques infeasible, and even intelligently derived R-D optimization algorithms have a complexity on the order of 10^{12} to 10^{14} [5, 6].

While several attempts have been made to include complexity considerations in video coding, these techniques have been largely *ad hoc* and heuristic, and no actual optimization is performed [7, 8, 9]. Rather, a subset of coding parameter choices is selected (e.g., using a reduced number of search locations for motion estimation) and algorithmic simplifications are enforced (e.g., simple thresholding for mode decisions), and then the effect of each parameter choice and simplification on both performance and complexity reduction is quantified. An initial attempt at optimization is made in [10], where the coding choices are limited, but then a local search on these is performed to minimize complexity while meeting a distortion constraint. A substantial assumption in this approach is that distortion is convex in the coding parameters; additionally, rate is not considered.

There has been prior work done in power-rate-distortion optimization for a complexity scalable video coder, where complexity is addressed in terms of a power constraint [11]. R-D-C optimization of integer motion estimation in H.264 has been discussed in [12]. In multimedia transmission, R-D-C adaptation of the bitstream to receiver constraints and network conditions has been discussed in [13]. Complexity-distortion analysis of H.264 decoder has been done in [14], where they found the use of I-frames or I and P-frames to outperform B-frame structures when using the same bitrate constraints.

In this paper, we introduce a distortion-complexity optimization approach for the H.264 encoder. We assume that the operating bitrate is known. We consider encoding time and mean squared error (MSE) as our measure for complexity and distortion. Specifically, we obtain the MSE of luma (Y) component for each frame and use its average over the video as a measure for distortion.

Typically, to find the best encoder parameters, a large number of tests will have to be conducted. Each test results in a distortion-complexity point as shown in Fig 1. The best encoder parameters correspond to the points on the D-C convex hull of Fig 1. Even with fixed rate and current available fast computers, the process of finding the D-C convex hull is very time consuming.

In this paper, we propose two fast approaches that use the generalized Breiman, Fried-

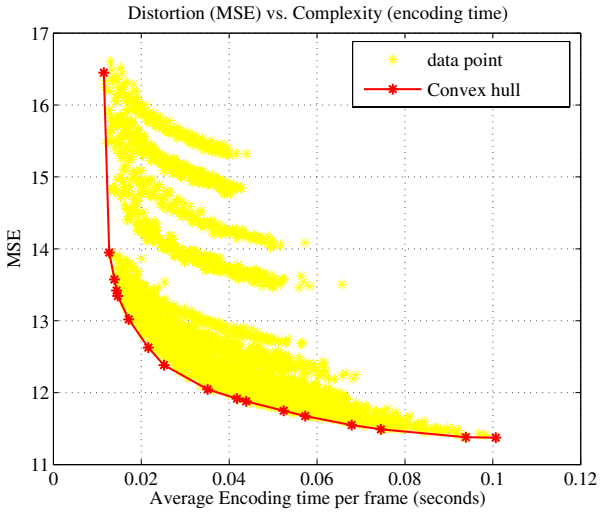

Figure 1: Distortion (MSE) vs. complexity (average encoding time).

man, Olshen, and Stone (GBFOS) algorithm [15] for obtaining parameter settings that result in D-C points that are close to optimal. The algorithms are tested using x264, an open source H.264 encoder [16]. The x264 encoder has been compared with different commercial H.264 encoders in a recent study [17], and it was found to provide the best quality.

This paper is organized as follows. In Section 2, we describe two algorithms for D-C optimization of the H.264 encoder. In Section 3, we describe the test procedure and Section 4 includes a performance comparison of our algorithms to the convex hull points. We also show that both the algorithms perform well for different bitrates and a wide variety of video content, and is robust to changes in the training data set. We conclude in Section 5.

