DIRECTED ACYCLIC GRAPH BASED MODE OPTIMIZATION FOR H.263 VIDEO ENCODING

Gene Cheung

Hewlett-Packard Laboratories
3-8-13, Takaido-Higashi, Suginami-ku
Tokyo 168-0072 Japan

ABSTRACT

Optimal mode selection for video coding is important in minimizing visual distortion given a rate constraint, and it has been studied in the literature using a single previous macroblock dependency assumption. In this paper, we relax the assumption to multiple-macroblock and develop a directed acyclic graph based procedure that solves the now generalized optimization problem. We demonstrate that in certain cases, multiple block dependencies can be accounted for while remaining computationally feasible. The optimization is sufficiently general that it can be applied to most multimode block-based video coding standards. Experiments show coding gain of up to 0.32dB over UBC's TMN10 implementation. We conjecture that the more general optimization can be more flexibly applied to irregular video objects, as encountered in multiple-VOP mode in MPEG4.

1. INTRODUCTION

The continuing rapid advance of computing technologies means increasingly more complex video encoding algorithms are tolerated and accepted for the sake of better coding performance. ITU-T video coding standard H.263 [1], for example, outperforms its parent H.261 partially because it adopts a more complex picture description syntax, which provides larger flexibility to adapt to a variety of scenes. The cost of better performance, of course, is the higher computational cost associated with finding the best description for a given group of pictures within the now-enlarged confine of syntax specification due to the increased generality.

In the case of H.263, because each macroblock (MB) has a choice of coding mode and quantization step size, finding the best description for a group of pictures on a frame-by-frame basis means selecting a set of modes and quantization step sizes for a given frame that minimize the visual distortion subject to a rate constraint. As an obviously important problem, there are numerous existing works that address the problem [2, 3, 4, 5]. A popular simplifying assumption among these works is that MB dependencies are restricted to be single — each MB depends directly only on one other MB. While this assumption leads to a simple MB-to-MB relationship — one-dimensional dependencies — and a straightforward Viterbi solution, in reality MB-to-MB dependencies are more complex due to differential coding of several neighboring MBs' information.

In this paper, we generalize the single-MB dependency assumption to multiple-MB — two-dimensional dependencies. We show that by viewing the dependency graph as a directed acyclic graph (DAG) and exploiting DAG's properties, the multiple-MB dependencies during mode optimization may not be too computationally prohibitive. For H.263, the algorithm complexity can be controlled by limiting the number of MB rows we are optimizing simultaneously, with [2] being our base case. The optimization discussed here is general enough that it is applicable to most mode selection problems in existing and future multimode block-based video coding standards.

We first give the background and formulation of the optimization problem in section 2. We then develop the optimization procedure in section 3, 4 and 5. Results are presented in section 6. Finally, we conclude in section 7. The discussion focuses on mode selection only, since variation of the same techniques can be used to optimize quantization step size selection as well.

2. PROBLEM FORMULATION

The goal of rate-distortion optimization problems is to minimize the distortion subject to a rate constraint. Assuming the distortion metric is additive, the mode selection problem — selecting the best set of modes $\mathbf{M} = \{M_1, \ldots, M_N\}$ for the N MBs in a frame — can be expressed as follows:

$$\min_{\mathbf{M}} \sum_{i=1}^{N} D_i(\mathbf{M}) \quad \text{s.t.} \quad \sum_{i=1}^{N} R_i(\mathbf{M}) \leq R_s \tag{1}$$

where $D_i(\mathbf{M})$ is the resulting distortion of the ith MB (MB_i) given mode set $\mathbf{M}$, $R_i(\mathbf{M})$ is the resulting rate of MB_i given mode set $\mathbf{M}$, and R_s is the rate constraint of the frame.

2.1. Lagrangian Approximation

As a constrained problem (1) is difficult to solve, and the conventional approach is to solve instead the corresponding Lagrangian, expressed as follows:

$$\min_{\mathbf{M}} \sum_{i=1}^{N} D_i(\mathbf{M}) + \lambda R_i(\mathbf{M}) \tag{2}$$

It can be proven that for a given multiplier λ, an optimal solution to (2), $\mathbf{M}^o$, is also an optimal solution to (1) if $\sum_{i=1}^{N} R_i(\mathbf{M}^o) = R_s$. If the equality condition is not satisfied, then an appropriate value for λ must be derived to drive the sum to R_s while satisfying the inequality in (1), since the approximation bound is related to the numeric difference between the sum and R_s. The focus of this paper is not on the derivation of λ; a wealth of literature including [2, 5] have proposed techniques for finding the appropriate λ. Instead, we will focus on solving (2) given λ.

