A Generalized Output Pruning Algorithm for Matrix-Vector Multiplication and Its Application to Compute Pruning Discrete Cosine Transform

Yuh-Ming Huang, Ja-Ling Wu, IEEE Senior Member, and Chi-Lun Chang

Communications and Multimedia Laboratory
Dept. of Computer Science and Information Engineering
National Taiwan University,Taipei, Taiwan
E-Mail: ymhuang@ncnu.edu.tw or wjl@cmlab.csie.ntu.edu.tw

Abstract - In this paper, a generalized output pruning algorithm for matrix-vector multiplication is proposed first. Then the application of the proposed pruning algorithm to compute pruning Discrete Cosine Transform (DCT) is addressed. It is shown that, for a given decomposition of the matrix of the transform kernel and the pruning pattern, the unnecessary operations for computing an output pruning DCT can be eliminated thoroughly by using the proposed algorithm.

I. INTRODUCTION

Recently, a lot of one-dimensional (1-D) and two-dimensional (2-D) fast pruning DCT algorithms, for computing only the lower-frequency components, have been proposed in [1],[2],[3]. However, to the most of our knowledge, no known generalized pruning method can be directly applied to any orthogonal discrete transform, such as DCT, Discrete Fourier Transform (DFT), Discrete Hartley Transform (DHT), ..., etc. In this paper, a generalized output pruning algorithm for computing matrix-vector multiplication of any order is presented. It is shown that, for a given decomposition of the matrix, the unnecessary operations can be eliminated thoroughly. The efficient Pruning DCT algorithm can then be derived based on the prescribed pruning algorithm. Of course, the applicability of the proposed output pruning algorithm is not limited to DCT, actually, it can be applied to all well-known discrete orthogonal transforms, such as DFT, DHT, and Discrete Sine Transform (DST). However, in this write-up, pruning DCT algorithm is our only focus.

II A GENERALIZED OUTPUT PRUNING ALGORITHM FOR MATRIX-VECTOR MULTIPLICATION

Consider the operation of a general matrix-vector multiplication of order N, say $D_N=A_{N\times N}\times B_N$, and assume only partial multiplication outputs D_j (where $1\leq j\leq N$) are required. It follows that we can speed up the aforecited computation by pruning the unnecessary operations.

To reduce the computational complexity, we decompose the matrix $A_{N\times N}$ into a product of a sequence of more-sparse matrices of the same order, that is, $A_{N\times N}=\prod_{i=0}^{k-1} C^{i}_{N\times N}$. Then, the operation $A_{N\times N}\times B_N$ can be computed as

$$C^{0}_{N\times N}\times C^{1}_{N\times N}\times\cdots\times C^{k-1}_{N\times N}\times B_N,$$

where $C^{i}_{N\times N}$ is more sparse than $A_{N\times N}$ in general. By the associative property of matrix-vector multiplication, D_N can be computed recursively as follows.

$$\begin{cases} B^{0}_{N} = B_N \\ B^{i}_{N} = C^{k-i}_{N\times N}\times B^{i-1}_{N}, 1\le i\le k. \\ D_N = B^{k}_{N} \end{cases} \tag{1}$$

Since there are k stages of matrix-vector multiplication of order N in (1), no matter what kind of output pruning pattern is, k×N bits are required to record whether each $B^{i,j}_{N}$ has to be computed or not, where $B^{i,j}_{N}$ is the inner-product of the j-th row vector of $C^{k-i}_{N\times N}$ and the output vector B^{i-1}_{N} of the previous stage.

In this section, a more efficient algorithm for computing output-pruning matrix-vector multiplication is presented. In this algorithm, only $\lceil \log(k+1)\rceil\times N$ bits are required to record whether the partial results $B^{i,j}_{N}$ has to be computed or not. In other words, we need an array , say M of order N with each entry of $\lceil \log(k+1)\rceil$ bits in size, to record which operations are required or unnecessary.

If the computation of $D_N[i]$ is necessary, then initially let M[i]=0; otherwise let M[i]=255 or a large integer. The final value of each entry of M will evolve gradually through the computation of $C^{0}_{N\times N}$ to that of the $C^{k-1}_{N\times N}$ and will be pre-computed and stored with respect to the characteristics of the concerned matrix $C^{i}_{N\times N}$, described as follows.

(A) **Encoding Processes :**

Let T be a control or threshold parameter and its value is set to be zero, initially.

