Copyright © 1979 American Telephone and Telegraph Company
THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 58, No. 9, November 1979
Printed in U.S.A.

An Ordering Scheme for Facsimile Coding

By F. W. MOUNTS, E. G. BOWEN, and A. N. NETRAVALI

(Manuscript received June 19, 1979)

We give simulation results using one of our ordering algorithms for the coding of eight CCITT *test documents. These algorithms do not make any approximations and therefore can reproduce the documents exactly at the receiver. The code design is similar to the one-dimensional modified Huffman code that has been proposed by the* CCITT *as a standard. One-dimensional run-length coding using the modified Huffman code results in 445,316 bits per document on the average, which can be transmitted in 92.77 seconds using a transmission rate of 4800 bits per second. Our ordering algorithm, which is two-dimensional in nature, requires, on the average, 264,632 bits per document, or 55.13 seconds per document for the same transmission rate. Thus, the ordering scheme reduces the transmission time by approximately 41 percent, compared to one-dimensional run-length coding.*

I. INTRODUCTION

This paper presents simulation results using one of our ordering schemes[1, 2] for efficient coding of two-level (black-and-white) facsimile pictures using the eight CCITT (International Telegraph and Telephone Consultative Committee) test documents* as the source data. The scheme, we consider, allows exact reproduction of the documents at the receiver. Specifically, we give results on compression factors using a seven-element predictor, and a modified Huffman code for coding of the ordered data. It is assumed that every kth ($k = 2, 4, \infty$) line is encoded by a one-dimensional run-length code to limit the vertical propagation of the transmission errors. In our previous papers,[1–3] we gave entropies of run lengths of the ordered data using predictors and

* The eight CCITT study group XIV test documents are those used for the Graphics Coding Contest of the 1976 Picture Coding Symposium. Each image contains 2128 lines with 1728 pels per line. The scanning density in both directions is approximately 8 dots/mm.

ordering parameters specifically optimized for each document. In this paper, we give results when the predictor, the ordering parameters, and the Huffman code are obtained by averaging the relevant statistics for all eight CCITT documents.

In the basic ordering scheme, we make a prediction of the present element using the surrounding previously transmitted picture elements, and classify it as "good" or "bad," depending upon the probability of the prediction being in error, conditioned on the specific values of the surrounding picture elements. We then change the relative order of the prediction errors corresponding to picture elements along a scan line, using the "goodness" of the prediction in such a way as to increase the average run length of the black ("1") and/or white ("0") elements and then transmit the run lengths.

Several variations of the ordering algorithm have existed for some time.[4-6] We describe one specific variation of the algorithm, which we believe is simple to implement and can be easily extended to the coding of two-level pictures dithered to give the appearance of gray-level.[2] The results of computer simulations are given in Section III. They indicate that, on the average, a CCITT document can be transmitted with 264,632 bits, which would require 55.13 seconds using a transmission channel at 4800 bits per second and $k = \infty$. A one-dimensional modified Huffman code, on the other hand, would require 445,316 bits, on the average, and 92.77 seconds per document for the same transmission rate.* Thus, use of the two-dimensional ordering algorithm decreases the transmission time by 41 percent. Use of $k = 4$, however, increases the average bits per document to 307,310 and the average transmission time to 64.02 seconds.

II. CODING ALGORITHM

In this section, we describe our coding algorithm in detail. The coding algorithm consists of four steps: (*i*) prediction, (*ii*) ordering, (*iii*) dropping a specific sequence from transmission, and (*iv*) run-length coding of ordered data. We give details of each of these steps below.

2.1 Prediction algorithm

The first step in the ordering algorithm consists of making a prediction of the present picture element using the already-transmitted surrounding picture elements. We define a state S_i using the seven surrounding picture elements, A,B,C,D,E,F, and G, as shown in Fig. 1. There are 128 states. The predictor is developed in a standard way[1] as

* Without any compression techniques, each document requires 3,702,720 bits and can be transmitted in 12.86 minutes using a transmission rate of 4800 bits per second.