2 GBFOS for Distortion-Complexity Optimization

The GBFOS algorithm is an algorithm for optimally pruning tree-structure classifiers and coders [15],[18]. It has also been used for optimal bit allocation for classified vector quantizer with multiple classes [19]. A *tree functional* is a real-valued function of trees, such as average distortion or rate [18]. The GBFOS algorithm requires the tree functionals to be monotonic and linear or affine. While distortion and complexity of the H.264 encoder parameters are not additive, here we make an assumption that they are, which will result in points that are not necessarily optimal. This assumption is made when we vary one of the encoder parameters while setting other parameters to its lowest MSE option, and attribute the resulting distortion-complexity to the variable parameter alone. However, the distortion-complexity depends upon all encoder parameters. In this section, we describe two approaches that use the GBFOS algorithm for obtaining encoder parameter settings. The first approach requires only one iteration of encodings, and we call this the *GBFOS-basic* algorithm. The second algorithm requires more than one iteration of encodings, and we call this the *GBFOS-iterative* algorithm.

In our optimization problem, let $\{1, \ldots, M\}$ be a set of M encoder parameters, with

each parameter i having n_i options $\{1, \ldots, n_i\}$ arranged in order of decreasing MSE. We denote a parameter setting as a vector $m = (m_1, \ldots, m_M)$. For each parameter i, we vary $m_{i,j} = (n_1, \ldots, n_{i-1}, j, n_{i+1}, \ldots, n_M)$, for $1 \leq j \leq n_i$, and obtain its corresponding distortion $d_i(m_{i,j})$ and complexity $c_i(m_{i,j})$. From the l_i points on the convex hull of the distortion-complexity plot, we obtain parameter settings and corresponding slopes. (The number of convex hull points l_i may be less than the total number of parameter options n_i.) We reindex the points on the convex hull from 1 to l_i. Therefore, the parameter setting of a convex hull point is given by $q_{i,j} = (n_1, \ldots, n_{i-1}, j, n_{i+1}, \ldots, n_M)$, where $1 \leq j \leq l_i$ and $l_i \leq n_i$. The vector $q_{i,j}$ and corresponding slopes are used in the initialization step of both the GBFOS-basic and GBFOS-iterative algorithms. The resulting encoder parameter settings selected by each algorithm is given by vectors p.

We define the slope between points on the D-C convex hull as

$$S_i(q_{i,j}, j) = -\frac{d_i(q_{i,j}) - d_i(q_{i,j-1})}{c_i(q_{i,j}) - c_i(q_{i,j-1})}. \tag{1}$$

2.1 GBFOS-Basic Algorithm

In the GBFOS-basic algorithm, we successively compare slopes from D-C plots of different parameters to prune the parameter with the least slope. A new encoder parameter setting is obtained at the end of each slope comparison. The GBFOS-basic algorithm is given in Fig 2. We first initialize the vector p to the minimum distortion and maximum encoding complexity parameter setting as given in step (1). At the end of each iteration in step (2), we obtain a new parameter setting p.

1. Initialize $p = (l_1, \ldots, l_i, \ldots, l_M)$ and for each parameter i, set $q_i = (n_1, \ldots, n_{i-1}, l_i, n_{i+1}, \ldots, n_M)$.
2. Repeat until $p = (1, \ldots, 1)$:
   (a) Choose parameter i^* that results in the lowest slope $S_{i^*}(q_{i^*}, l_{i^*})$.
   (b) Set $l_{i^*} = l_{i^*} - 1$. Set $q_{i^*} = (n_1, \ldots, n_{i^*-1}, l_{i^*}, n_{i^*+1}, \ldots, n_M)$.
   (c) Determine new $p = (l_1, \ldots, l_{i^*-1}, l_{i^*}, l_{i^*+1}, \ldots, l_M)$.

Figure 2: The GBFOS-Basic algorithm.

2.2 GBFOS-Iterative Algorithm

In the GBFOS-iterative algorithm, we compare the D-C slopes of different parameters in each iteration. Based upon the lowest slope parameter obtained in each iteration, we determine the parameters for which encodings have to be performed in the next iteration. At the end of each iteration, a new parameter setting is obtained. The GBFOS-iterative algorithm is given in Fig 3. It uses the same initialization step as the GBFOS-basic algorithm. In step 2c(ii), the number of points on the convex hull l_i' may be less than l_i. Therefore, the parameters are reindexed from 1 to l_i'.