0-7803-6725-1/01/$10.00 ©2001 IEEE

2.2. Complexity of Dependency

With a closer look at (2), one notices that distortion and rate of MB_i does not directly depend on modes of all the other MBs for most video coding standards. In particular, for INTER frame of H.263, a *candidate predictor* is first calculated for each MB using the MB's three neighboring MBs; the actual motion vector for the MB is the sum of the candidate predictor and the differentially encoded vector for this MB. This means MB_i depends directly only on the mode selection of its three neighboring MBs[1].

Previous works [2, 4] have assumed this dependency is too complex and made the following simplifying assumption to ease the optimization[2]: the rate and distortion of MB_i depends only on the mode of at most one other MB (typically left neighboring MB) besides its own mode M_i. With this assumption, and assuming MBs are numbered from left-to-right, top-down, (2) simplifies to:

$$\min_{\mathbf{M}} \sum_{i=1}^{N} D_i(M_{i-1}, M_i) + \lambda R_i(M_{i-1}, M_i) \qquad (3)$$

The main benefit of using this assumption is that a single dependency relationship leads simply to a special state transition diagram (SD) called a trellis, and the Viterbi algorithm can be used to find the optimal solution to (3) by finding the least cost path of length N through the trellis.

Viterbi algorithm can be viewed as a dynamic programming solution to the problem of finding the shortest path through a *directed acyclic graph* (DAG). DAG is a directed graph with no cycles, and finding the shortest path in a DAG is significantly easier than a general graph [6]. The main contribution of this paper is to show that a multiple-MB dependency relationship can still lead to a state diagram that is a DAG, and hence we can still use a Viterbi-like dynamic programming algorithm to solve the shortest path problem. In the new few sections, we will discuss step-by-step a procedure to construct the corresponding state diagram given the multiple-MB dependency relationship.

3. TOPOLOGICAL SORT

Given a differentially coded block based coding syntax such as H.263, we can construct a *dependency graph* (DG) that shows how each MB is directly dependent on the coded information of its neighboring MBs. For a running example, we see in the dependency graph of Figure 1 that MB_3 is directly dependent on the coded information of MB_1 and MB_2. In general, a DG forms a partial order; the ordering tells us the set of MBs that are first encoded before a given MB due to differential coding.

Since a DG is a partial order, it is also a DAG — a cycle would mean half of the cycle must precede the other and vice versa, which is illogical. Given it is a DAG, one can assign labels $1 \ldots N$ to the N nodes in a DAG with nodes $\mathbf{V}$ and edges $\mathbf{E}$ such that the following property is satisfied for all nodes:

$$\forall i, j \in \mathbf{V}, \quad \text{if } \exists (i, j) \in \mathbf{E}, \quad \text{then } i < j \qquad (4)$$

[1] If Advanced Prediction mode is used, overlapped block motion compensation may cause a MB to depend on the right and/or bottom MB as well. We assume this dependency is secondary and choose to ignore it here.

[2] [2] introduced a decomposition to include the right MB mode in the optimization as well without violating the single dependency assumption. Our technique is more general and applies to any type of dependency.

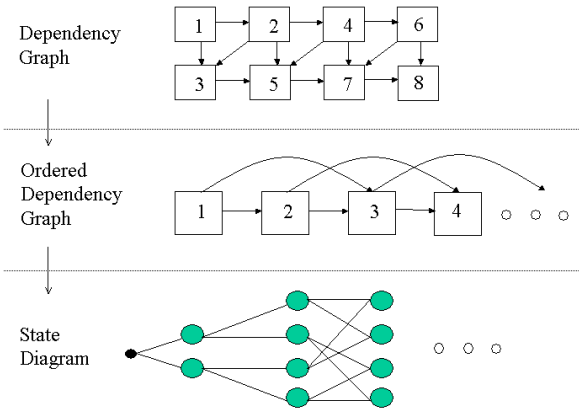


Fig. 1. Example using Proposed Optimization

```
initialize n := 1.
initialize S := {i ∈ V| ∄(.,i) ∈ E}.
while S ≠ ∅,
-> select i ∈ S & remove it from S.
-> remove ∀(i,j) ∈ E, j ∈ V.
-> forall j ∈ V such that ∄(.,j) ∈ E, add to S.
-> label i ∈ V as n.  n++.
```