<1> If $C^{i}_{N\times N}$ is a permutation matrix, that is for any vector V_N of order N, the result of the matrix-vector multiplication $C^{i}_{N\times N}\times V_N$ is just a position swapping of V_N. In this case, the entries of M are unchanged in value but permuted according to the inverse permutation matrix $(C^{i}_{N\times N})^{-1}$, and the value of T is unchanged, neither.

<2> If $C^{i}_{N\times N}$ is a diagonal matrix, that is, all the entries of $C^{i}_{N\times N}$ are equal to zero except the diagonal components. In this case, the values of each entry of M and T will keep unchanged.

<3> If $C^{i}_{N\times N}$ is a general diagonal matrix, that is, all the diagonal components of it are not equal to zero and no constraint is set to the non-diagonal components. In this case, the value of T will be increased by one. The value of T (equal to T_t) is used as a threshold for indicating the fact that: in the matrix-vector multiplication stage, say $C^{i}_{N\times N} \times B^{k-i-1}_{N} = B^{k-i}_{N}$, some output entry $B^{k-i,s}_{N}$ is unnecessary (i.e. M[s] = 255), whereas the entry $B^{k-i-1,s}_{N}$ of the input vector B^{k-i-1}_{N} is required to compute some output entry $B^{k-i,r}_{N}$. That is, the s-th input entry $B^{k-i-1,s}_{N}$ has to be computed correctly before dealing with the matrix-vector multiplication $C^{i}_{N\times N} \times B^{k-i-1}_{N}$, but after then, the s-th output entry $B^{k-i,s}_{N}$ is no use for later stages. In other words, if $M[r] < T_t$, and M[s] = 255, then we set $M[s] = T_t$.

<4> If $C^{i}_{N\times N}$ can be decomposed into a product of a general diagonal matrix and a permutation matrix, or vice versa. In this case, the array M will be processed by using the merged methodologies presented in <1> and <3>.

<5> The other matrix forms which don't belong to those of the above four types are categorized as type <5>. Notice that those matrices discussed in <1>, <2>, and <3> are special subsets of <4>. Hence, by definition, those matrices of type <5> can't be decomposed into a product of a general diagonal matrix and a permutation matrix. Moreover, according to the following two corollaries, we will deduce that each matrix of type <5> is a linearly dependent matrix.

Corollary 1. If a matrix of size N×N cannot be decomposed into a product of a general diagonal matrix and a permutation matrix of the same size, then its determinant is equal to 0.
Proof. We can prove this corollary by induction on N; however, we omit the details due to page limit.

Corollary 2. If the determinant of a matrix H of size N×N is equal to 0, then H is a linearly dependent matrix.

Hence, the matrix discussed in <5> is linearly dependent. Since the determinant has the following property: If

$$A_{N\times N} = \prod_{i=0}^{k-1} C^{i}_{N\times N}, \tag{2}$$

Then

$$\mathrm{Det}(A_{N\times N})=\prod_{i=0}^{k-1} Det(C^{i}_{N\times N}). \tag{3}$$

Therefore, by corollary 2 we know that $A_{N\times N}$ is linearly dependent if $C^{i}_{N\times N}$ is a linearly dependent matrix. Therefore, for any well-defined discrete transform matrix $A_{N\times N}$, which is linearly independent, it will never be categorized as a type <5> matrix.

For the sake of convenience, those sets composed of the matrices discussed in <1>, <2>, <3>, and <4> are respectively denoted by P, D, GD, and PGD.

As we have obtained the final values for each entry of M through the computation of $C^{0}_{N\times N}$ to that of the $C^{k-1}_{N\times N}$, then with the help of M, all the unnecessary operations for the computation of $C^{0}_{N\times N}\times C^{1}_{N\times N}\times\cdots\times C^{k-1}_{N\times N}\times B_N$ can be eliminated thoroughly. The values of all the entries of M will keep unchanged during this process, but may just be permuted in certain occasions. Let the final accumulated value of T be denoted by T_f. This means, among those matrices, $C^{i}_{N\times N}$ $0\leq i\leq k-1$, there are T_f matrices belong to GD or PGD. In other words, after the matrix decomposition, the value T_f is pre-computable.

Now, let us show how the unnecessary operations for the computation of $C^{0}_{N\times N}\times C^{1}_{N\times N}\times\cdots\times C^{k-1}_{N\times N}\times B_N$ can be eliminated thoroughly with the aid of M and T.