1. Initialize $p = (l_1, \ldots, l_i, \ldots, l_M)$ and for each parameter i, set $q_i = (n_1, \ldots, n_{i-1}, l_i, n_{i+1}, \ldots, n_M)$.
2. Repeat until $p = (1, \ldots, 1)$ in step 2(b):
   (a) Find i^* that results in the lowest slope $S_{i^*}(q_{i^*}, l_{i^*})$ using Equation (1).
   (b) Set $l_{i^*} = l_{i^*} - 1$. Set $p = (l_1, \ldots, l_{i^*-1}, l_{i^*}, l_{i^*+1}, \ldots, l_M)$.
   (c) For each $i \neq i^*$:
      i. Obtain $d_i(m_{ij})$ and $c_i(m_{ij})$ for $m_{ij} = (l_1, \ldots, l_{i-1}, j, l_{i+1}, \ldots, l_M)$, where $1 \leq j \leq l_i$. Obtain the D-C plot and its convex hull points.
      ii. Reindex the parameters of the convex hull points from 1 to l'_i, where $l'_i \leq l_i$.
      iii. Set $q_i = (l_1, \ldots, l_{i-1}, l'_i, l_{i+1}, \ldots, l_M)$. Set $l_i = l'_i$.

Figure 3: The GBFOS-Iterative algorithm.

3 Experiments

In our tests, we use video sequences from three data sets. One of the data sets contains 15 QCIF video sequences [20] (Standard). The other two data sets are of American Sign Language videos created at the University of Washington, with one set (ASL-1) containing 10 video clips at QCIF resolution and the other set (ASL-2) containing 10 video clips at 320x240 resolution. These video clips are encoded using x264 (March 26, 2006 version) [16] at 30 frames per second (fps). We choose three target bitrates: 30 kb/s, 150 kb/s and 300 kb/s. Tests are conducted on a Linux machine with a 2.8 GHz Intel CPU.

The following x264 encoding parameters are considered in our D-C optimization: number of reference frames (`ref`); sub-pixel motion estimation method (`subme`); partition size for intra and inter motion compensation (`part`); quantization approach (`trellis`); DCT size (`dct`); motion estimation method (`me`); motion search method (`ms`); and the use of mixed reference (`mixed-refs`) [16]. We found that the parameters `dct`, `me`, `ms` and `mixed-refs` can be set to their best settings, namely DCT 4x4 or 8x8, quarter pel, uneven multihexagon and enabling the use of mixed reference, respectively, for little additional complexity. The D-C optimization is therefore just done for the encoding parameters `ref`, `subme`, `part` and `trellis`. The parameter `ref` has 16 options $(1, \ldots, 16)$, `subme` has 7 options $(1, \ldots, 7)$, `part` has 10 options and `trellis` has three options $(0, 1, 2)$. The options of all the above parameters are arranged in decreasing order of MSE.

We first determine the slopes of each encoder parameter from its D-C plot using the procedure given in Section 2. For example, to obtain the D-C plot for the number of reference frames (`ref`) we set `subme` = 7, `part` = (P8x8,P4x4,B8x8,I8x8,I4x4) and `trellis` = 2. Figure 4 gives the D-C plots for `ref`, `subme`, `part` and `trellis` corresponding to the ASL-1 data set at a bitrate of 30 kb/s. Clearly, these D-C curves are not convex. The parameter settings and slopes corresponding to the convex hull points are chosen as input to both the GBFOS-basic and GBFOS-iterative algorithms. The GBFOS-iterative uses this data in the second iteration. For example in Figure 4, we find the change in `part` from