Fig. 2. Topological Sort

An order that satisfies this property is called a *topological order* [6]. In words, if there is a directed edge from node i to j, label i must be smaller than j. Essentially the partial order is mapped to a total order without violating the original ordering rules of the partial order. A topological sort algorithm is shown in Figure 2.

Notice that in step 1 of the `while` loop, the algorithm does not specify the criterion of node selection in $\mathbf{S}$. In fact, any selection would be a valid topological order. We will discuss a particular criterion later in section 5 for our optimization. After assigning the labels, the MBs can be redrawn from left to right in topological order. We call this graph the *ordered dependency graph* (ODG). Figure 1 shows the ODG for the running example. Notice the edges are all pointing from left to right — a consequence of a valid topological sort.

4. ODG TO STATE DIAGRAM

By redrawing the MBs in a straight line, the ODG reminds us of the single MB dependency assumption [2, 4] where a MB depends only on the left MB. In that case, the optimal mode selection problem is solved by first drawing a special SD (trellis) that corresponds to the associated DG with N stages corresponding to the N MBs. Each stage in the trellis has as many states as there are modes $|\mathcal{M}|$, and $|\mathcal{M}| * |\mathcal{M}|$ edges connect the states between two consecutive stages. The cost of traversing each edge (M_{i-1}, M_i) between two states in the trellis is the Lagrangian cost of going from mode M_{i-1} of stage $i-1$ to mode M_i of stage i, i.e. cost $D_i(M_{i-1}, M_i) + \lambda R_i(M_{i-1}, M_i)$ in (3). Viterbi algorithm is then used to find the shortest path in the trellis.

4.1. Drawing Nodes in SD

In our case, the same simple trellis is insufficient to represent the possible state transitions due to the multiple MB dependencies. For example in Figure 1, MB_3 depends on both MB_1 and MB_2, meaning $|\mathcal{M}|$ incoming edges for each state in stage 3 is not enough to represent all possible mode combination of MB_1 and MB_2. To overcome this limitation, we increase the number of states in the stage 2 to $|\mathcal{M}|^2$ so that the mode used in stage 1 can be preserved. SD in Figure 1 shows the case when $|\mathcal{M}| = 2$. Each edge $((M_1, M_2), M_3)$ now carries the following Lagrangian cost:

$$D_3(M_1, M_2, M_3) + \lambda R_3(M_1, M_2, M_3) \quad (5)$$

An important observation is that since there is no outgoing edges from MB_1 in the ODG that arrive beyond MB_3 in ODG, there is no need to preserve the mode used in stage 1 after stage 3 in SD. More generally, we find the number of states for each stage in SD as follows[3]:

SD Node Construction Procedure: Each stage i has $|\mathbf{h}_i|$ memories, preserving modes of stages $\mathbf{h}_i = \{h_{i,1}, \ldots, h_{i,|\mathbf{h}_i|}\}$. Each stage has $|\mathcal{M}|^{|\mathbf{h}_i|}$ states. $\mathbf{h}_1$ is $\{MB_1\}$. To find $\mathbf{h}_i$ for $i > 1$, apply the following rules in increasing order in i. **Rule 0:** Initialize $\mathbf{h}_{i+1} := \mathbf{h}_i \cup MB_{i+1} \setminus MB_i$. **Rule 1:** If there exists outgoing edges from MB_i in ODG to MBs other than MB_{i+1}, then $\mathbf{h}_{i+1} := \mathbf{h}_{i+1} \cup MB_i$. **Rule 2:** If the origin of an incoming edge at MB_{i+1}, say MB_j, has no outgoing edge beyond MB_{i+1}, then $\mathbf{h}_{i+1} := \mathbf{h}_{i+1} \setminus MB_j$.