(B) Decoding Process :

First, let $T = T_f$. The elimination processes for unnecessary operations are deduced gradually through the computation of $C^{k-1}_{N\times N}\times B_N$ to that of the $C^{0}_{N\times N}\times B^{k-1}_{N}$.

<1> If $C^{i}_{N\times N}\in$ P (the permutation matrix set), then the entries of M will be swapped according to the permutation pattern defined by $C^{i}_{N\times N}$.

<2> If $C^{i}_{N\times N}\in$ D (the diagonal matrix set), then the j-th output $B^{k-i,j}_{N}$ needs to be computed only when $M[j]\leq T$, otherwise it can be left out for pruning the unnecessary operations.

<3> If $C^{i}_{N\times N}\in$ GD (the general diagonal matrix set), then the j-th output $B^{k-i,j}_{N}$ needs to be computed only when $M[j]< T$, otherwise it can be left out for pruning the unnecessary operations. Furthermore, the value of T will be decreased by one in this case.

<4> If $C^{i}_{N\times N}\in$ PGD (the product of general diagonal and permutation matrix

set), it follows that $C^{i}_{N\times N}$ can be decomposed into a product of a general diagonal matrix D_g and a permutation matrix P_p (or vice versa). Then, the entries of M will be permuted according to the P_p first, and the j-th output $B_N^{k-i,j}$ has to be computed only when $M[j] < T$, otherwise it can be left out for pruning . Of course, the value of T will also be decreased by one in this case.

The above statements described the detailed procedures of the proposed pruning algorithm. Because the final value of T, i.e. T_f, will not be greater than k, only $\lceil \log(k+1) \rceil \times N$ bits are required to record the evolution process of M. The correctness of the proposed algorithm can be verified by corollary 3 and lemma 1.

Corollary 3. Let $A_{N\times N}$ $(=\prod_{i=0}^{k-1} C^{i}_{N\times N})$ be a linearly independent matrix. From the proposed algorithm, it can be deduced that, in the computation of $A_{N\times N}\times B_N$, the j-th entry $B_N[j]$ of the input vector B_N is necessary only when $M[j] \leq T_f$.
Proof. We can prove this corollary by induction on k; however, we omit the details due to page limit.

Lemma 1. Let $A_{N\times N}$ $(=\prod_{i=0}^{k-1} C^{i}_{N\times N})$ be a linearly independent matrix. Then all the unnecessary operations can be eliminated thoroughly by the above proposed pruning algorithm.
Proof. We can prove this Lemma by induction on k; however, we omit the details due to page limit.

From Lemma 1, for a linearly independent matrix $A_{N\times N}$, we know that the unnecessary operations can be eliminated thoroughly when only the partial outputs of the matrix-vector multiplication $A_{N\times N} \times B_N$ are required. But, for a special pruning pattern, does there exist another scheme which can be used to further reduce the number of required operations. In the next corollary, we show that the number of required operations cannot be reduced by just utilizing the permutation technique. Moreover, for $C^{i}_{N\times N} \in$ PGD, the gain of pruning will not be changed even if we apply a different decomposition to $C^{i}_{N\times N}$. And this will be shown in Lemma 2.

Corollary 4. Let $A_{N\times N}$ be a linearly independent matrix and P_C be a permutation matrix. For any pruning pattern, on computing of the following expressions $A_{N\times N}\times B_N$, $(A_{N\times N}P_c)(P_c^{-1}B_N)$, and $P_c(P_c^{-1}A_{N\times N})B_N$, the simplification gains obtained from pruning the unnecessary operations will be the same.

Lemma 2. Let $A_{N\times N}$ be a linearly independent matrix. Based on the proposed pruning algorithm, the simplification gain will keep unchanged even though we apply a different decomposition to the matrix $C^{i}_{N\times N}$ $(\in \mathrm{PGD})$.

Proof. This lemma can be proved by examining all possible components of $C^{i}_{N\times N}$, we omit the details also due to page limit.

Therefore, in the proposed algorithm, for a given decomposition of the matrix $A_{N\times N}$, the simplification gain is always the same even if we have applied some modifications to the decomposition of the matrix $C^{i}_{N\times N}$. On the other hand, more effective decomposition of the matrix $A_{N\times N}$ is necessary if we want to obtain better simplification gain.