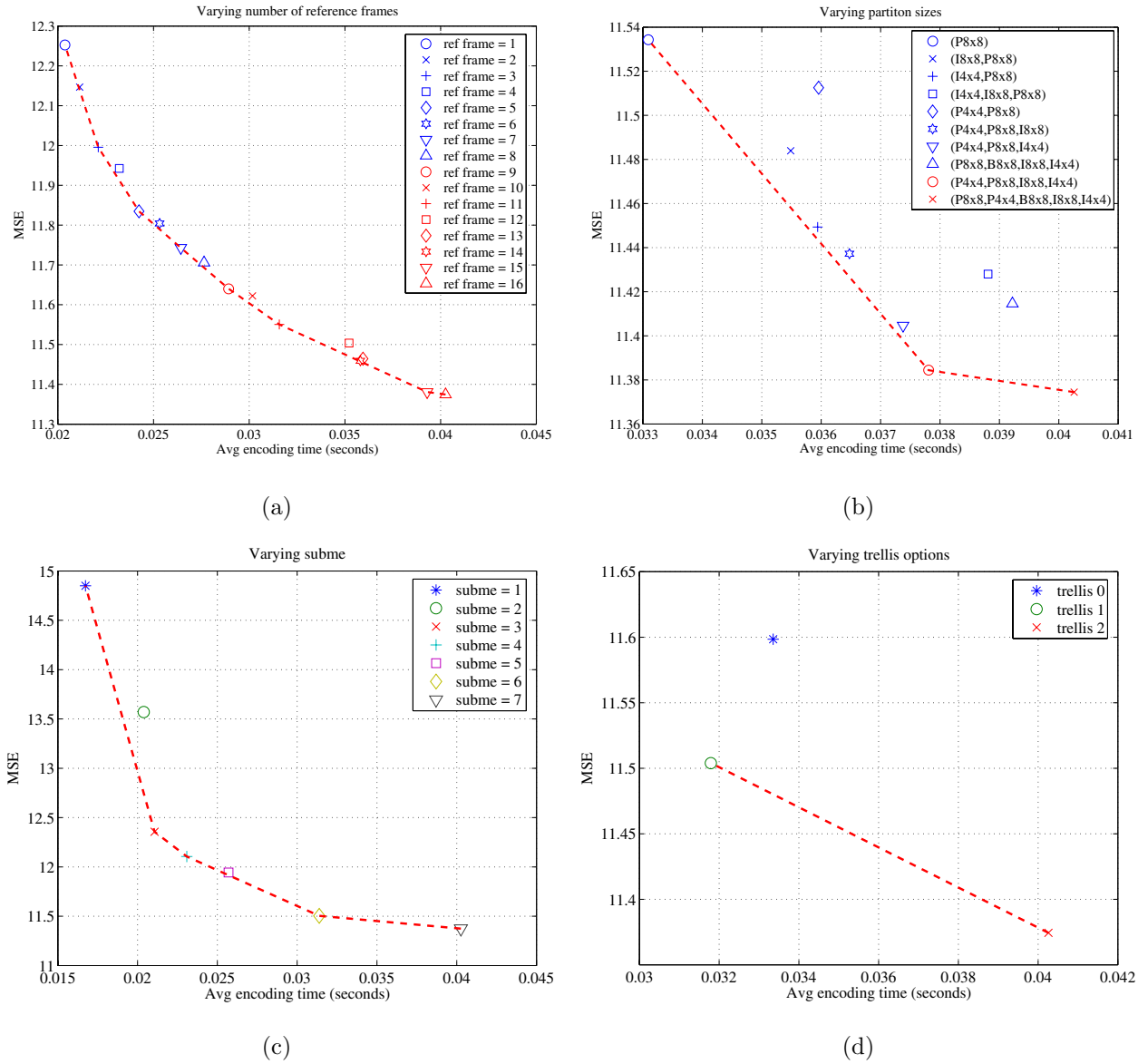

Figure 4: Distortion-complexity plots obtained by varying (a) number of reference frames, (b) partition sizes (the options are listed in the legend), (c) subme values and (d) trellis options. The convex hull points of each plot is shown connected by a dashed line.

(P8x8,P4x4,B8x8,I8x8,I4x4) to (P4x4,P8x8,I8x8,I4x4) results in the lowest slope. In the second iteration, we obtain separate D-C plot for parameters `ref`, `subme` and `trellis` by setting `part` = (P4x4,P8x8,I8x8,I4x4) and other parameters to settings in the previous iteration. Specifically, to generate the D-C plot for `ref` in the second iteration, we set `subme` = 7 and `trellis` = 2 and `part` = (P4x4,P8x8,I8x8,I4x4). The above procedure is repeated in each successive iteration.