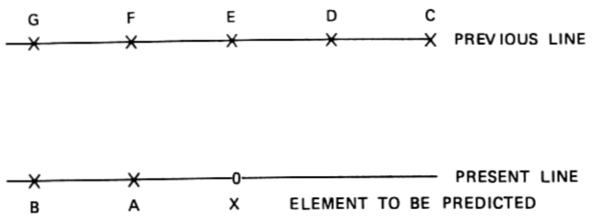

Fig. 1—The configuration of the seven elements constituting the state which is used to construct the predictor used for forward ordering.

the one which minimizes the probability* of making an error, given that a particular state has occurred. Thus, the prediction $C(S_i)$, for state S_i, is given by:

$$\begin{aligned} C(S_i) &= \text{“black,” if } P(X = \text{“black”} \mid S = S_i) \geq 0.5 \\ &= \text{“white,” otherwise,} \end{aligned}$$

where $P(.\,|\,.)$ is the experimentally determined conditional probability. An error in the prediction is denoted by "1" (black) and no error is denoted by "0" (white). A boundary of "0" is assumed wherever necessary to develop this prediction. This predictor varies from picture to picture. We have chosen the predictor to be the average over all the eight CCITT documents. Table I gives the relevant statistics and the predictor for each of the 128 states.

2.2 Coding

The ordering algorithm can be illustrated by considering a memory of 1728 cells (equal to the number of elements per line). Suppose the cells of this memory are numbered from 1 to 1728; we classify the states used for prediction into two categories, "good" or "bad." "Good" states are those for which the probability of the prediction being in error, conditional on that state, is less than or equal to a given threshold. All the other states are "bad." The classification of states into "good" and "bad" was based on the average over the eight CCITT documents using a goodness threshold of 0.90. This classification is given in Table I.

In the process of ordering, if the first element of the present line has a state classified as "good," then we put the value of the prediction error corresponding to it in memory cell 1; if, on the other hand, the state is classified as "bad," then we put the prediction error corresponding to it in memory cell 1728. We continue in this manner; the prediction error for the ith element of the present line is put in the unfilled memory cell of the smallest or the largest index, depending on

* This is computed by taking sample means over the eight CCITT documents.

Table I—Prediction, probability of correct prediction, and the "goodness" of the prediction for each state

Both forward and reverse ordering are considered with the appropriate definition of the state.

	Forward Ordering				Reverse Ordering			
No.	State Configuration	Prediction	Probability of Correct Prediction	Goodness (G or B)	State Configuration	Prediction	Probability of Correct Prediction	Goodness (G or B)
0	00000 00X	0	0.999	G	00000 X00	0	0.999	G
1	00000 01X	1	0.834	B	00000 X10	1	0.833	B
2	00000 10X	0	0.982	G	00000 X01	0	0.983	G
3	00000 11X	1	0.767	B	00000 X11	1	0.770	B
4	00001 00X	0	0.965	G	10000 X00	0	0.963	G
5	00001 01X	1	0.931	G	10000 X10	1	0.917	G
6	00001 10X	0	0.918	G	10000 X01	0	0.928	G
7	00001 11X	1	0.924	G	10000 X11	1	0.900	G
8	00010 00X	0	0.776	B	01000 X00	0	0.745	B
9	00010 01X	1	0.951	G	01000 X10	1	0.939	G
10	00010 10X	0	0.673	B	01000 X01	0	0.724	B
11	00010 11X	1	0.960	G	01000 X11	1	0.959	G
12	00011 00X	0	0.833	B	11000 X00	0	0.846	B
13	00011 01X	1	0.985	G	11000 X10	1	0.983	G
14	00011 10X	0	0.787	B	11000 X01	0	0.783	B
15	00011 11X	1	0.973	G	11000 X11	1	0.955	G
16	00100 00X	0	0.602	B	00100 X00	0	0.607	B
17	00100 01X	1	0.948	G	00100 X10	1	0.979	G
18	00100 10X	0	0.617	B	00100 X01	0	0.540	B
19	00100 11X	1	0.936	G	00100 X11	1	0.945	G
20	00101 00X	0	0.629	B	10100 X00	0	0.640	B
21	00101 01X	1	0.793	B	10100 X10	1	0.878	B
22	00101 10X	1	0.571	B	10100 X01	1	0.545	B
23	00101 11X	1	0.972	G	10100 X11	1	0.886	B
24	00110 00X	1	0.819	B	01100 X00	1	0.784	B
25	00110 01X	1	0.992	G	01100 X10	1	0.994	G
26	00110 10X	1	0.623	B	01100 X01	1	0.656	B

Table I—(*Continued*)