III. A Concrete Example

In general, the proposed general output pruning algorithm is suitable for the matrix-vector multiplication of any type: integer, real, or complex. Consider the matrix-vector multiplication of $A_{5\times5}\times B_5$ as an example, where

$$A_{5\times5}=\begin{bmatrix}1&0&0&1&0\\0&0&1&0&0\\1&0&0&0&1\\0&1&0&1&0\\1&0&1&0&0\end{bmatrix}\times\begin{bmatrix}-2.3&0&0&0&0\\0&5.0&0&0&0\\0&0&-0.4&0&0\\0&0&0&4.7&0\\0&0&0&0&3.2\end{bmatrix}\times\begin{bmatrix}0&1&0&1&0\\1&0&1&0&0\\0&0&0&1&0\\0&1&0&0&1\\1&1&0&0&0\end{bmatrix}$$

and B_5 is an arbitary 5×1 vector.

Let the above three decomposed matrices be denoted by $C^{a}_{5\times5}$, $C^{2}_{5\times5}$, and $C^{b}_{5\times5}$, respectively. Suppose only the last two entries of $A_{5\times5}\times B_5$ are required. According to the proposed approach, we set M= [255,255,255,0,0] and T=0, initially. Because $C^{a}_{5\times5}$ and $C^{b}_{5\times5}\in$ PGD, $C^{a}_{5\times5}$ and $C^{b}_{5\times5}$ can be respectively decomposed as

$$C^{a}_{5\times5}=\begin{bmatrix}0&0&0&1&0\\0&0&1&0&0\\0&0&0&0&1\\0&1&0&0&0\\1&0&0&0&0\end{bmatrix}\times\begin{bmatrix}1&0&1&0&0\\0&1&0&1&0\\0&0&1&0&0\\1&0&0&1&0\\1&0&0&0&1\end{bmatrix}\overset{\Delta}{=}C^{0}_{5\times5}\times C^{1}_{5\times5}$$

and

$$C^b_{5\times5} = \begin{bmatrix}1&0&1&0&0\\0&1&0&0&1\\0&0&1&0&0\\1&0&0&1&0\\1&0&0&0&1\end{bmatrix}\times\begin{bmatrix}0&1&0&0&0\\0&0&1&0&0\\0&0&0&1&0\\0&0&0&0&1\\1&0&0&0&0\end{bmatrix}\overset{\Delta}{=}C^3_{5\times5}\times C^4_{5\times5}.$$

Through the computation of $C^0_{5\times5}$ to that of $C^4_{5\times5}$, the values of each entry of the array M and threshold parameter T can be obtained gradually through the following five steps, as shown in Fig. 1.

Step 1: Since $C^0_{5\times5}\in$ P, the value of T in the encoding process is kept unchanged (=0) and the entries of M are permuted according to the permutation pattern defined by the inverse matrix of $C^0_{5\times5}$, say $(C^0_{5\times5})^{-1}$.

Step 2: Since $C^1_{5\times5}\in$ GD, the value of T is increased by 1 (=1). Since M[0] < 1 and M[2] = 255, we set M[2] =1 and since M[1] <1 and M[3]=255, thus we set M[3]=1.

Step 3: Since $C^2_{5\times5}\in$ D, the values of T and M are kept unchanged.

Step 4: Since $C^3_{5\times5}\in$ GD, the value of T is increased by 1 (=2). And, since M[1] < 2 and M[4] = 255, we set M[4] =2.

Step 5: Since $C^4_{5\times5}\in$ P, the value of T is kept unchanged and the entries of M are permuted according to $(C^4_{5\times5})^{-1}$.

$$\begin{bmatrix}255\\255\\255\\0\\0\end{bmatrix}\xrightarrow[C^0_{5\times5}]{T=0}\begin{bmatrix}0\\0\\255\\255\\255\end{bmatrix}\xrightarrow[C^1_{5\times5}]{T=0}\begin{bmatrix}0\\0\\1\\1\\255\end{bmatrix}\xrightarrow[C^2_{5\times5}]{T=1}\begin{bmatrix}0\\0\\1\\1\\255\end{bmatrix}\xrightarrow[C^3_{5\times5}]{T=1}\begin{bmatrix}0\\0\\1\\1\\2\end{bmatrix}\xrightarrow[C^4_{5\times5}]{T=2}\begin{bmatrix}2\\0\\0\\1\\1\end{bmatrix}T=2=T_f$$

Figure 1. The evolution of the array M and the threshold parameter T.