4 Results

In this section, we compare the performance of the GBFOS-basic and GBFOS-iterative algorithms to the exhaustive search approach. We also test our algorithms for robustness to change in data content. Let $n_1, ..., n_M$ be the number of options of each M parameters. Then the number of encodings required per video sequence for obtaining the convex hull points exhaustively is

$$N_{convex} = \prod_{i=1}^{M} n_i, \tag{2}$$

while for the GBFOS-basic algorithm, it is

$$N_{GBFOS} = 1 - M + \sum_{i=1}^{M} n_i. \tag{3}$$

The difference in the number of tests required can be quite large, even when using few parameters. For example, using four parameters (`part`, `trellis`, `ref` and `subme`), we obtain $N_{convex} = 3360$ and $N_{GBFOS} = 33$. Therefore, in this example, GBFOS-basic takes about 1% of the total number of tests required for obtaining the convex hull points. Clearly, the total number of tests for obtaining the convex hull points will increase with the number of encoder parameters.

We test the proposed algorithms using three data sets and three target bitrates. Each GBFOS algorithm parameter set corresponds to a D-C point and we compare it with the closest convex hull point obtained by testing each of the 3360 possible encoder parameter options. We use peak-signal-to-noise ratio (PSNR) of the luma component as our measure for distortion, defined as

$$PSNR = 10 \log_{10} \frac{255^2}{MSE}, \tag{4}$$

where MSE is the average mean squared error of the luma component per frame. Figure 5 shows the PSNR vs. average encoding time for the GBFOS-basic, GBFOS-iterative algorithm and convex hull points, respectively, for the ASL-1 data set at 30 kb/s. As expected, both the GBFOS-basic and GBFOS-iterative algorithm have points that do not fall on the convex hull, due to the sub-optimality of the independence assumption. As expected, we find that the GBFOS-iterative points are closer to the convex hull than the GBFOS-basic points, but both algorithms perform very well.

We compute the PSNR difference between each GBFOS point and the closest convex hull point that has greater or equal PSNR value. The maximum PSNR difference over all points is used to evaluate the performance of each of our algorithms. In Figure 5, the maximum PSNR difference for both our algorithms corresponds to the point that has the lowest PSNR (fastest encoding speed). We have found that as the PSNR decreases, both the GBFOS algorithms chooses parameter options such that they either remain unchanged or a lower complexity option is chosen. For example, in Figure 5, the parameter settings for the second highest PSNR point are the same as the highest PSNR point, except for the partition size that has a lower complexity option. However, this is not necessarily true for the convex hull points. Tables 1 and 2 give the number of encodings per video sequence

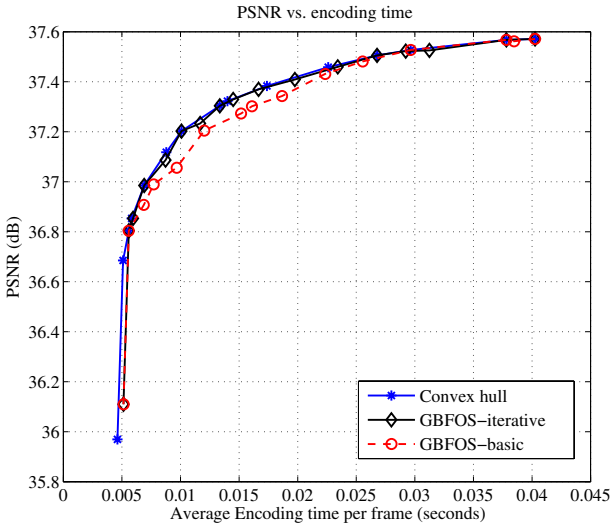

Figure 5: PSNR vs. average encoding time per frame for the convex hull points, GBFOS-iterative and GBFOS-basic algorithm for ASL-1 data set at 30 kb/s.

Table 1: Results for the GBFOS-basic algorithm: maximum PSNR difference from the convex hull and number of encodings.