Using this procedure, we can find the prescribed memories $\mathbf{h}_i$, and hence the number of states in each stage i.

4.2. Drawing Edges in SD

To complete the SD, we need to connect the states between two neighboring stages in SD with directed edges in a way that modes of specified past stages are preserved. Towards that goal, we do the following:

SD Edge Construction Procedure: first label states in each stage i top-down with a number with base $|\mathcal{M}|$ and $|\mathbf{h}_i|$ digits, with numerical value from 0 to $|\mathcal{M}|^{|\mathbf{h}_i|} - 1$. The number label represents a particular permutation of modes in set $\mathbf{h}_i$. For each state x in stage i, draw an edge to a state y in stage $i-1$ iff the mode permutation reflected in x's label *matches* the permutation reflected in y's label.

By *match*, we mean x's permutation of modes in $\mathbf{h}_i \setminus MB_i$ is the same as y's permutation. We do that by checking the digit in each of x's and y's label that correspond to the same past stage mode are the same. Note that a MB_j in $\mathbf{h}_i \setminus MB_i$ must also be permuted in stage $i-1$ by the node construction procedure.

Figure 3 shows two examples of SDs from ODGs. The node labels in ODGs are in alphabets to avoid confusion. In both cases, $|\mathcal{M}| = 2$, and the number labels for states are the binary numbers inside the nodes. The bottom of Figure 1 shows the SD corresponding to the running example.

After the SD is constructed, a Viterbi-like dynamic programming algorithm can be used to find the shortest path of length N through the SD to find the optimal mode selection for the N MBs.

[3]In fact, $\mathbf{h}_i$ is simply the set of origins with edges drawn from left of MB_i to right of MB_i in ODG plus MB_i. We propose the construction procedure because searching for edges as above is difficult in practice.

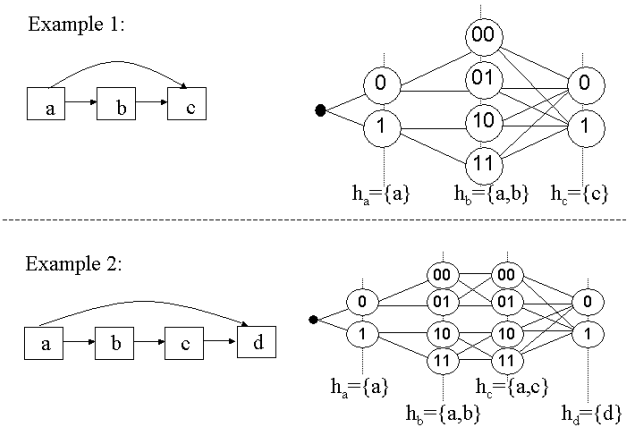


Fig. 3. More Examples of State Diagram

5. COMPLEXITY CONSIDERATION

We first compute the running time of the dynamic programming algorithm used to find the least-cost path through the SD, after constructing the SD using the above discribed procedure. (Construction of SD can be performed offline, and hence it is less important.) For each state in stage i of SD, we find the shortest path to that state from a maximum of $|\mathcal{M}|^{|\mathbf{h}_{\max}|}$ states in stage $i-1$, where $\mathbf{h}_{\max} = \arg\max_{i=1\ldots N} |\mathbf{h}_i|$. The maximum number of states in stage i is the same, and there are N stages. Hence, we can conclude that the running time is $O(N|\mathcal{M}|^{2|\mathbf{h}_{\max}|})$.

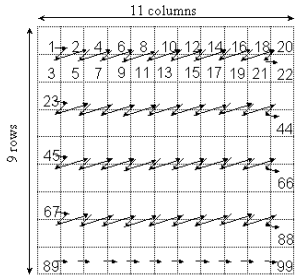
To reduce the complexity of the algorithm, we can only try to minimize $|\mathbf{h}_{\max}|$, since N and $|\mathcal{M}|$ are given parameters of the problem instance. Recall from section 3 that in step 1 of the `while` loop of the topological sort, any nodes with no incoming edges (any nodes in $\mathbf{S}$) can be selected as the next node to be labeled without violating the topological order. We exploit this degree of freedom in an attempt to minimize $|\mathbf{h}_{\max}|$. We do that by selecting a node in $\mathbf{S}$ at iteration i of the loop that yields the minimum number of states for stage i. For the running example in Figure 1, selecting MB_4 when $i = 3$ would lead to $|\mathbf{h}_3| = 3$, while selecting MB_3, as shown in the figure, leads to $|\mathbf{h}_3| = 2$.