Now, the array M and the threshold parameter T can be used to facilitate the output pruning computation of the matrix-vector multiplication $A_{5\times5}\times B_5$ more efficiently. That is, the unnecessary operations can be eliminated thoroughly through the following five steps of the decoding process, as shown in Fig. 2.

Step 1: Since $C^4_{5\times5}\in$ P, the value of T is kept unchanged (=2) and the entries of M and the input vector B_5 are permuted according to the permutation pattern defined by the matrix $C_{5\times 5}^{4}$.

Step 2: Since $C_{5\times 5}^{3} \in$ GD and $M[4] \geq 2$, the 4-th output (marked as ? in Fig. 2) will be left out for pruning and the other outputs are kept for later computation. Moreover, the value of T is decreased by 1 (=1).

Step 3: Since $C_{5\times 5}^{2} \in$ D and $M[j] \leq 1$, $0 \leq j \leq 4$, the j-th outputs are necessary. The values of T and M are kept unchanged

Step 4: Since $C_{5\times 5}^{1} \in$ GD, $M[2] \geq 1$ and $M[3] \geq 1$, the 2nd and the 3rd outputs (marked as ? in Fig. 2) will be left out for pruning and only the 0-th and the 1st outputs are kept for later computation. Moreover, the value of T is decreased by 1 (=0).

Step 5: Since $C_{5\times 5}^{0} \in$ P, the value of T is kept unchanged (=0) and the entries of M and the input vector B_5^4 are permuted according to $C_{5\times 5}^{0}$.

The above example guarantees that all the unnecessary operations are eliminated thoroughly. The entry $B_5[j]$ of the input vector B_5 is redundant if the j-th entry of the array M is equal to 255, initially in Fig. 2.

$$\begin{bmatrix}2\\0\\0\\1\\1\end{bmatrix} \xRightarrow[C_{5\times 5}^{4}]{T=2} \begin{bmatrix}0\\0\\1\\1\\2\end{bmatrix} \xRightarrow[C_{5\times 5}^{3}]{T=2} \xRightarrow[\substack{M[j]<T\\ \text{done}}]{\substack{M[j]\geq T\\ \text{undone}}} \begin{bmatrix}0\\0\\1\\1\\2(?)\end{bmatrix} \xRightarrow[C_{5\times 5}^{2}]{T=1} \xRightarrow[\substack{M[j]\leq T\\ \text{done}}]{\substack{M[j]>T\\ \text{undone}}} \begin{bmatrix}0\\0\\1\\1\\2(?)\end{bmatrix} \xRightarrow[C_{5\times 5}^{1}]{T=1} \xRightarrow[\substack{M[j]<T\\ \text{done}}]{\substack{M[j]\geq T\\ \text{undone}}} \begin{bmatrix}0\\0\\1(?)\\1(?)\\2(?)\end{bmatrix} \xRightarrow[C_{5\times 5}^{0}]{T=0} \begin{bmatrix}1(?)\\1(?)\\2(?)\\0\\0\end{bmatrix} T=0$$

Figure 2. The unnecessary operations are pruned thoroughly through the decoding process of the proposed algorithm.

IV. The APPLICATION OF THE PROPOSED OUTPUT PRUNING ALGORITHM TO THE COMPUTATION OF PRUNING DISCRETE COSINE TRANSFORM

Since DCT is an orthogonal discrete transform, its transform kernel matrix must be a linearly independent matrix. That is, the pruning algorithm presented in section II can directly be applied to derive efficient pruning DCT algorithms. Moreover, all well-known DCT algorithms (such as [4-6]) and pruning DCT algorithms (such as [1-3]) can be modeled as a matrix-vector multiplication with known decompositions of the DCT transform kernel matrix.

Since the optimism of the proposed pruning algorithm is decomposition dependent, we can not only derive effective pruning DCT algorithms but also compare the effectiveness of matrix decomposition corresponding to each existing fast algorithm, by checking the complexities of the so-obtained pruning algorithms.