		Forward Ordering				Reverse Ordering		
No.	State Configuration	Prediction	Probability of Correct Prediction	Goodness (G or B)	State Configuration	Prediction	Probability of Correct Prediction	Goodness (G or B)
27	00110 11X	1	0.967	G	01100 X11	1	0.976	G
28	00111 00X	1	0.649	B	11100 X00	1	0.639	B
29	00111 01X	1	0.996	G	11100 X10	1	0.998	G
30	00111 10X	1	0.524	B	11100 X01	1	0.529	B
31	00111 11X	1	0.985	G	11100 X11	1	0.987	G
32	01000 00X	0	0.971	G	00010 X00	0	0.949	G
33	01000 01X	1	0.522	B	00010 X10	0	0.507	B
34	01000 10X	0	0.906	G	00010 X01	0	0.860	B
35	01000 11X	1	0.636	B	00010 X11	1	0.646	B
36	01001 00X	0	0.934	G	10010 X00	0	0.899	B
37	01001 01X	1	0.503	B	10010 X10	1	0.753	B
38	01001 10X	0	0.556	B	10010 X01	0	0.800	B
39	01001 11X	1	0.751	B	10010 X11	1	0.692	B
40	01010 00X	0	0.854	B	01010 X00	0	0.885	B
41	01010 01X	1	0.583	B	01010 X10	1	0.635	B
42	01010 10X	0	0.684	B	01010 X01	1	0.529	B
43	01010 11X	1	0.813	B	01010 X11	1	0.878	B
44	01011 00X	0	0.910	G	11010 X00	0	0.864	B
45	01011 01X	0	0.516	B	11010 X10	1	0.730	B
46	01011 10X	0	0.645	B	11010 X01	1	0.526	B
47	01011 11X	1	0.837	B	11010 X11	1	0.796	B
48	01100 00X	0	0.661	B	00110 X00	0	0.533	B
49	01100 01X	1	0.973	G	00110 X10	1	0.983	G
50	01100 10X	0	0.895	B	00110 X01	0	0.692	B
51	01100 11X	1	0.759	B	00110 X11	1	0.838	B
52	01101 00X	0	0.783	B	10110 X00	0	0.697	B
53	01101 01X	1	0.868	B	10110 X10	1	0.941	G
54	01101 10X	0	0.765	B	10110 X01	0	0.596	B
55	01101 11X	1	0.729	B	10110 X11	1	0.827	B

Table I—(*Continued*)

	Forward Ordering				Reverse Ordering			
No.	State Configuration	Prediction	Probability of Correct Prediction	Goodness (G or B)	State Configuration	Prediction	Probability of Correct Prediction	Goodness (G or B)
56	01110 00X	1	0.726	B	01110 X00	1	0.739	B
57	01110 01X	1	0.997	G	01110 X10	1	0.998	G
58	01110 10X	0	0.594	B	01110 X01	1	0.565	B
59	01110 11X	1	0.950	G	01110 X11	1	0.976	G
60	01111 00X	1	0.530	B	11110 X00	1	0.502	B
61	01111 01X	1	0.996	G	11110 X10	1	0.996	G
62	01111 10X	0	0.768	B	11110 X01	0	0.615	B
63	01111 11X	1	0.961	G	11110 X11	1	0.975	G
64	10000 00X	0	0.996	G	00001 X00	0	0.996	G
65	10000 01X	1	0.768	B	00001 X10	1	0.637	B
66	10000 10X	0	0.997	G	00001 X01	0	0.999	G
67	10000 11X	0	0.648	B	00001 X11	0	0.665	B
68	10001 00X	0	0.982	G	10001 X00	0	0.986	G
69	10001 01X	1	0.757	B	10001 X10	1	0.689	B
70	10001 10X	0	0.988	G	10001 X01	0	0.991	G
71	10001 11X	1	0.614	B	10001 X11	1	0.607	B
72	10010 00X	0	0.902	G	01001 X00	0	0.890	B
73	10010 01X	1	0.625	B	01001 X10	1	0.682	B
74	10010 10X	0	0.717	B	01001 X01	0	0.843	B
75	10010 11X	1	0.880	B	01001 X11	1	0.828	B
76	10011 00X	0	0.924	G	11001 X00	0	0.937	G
77	10011 01X	1	0.748	B	11001 X10	1	0.767	B
78	10011 10X	0	0.906	G	11001 X01	0	0.927	G
79	10011 11X	1	0.826	B	11001 X11	1	0.789	B
80	10100 00X	0	0.879	B	00101 X00	0	0.878	B
81	10100 01X	1	0.660	B	00101 X10	1	0.719	B
82	10100 10X	0	0.531	B	00101 X01	1	0.595	B
83	10100 11X	1	0.882	B	00101 X11	1	0.930	G
84	10101 00X	0	0.897	B	10101 X00	0	0.768	B