Bitrates	ASL-1 data set		ASL-2 data set		Standard data set	
	Max PSNR diff (dB)	Number of encodings	Max PSNR diff (dB)	Number of encodings	Max PSNR diff (dB)	Number of encodings
30 kb/s	0.575	33	0.214	33	0.184	33
150 kb/s	0.707	33	0.493	33	0.435	33
300 kb/s	0.535	33	0.152	33	0.438	33

Table 2: Results for the GBFOS-iterative algorithm: maximum PSNR difference from the convex hull and number of encodings.

Bitrates	ASL-1 data set		ASL-2 data set		Standard data set	
	Max PSNR diff (dB)	Number of encodings	Max PSNR diff (dB)	Number of encodings	Max PSNR diff (dB)	Number of encodings
30 kb/s	0.575	189	0.283	102	0.184	121
150 kb/s	0.141	161	0.385	81	0.113	250
300 kb/s	0.087	268	0.231	180	0.398	188

and the maximum PSNR difference for both our algorithms for different bitrates and data sets. We find that the maximum PSNR difference over different data set and bitrates for GBFOS-basic and GBFOS-iterative is 0.707 dB and 0.575 dB, respectively. GBFOS-basic takes only 33 encodings while GBFOS-iterative takes a maximum of 268 encodings.

To test for the robustness of our algorithms for change in data content, the encoder

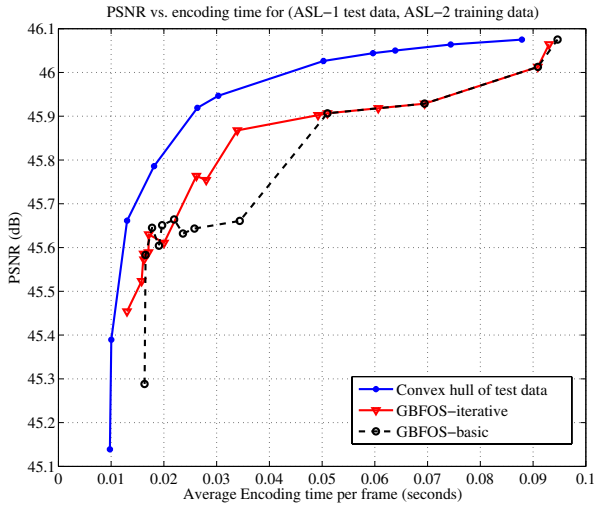

Figure 6: PSNR vs. average encoding time for ASL-1 test data as applied to parameter settings found with the GBFOS-basic and GBFOS-iterative algorithms on the ASL-2 training data set at 150 kb/s.

Table 3: Maximum PSNR difference from the convex hull for the GBFOS-basic and GBFOS-iterative algorithm for different (test set, training set) pairs and bitrates.

Bitrates	Maximum PSNR difference(dB)					
	(ASL-1, ASL-2)		(ASL-1, Standard)		(Standard, ASL-1)	
	Basic	Iterative	Basic	Iterative	Basic	Iterative
30 kb/s	0.17	0.265	0.422	0.422	0.243	0.18
150 kb/s	0.3	0.217	0.538	0.211	0.549	0.145
300 kb/s	0.223	0.051	0.314	0.288	0.213	0.167

parameters obtained from one data set (training set) are applied to a different data set (test set). We use three (test set, training set) pairs, namely (ASL-1, ASL-2), (ASL-1, Standard) and (Standard, ASL-1) for three different bitrates: 30 kb/s, 150 kb/s and 300 kb/s. In each case, we obtain the maximum PSNR difference of our algorithms to the convex hull points obtained from the training data and applied to the test data. Figure 6 gives the PSNR vs. encoding time for (ASL-1, ASL-2) data set pair at 150 kb/s. The maximum PSNR difference for both our algorithms for different (test set, training set) pairs at different bitrates is listed in Table 3, and we find they are within 0.55 dB.