The following data are obtained by applying the proposed output pruning algorithm to derive efficient pruning DCT algorithms, based on the matrix decompositions presented in [1,5,6]. For the 1-D DCT of length 64, table 1 listed the numbers of required multiplications and additions for the corresponding pruning DCT algorithms with respect to different pruning patterns. The percentages of the required multiplications and additions of the so-obtained pruning algorithms to that the original (un-pruned) algorithms are also listed in table 1 (in the columns headed by the notation %), as an index for checking the effectiveness of each corresponding matrix decomposition, from the view point of output pruning.

The reason for choosing this special transform length comes from the fact that the numbers of required multiplications and additions for all the algorithms presented in [1,5,6] are the same and equal to 193 and 513, respectively.

The most well-known pruning DCT algorithm presented in [1] gives the same complexities as listed in the first column of table 1. This fact verifies the correctness and effectiveness of the proposed pruning algorithm. As for the other two algorithms (or matrix decompositions), the gain obtained from pruning is less significant. The number of pruned multiplications is larger in the Winograd's approach, whereas the number of pruned additions is larger in the Lee's approach. In fact, these characteristics can be observed and explained from their corresponding algorithm structures: In the Winograd's DCT algorithm, the required multiplications are post-processing oriented; whereas, in the Lee's DCT algorithm, the most post-processing oriented operations are additions. That is, if the complexity of multiplication is the major concern, then the pruning gain will be more significant when the required multiplications of the algorithm are nearly post-processing oriented.

V. Conclusions

In this paper, the derivation of efficient pruning DCT algorithms is the major focus. Since the effectiveness of the proposed output pruning algorithm is matrix-decomposition-dependent, some existing well-known DCT and pruning DCT algorithms are examined. Simulation results show that the resultant pruning DCT algorithm, derived based on the proposed approach, and the matrix decomposition presented in [1] needs the same computational complexity as that of the best existing pruning DCT algorithm [1] does (for some regular pruning patterns). From the derivation presented in section II, it follows that there is not any restriction on the pruning pattern for the proposed approach. In other words, the resultant pruning DCT algorithm can work as well as some well-known pruning algorithms, but it dispenses with the pruning pattern constraint that most of the other pruning approaches may have (such as the ones given in [1-3]).

VI. References

[1] ZhongDe Wang, "Pruning the Fast Discrete Cosine Transform," IEEE Trans. on Communication vol. 39 no. 5, May, pp. 640-643, 1991..

[2] Athanassiou N. Skodras, "Fast Discrete Cosine Transform Pruning,"

IEEE Trans. on Signal processing, vol. 42, no. 7, July, pp. 1833-1837, 1994.

[3] C. A. Christopoulos, J. Bormans, J. Cornelis, A. N. Skodras, "The Vector-Radix Fast Cosine Transform : Pruning and Complexity Analysis," Signal Processing, vol. 43, pp. 197-205, 1995.

[4] Hsieh S. Hou, "A Fast Recursive Algorithm for Computing the Discrete Cosine Transform," IEEE Trans. on Acoustics, Speech and Signal Processing, vol. Assp-35, no. 10, Oct. 10, pp. 1455-1461, 1987.

[5] B. G. Lee, "A New Algorithm to Compute the Discrete Cosine Transform," IEEE Trans. on Acoustics, Speech and Signal Processing, vol. Assp-32, no. 6, pp. 1243-1245, 1984

[6] Ephraim Feig, Shmuel Winograd, "Fast Algorithms for the Discrete Cosine Transform," IEEE Trans. on Signal Processing, vol. 40, no.9, Sep., pp. 2174-2193, 1992

The Referred Matrix Decompositions / Pruning Patterns	Z. Wang [1]		Winograd [6]		B. G. Lee [5]	
	No. of mults/ No. of adds	%	No. of mults/ No. of adds	%	No. of mults/ No. of adds	%
Only first 2 outputs are required	33/126	17.1/24.6	52/207	26.9/40.4	64/188	33.1/36.6
Only first 4 outputs are required	65/204	33.7/39.8	85/298	44.0/58.1	110/279	57.0/54.4
Only first 8 outputs are required	97/288	50.3/56.1	109/364	56.5/71.0	146/351	75.6/68.4
Only first 16 outputs are required	129/372	66.8/72.5	133/425	69.0/82.8	174/411	90.2/80.1
Only first 32 outputs are required	161/450	83.4/87.7	161/481	83.4/93.8	192/463	99.5/90.3

Table 1. The numbers of required multiplications and additions for the resultant pruning DCT algorithms with respect to different matrix decompositions and different pruning patterns.