Table I—(*Continued*)

		Forward Ordering				Reverse Ordering		
No.	State Configuration	Prediction	Probability of Correct Prediction	Goodness (G or B)	State Configuration	Prediction	Probability of Correct Prediction	Goodness (G or B)
85	10101 01X	1	0.739	B	10101 X10	1	0.700	B
86	10101 10X	1	0.623	B	10101 X01	1	0.611	B
87	10101 11X	1	0.846	B	10101 X11	1	0.860	B
88	10110 00X	0	0.551	B	01101 X00	0	0.510	B
89	10110 01X	1	0.912	G	01101 X10	1	0.889	B
90	10110 10X	1	0.637	B	01101 X01	1	0.505	B
91	10110 11X	1	0.921	G	01101 X11	1	0.881	B
92	10111 00X	0	0.712	B	11101 X00	0	0.767	B
93	10111 01X	1	0.959	G	11101 X10	1	0.900	G
94	10111 10X	1	0.561	B	11101 X01	1	0.528	B
95	10111 11X	1	0.959	G	11101 X11	1	0.933	G
96	11000 00X	0	0.991	G	00011 X00	0	0.983	G
97	11000 01X	1	0.795	B	00011 X10	1	0.812	B
98	11000 10X	0	0.997	G	00011 X01	0	0.998	G
99	11000 11X	0	0.711	B	00011 X11	0	0.709	B
100	11001 00X	0	0.978	G	10011 X00	0	0.965	G
101	11001 01X	1	0.795	B	10011 X10	1	0.826	B
102	11001 10X	0	0.982	G	10011 X01	0	0.989	G
103	11001 11X	0	0.576	B	10011 X11	0	0.600	B
104	11010 00X	0	0.894	B	01011 X00	0	0.892	B
105	11010 01X	1	0.686	B	01011 X10	1	0.625	B
106	11010 10X	0	0.678	B	01011 X01	0	0.808	B
107	11010 11X	1	0.703	B	01011 X11	1	0.664	B
108	11011 00X	0	0.942	G	11011 X00	0	0.958	G
109	11011 01X	1	0.810	B	11011 X10	1	0.735	B
110	11011 10X	0	0.914	G	11011 X01	0	0.951	G
111	11011 11X	1	0.617	B	11011 X11	1	0.602	B
112	11100 00X	0	0.934	G	00111 X00	0	0.885	B
113	11100 01X	1	0.922	G	00111 X10	1	0.941	G

Table I—(*Continued*)

	Forward Ordering				Reverse Ordering			
No.	State Configuration	Prediction	Probability of Correct Prediction	Goodness (G or B)	State Configuration	Prediction	Probability of Correct Prediction	Goodness (G or B)
114	11100 10X	0	0.975	G	00111 X01	0	0.969	G
115	11100 11X	1	0.806	B	00111 X11	1	0.805	B
116	11101 00X	0	0.951	G	10111 X00	0	0.900	G
117	11101 01X	1	0.885	B	10111 X10	1	0.910	G
118	11101 10X	0	0.920	G	10111 X01	0	0.908	G
119	11101 11X	1	0.748	B	10111 X11	1	0.753	B
120	11110 00X	0	0.756	B	01111 X00	0	0.724	B
121	11110 01X	1	0.975	G	01111 X10	1	0.972	G
122	11110 10X	0	0.849	B	01111 X01	0	0.813	B
123	11110 11X	1	0.929	G	01111 X11	1	0.926	G
124	11111 00X	0	0.830	B	11111 X00	0	0.835	B
125	11111 01X	1	0.957	G	11111 X10	1	0.953	G
126	11111 10X	0	0.848	B	11111 X01	0	0.847	B
127	11111 11X	1	0.989	G	11111 X11	1	0.989	G

whether the state corresponding to the *i*th element is "good" or "bad." When the memory is filled, its cells are read in numerical order and the contents are run-length encoded. It is easy to see that the present line can be easily reconstructed from the knowledge of the run lengths of the ordered data, since the ordering sequence is known explicitly to the receiver.