5 Conclusion

In this paper, we proposed the GBFOS-basic and the GBFOS-iterative algorithms for obtaining H.264 encoder parameter settings that result in excellent distortion-complexity performance. The GBFOS-basic and GBFOS-iterative algorithms take only about 1% and 8%,

respectively, of the total number of tests required to find the optimal parameter settings using an exhaustive search. Both the algorithms perform within a maximum PSNR difference of 0.71 dB when using the same training and test data set, and they are robust to changes in both test and training data sets. Finally, as expected, the GBFOS-iterative algorithm performs better than the GBFOS-basic algorithm, at a cost of an increased number of encodings.

References

[1] *Advanced video coding for generic audiovisual services*. ITU-T Recommendation H.264, March 2005.

[2] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Techn.*, vol. 13, no. 7, pp. 560–576, July 2003.

[3] J. Cardinal, "A Lagrangian optimization approach to complexity-constrained TSVQ," *IEEE Signal Processing Letters*, vol. 7, pp. 304–306, Nov. 2000.

[4] K. Lengwehasatit and A. Ortega, "Rate-complexity-distortion optimization for quadtree-based dct coding," in *Proceedings of ICIP*, Sept. 2000.

[5] Y. Sermadevi and S. S. Hemami, "Linear programming optimization for video coding under multiple constraints," in *Proceedings of the IEEE Data Compression Conference*, (Snowbird, UT), March 2003.

[6] Y. Yang and S. S. Hemami, "Generalized rate-distortion optimizations for motion-compensated video coding," *IEEE Trans. Circuits Syst. Video Techn.*, vol. 10, pp. 942–955, Sept. 2000.

[7] P. L. Tai, C. T. Liu, and J. S. Wang, "Complexity-adaptive search algorithm for block motion estimation," in *Proceedings of ICIP*, Oct. 2001.

[8] J. Zhang, Y. He, S. Yang, and Y. Zhong, "Performance and complexity joint optimization for H.264 video coding," in *Proceedings of the 2003 International Symposium on Circuits and Systems*, May 2003.

[9] K. Ramkishor, P. Gupta, T. S. Raghu, and K. Suman, "Algorithmic optimizations for software-only MPEG-2 encoding," *IEEE Trans. on Consumer Electronics*, vol. 50, pp. 366–375, Feb. 2004.

[10] I. R. Ismaeil, A. Docef, F. Kossentini, and R. K. Ward, "A computation-distortion optimized framework for efficient DCT-based video coding," *IEEE Trans. on Multimedia*, vol. 3, pp. 298–310, Sept. 2001.

[11] Z. He, Y. Liang, L. Chen, I. Ahmad, and D. Wu, "Power-rate-distortion analysis for wireless video communication under energy constraints," *IEEE Trans. Circuits Syst. Video Techn.*, vol. 15, no. 5, pp. 645–658, May 2005.

[12] J. Stottrup-Andersen, S. Forchhammer, and S. M. Aghito, "Rate-distortion-complexity optimization of fast motion estimation in H.264/MPEG-4 AVC," in *Proceedings of ICIP*, pp. 111–114, Oct. 2004.

[13] M. van der Schaar and Y. Andreopoulos, "Rate-distortion-complexity modeling for network and receiver aware adaptation," *IEEE Trans. on Multimedia*, vol. 7, no. 3, pp. 471–479, June 2005.

[14] A. Ray and H. Radha, "Complexity-distortion analysis of H.264/JVT decoder on mobile devices," *Picture Coding Symposium*, December 2004.

[15] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*. Monterey, CA: Wadsworth and Brooks, 1984.

[16] "x264." http://developers.videolan.org/x264.html.

[17] "MPEG-4 AVC/H.264 video codec comparison." CS MSU Graphics & Media Lab Video Group, December 2005, http://www.compression.ru/video/index.htm.

[18] P. A. Chou, T. D. Lookabaugh, and R. M. Gray, "Optimal pruning with applications to tree-structured source coding and modeling," *IEEE Trans. on Information Theory*, vol. 35, no. 2, pp. 299–315, March 1989.

[19] E. A. Riskin, "Optimal bit allocation via the generalized BFOS algorithm," *IEEE Trans. on Information Theory*, vol. 37, no. 2, pp. 400–402, March 1991.

[20] "YUV QCIF samples." http://www.tkn.tu-berlin.de/research/evalvid/qcif.html.