If the complexity of the algorithm remains too high even after the proposed node selection in topological sort has been used, we have no choice but to further reduce the complexity at the cost of decrease in coding performance. For H.263, we accomplish that by reducing the number of MB rows that is being optimized at a time. Notice that when the number of MB rows is reduced to one, our optimization is same as the one in [2].

6. RESULTS

To test the effectiveness of the proposed optimization, we first applied the modified topological sort algorithm to the DG of MBs of QCIF size to obtain an ODG — we were optimizing two MB rows at a time in this case. The dependency pattern is the same the one shown in the DG of Figure 1. The result is the ODG shown in Figure 4.

We compared the coding efficiency of our algorithm to Univeristy of Britsh Columbia's TMN10 implementation and Wie-



A Topological Order for QCIF (2 rows)

Fig. 4. Topological Order for QCIF (2 Rows)

	TMN10	Wiegand96 (1 MB row)	Proposed (2 MB rows)
Carphone (38 kbps)	33.84 dB	33.97 dB	34.00 dB
Carphone (76.5 kbps)	37.03 dB	37.24 dB	37.25 dB
Carphone (126kbps)	39.92 dB	40.21 dB	40.24 dB

Fig. 5. Comparison of Coding Results

gand et. al. [2] for the `carphone` sequence. Mode selection in [2] is the same as the base case of our optimization when we fix the number of MB rows being optimized at 1. Tests were performed at 38kbps, 76.5kbps and 126kbps, with quantizer step size fixed at 10, 6 and 4 respectively for I and P frames. The frame rate was chosen to be 10 frames/s, and an intra refresh mode was periodically forced into the selection every 12 frames for every MB. The results in luminance PSNR are shown in Figure 5. We see that the proposed technique has up to 0.32dB improvement over TMN10 and 0.03dB improvement over [2].

7. CONCLUSION

In this paper, we show that even for multiple-node dependencies, one can construct a state diagram that is a directed acyclic graph, and hence one can still use a Viterbi-like dynamic programming algorithm to solve the mode selection problem. We accomplish that by first using a modified version of the topological sort algorithm to transform a dependency graph to an ordered dependency graph, then using a proposed construction procedure to construct the desired state diagram. The complexity of the dependency graph ultimately determines the algorithm complexity, so one can choose to simplify the complexity of the dependency graph *apriori*. In the case of H.263, for example, we show by decreasing the number of MB rows that is being optimized at a time, one can reduce the algorithm complexity.

While we demonstrated some coding gain over UBC's TMN-10 implementation of H.263, we were unable to show significant coding gain over mode selection techniques that assume single-MB dependency such as [2]. However, we conjecture that the proposed optimization is fundamentally more general, and hence conceivably can be more flexibly applied in a more general coding setting, such as for arbitrary shape video objects, as seen in multiple-VOP mode in MPEG4.

8. REFERENCES

[1] ITU-T Recommendation H.263, *Video Coding for Low Bitrate Communication*, February 1998.

[2] T. Wiegand et. al., "Rate-distortion optimized mode selection for very low bit rate video coding and the emergin h.263 standard," in *IEEE Trans. on CSVT*, April 1996, vol. 6, no.2.

[3] G. Schuster and A. Katsaggelos, "Fast and efficient mode and quantizer selection in the rate distortion sense for h.263," in *VCIP, SPIE*, March 1996, vol. 2727, no.2.

[4] D. Mukherjee and S. Mitra, "Combined mode selection and macroblock quantization step adaptation for the h.263 video encoder," in *ICIP*, 1997.

[5] G. Sullivan and T. Wiegand, "Rate-distortion optimization for video compression," in *IEEE SP Magazine*, November 1998.

[6] Cormen, Leiserson, and Rivest, *Introduction to Algorithms*, McGraw Hill, 1986.