2.3 Dropping of the sequence

The first sequence of the prediction errors of the type (0, 0, 0, ···, 0, 1) for each line is dropped. The following run is assumed to start with a "0." If it is indeed a run of "0," then it is assigned a proper run-length code. On the other hand, if the following run starts with a "1," a run of zero length of "0"s is transmitted first and then the code corresponding to the run of "1"s is transmitted. Two special cases ought to be mentioned. In one, if the prediction errors for a line generate a sequence 0, 0, 0, ···, 0, 1 (i.e., 1727 "0"s followed by a "1"), then this sequence is dropped and a special code word* (101011) is

* Identical to the second make-up code for the "0" codebook.

sent. In the second case, if the prediction errors for a line generate all "0"s, then the sequence is dropped and no special code word is transmitted except in a situation where the ambiguity with the end of message exists (i.e., occurrence of six consecutive end-of-line codes). In such a case, a special code word* (00110) is transmitted after the fifth consecutive end-of-line code. The length of the dropped sequence can be derived from the length of the rest of the decoded data in a line since the number of samples in a line is fixed and known.

2.4 *Coding of run lengths*

Each line is ordered from either left to right (forward) or from right to left (reverse), depending upon which direction requires the least number of coded bits. In the forward as well as the reverse ordering mode, the prediction and the "goodness" of a state are based on the average over all eight CCITT documents when ordered from left to right or right to left, respectively. The prediction and the "goodness" of a state for the reverse ordering mode are shown in Table I. A flag bit is used to define the direction of ordering to the receiver and is transmitted as the first bit of each line of coded data. The code for the run lengths for both modes of operation is based on the average over all the eight CCITT documents using the forward ordering mode. Only two sets of codes are used for coding the run lengths, one for run lengths of "0" and another for run lengths of "1." The structure of the codebook for the runs of "0" is similar to the standard one-dimensional code agreed upon by the CCITT. It uses a make-up code [MU] (when necessary) followed by a single terminating code [TC]. There are 64 terminating code words and 27 make-up code words. This codebook is given in Table II. The codebook for the runs of "1" consists of 10 terminating code words and a single make-up code word, which may be repeated a number of times depending on the length of the run being coded. This codebook is described in Table III. Both of these codebooks are constructed so that no combination of the code words of runs of "0" and "1" can result in a sequence of coded bits identical to the end-of-line code, which is taken to be the same 12-bit word specified by the CCITT (000000000001).

III. SIMULATION RESULTS

In this section, we give results of computer simulations using the algorithm described in the previous section, along with its variations. Also, for comparison, we give results for the standard one-dimensional Huffman code proposed by the CCITT. This is given in Table IV. Table IV shows that, on the average, a CCITT document can be transmitted

* Identical to the first make-up code for the "0" codebook.

Table II—Make-up and terminating codes for runs of "0"

A run of "0" is coded by using a make-up code (whenever necessary) and a terminating code, depending on the run length.

Terminating Codes for "0"

"0" Run Length	Code Word	"0" Run Length	Code Word
0	01110111	32	00101010
1	11	33	10100110
2	010	34	10101000
3	100	35	011110101
4	0001	36	011110010
5	1011	37	011101101
6	01101	38	011001110
7	00111	39	011001010
8	011111	40	011001001
9	011100	41	011001011
10	000001	42	011001000
11	101000	43	000000101
12	0111010	44	000011010
13	0110000	45	000000001
14	0000100	46	001001000
15	0010111	47	001011001
16	1010010	48	001000011
17	01111000	49	101001111
18	01100110	50	001000010
19	01100010	51	001011000
20	00001100	52	0111101110
21	00001011	53	0111101111
22	00001010	54	101010011
23	00000001	55	101001110
24	00000011	56	0111101101
25	00100111	57	0111100111
26	00100110	58	101010110
27	00100101	59	101010010
28	00100011	60	0111101001
29	00100000	61	0111101000
30	00100010	62	101010111
31	00101011	63	001011011

Make-Up Codes for "0"

"0" Run Length	Code Word	"0" Run Length	Code Word
64	00110	960	0010110101
128	101011	1024	01100111110
192	0000111	1088	01100111101
256	0010100	1152	01100111111
320	01100011	1216	01100111100
384	10101010	1280	000000000111
448	011101100	1344	000000000100
512	000000100	1408	000000000110
576	001001001	1472	0000000001011
640	0111101100	1536	00000000010100
704	0111100110	1600	000000000101011
768	0000110110	1664	0000000001010101
832	0000110111	1728	0000000001010100
896	0010110100		

Table III—Make-up and terminating codes for runs of "1"

Only one make-up code and 10 terminating code words are used. A make-up code may be repeated several times, depending upon the run length being coded.

Runs of "1" are coded in the form of

$$[\text{MU}], \cdots [\text{MU}]^i, [\text{TC}], \qquad 0 \le i \le 172,$$

where i represents the number of times the make-up code [MU] is transmitted. For a given run length (RL), the value of i is chosen such that

$$10\, i < \text{RL} \le 10\,(i+1).$$

The terminating code is the word assigned to (RL-10i) as follows:

(RL-10i)	Terminating Code Word
1	1
2	01
3	001
4	0001
5	00001
6	0000010
7	00000110
8	0000011110
9	00000111110
10	00000111111

The make-up code is assigned the binary word (000001110).

with 445,316 bits, which would take 92.77 seconds at a transmission rate of 4800 bits per second.*

The results for the two variations of the ordering scheme are given in Table V. In both, so as to limit the propagation of transmission errors in the reconstructed image, we employ the technique of coding every kth line of the original data using the standard one-dimensional modified Huffman code. This kth line may be coded by omitting the first sequence of 0, 0, 0, $\cdots$, 0, 1 or it can be coded in its entirety. Table V gives the results when the first sequence is omitted, whereas Table VI gives the results when the first sequence is transmitted.

Additional results are given in Tables V and VII for the case when the transmission time of each line is constrained to be at least 0 ms, 5 ms, or 10 ms; i.e., 0, 24, or 48 coded bits per line using a transmission rate of 4800 bits per second. When the transmission time per line is lower bounded, for example, by 5 ms and if the number of coded bits (including the end-of-line bits) for any particular line is less than 24, then fill bits are used to make up the difference. The fill bits are taken to be a string of all "0"s and are inserted prior to the end-of-line code. Table VII gives results when the best ordering direction is switched between left to right or right to left depending on which requires the smaller number of coded bits, and this switching is specified to the

* These figures include the bits required for the end-of-line code (=2128 × 12).

Table IV—Compression results using one-dimensional run-length coding with the modified Huffman code proposed by CCITT

Total coded bits and the time required to transmit include the end-of-line code and start- and end-of-message codes. No constraint on the transmission time for each line is imposed.

CCITT Image No.	Total Coded Bits	Time Required (in Seconds) to Transmit Using a 4800-bps Rate
1	220745	45.99
2	238521	49.69
3	417306	86.94
4	682195	142.12
5	430259	89.64
6	353650	73.68
7	757804	157.88
8	462051	96.26

Average total bits per document = 445,316.
Average transmission time per document = 92.77 seconds.

receiver by a flag bit transmitted as a first bit of coded data. A flag bit is also used for every kth line.

The simulation results show that both the forward and reverse ordering give approximately the same compression, with a slight preference for the forward ordering when $k = 4$ and $k = \infty$ and the reverse ordering when $k = 2$. Adaptive ordering, on the other hand, does better than either the forward or the reverse ordering for all values of k and all cases of fill that we studied. The approximate improvement is close to a second of transmission time per document by using adaptive ordering. Using adaptive ordering and 0-ms constraint on the transmission time, $k = 4$ results in a 16-percent increase in the transmission time, whereas $k = 2$ results in a 32-percent increase in the transmission time compared to $k = \infty$. However, with adaptive ordering and $k = 2$, there is still a 21-percent decrease in transmission time compared to one-dimensional run-length coding using the modified Huffman code. When $k = 2$ or $k = 4$, the scan lines that are coded by a simple run-length code may be transmitted with or without the first sequence of the form $0, 0, 0, \cdots, 0, 1$. In the case of adaptive ordering, dropping of the first sequence for the kth line decreases the transmission time by approximately one second compared to not dropping the first sequence. Obviously, this decrease is greater for $k = 2$ than for $k = 4$. The constraint on the transmission time for each line increases the overall transmission time of the document. Using the constraints of 10 ms, the overall transmission time is increased over the case of 5 ms by 4.1 seconds, for $k = \infty$.

Table V—Compression results with forward and reverse ordering

The numbers corresponding to the transmission time constraint of 0 ms do not include the end-of-line codes, but they do include bits necessary for start and end of message; 25,536 bits are necessary for transmitting the end-of-line code for each document. The first sequence of the form 0,0,0,· · ·,0,1 is dropped in all cases.

	Forwarding Ordering			Reverse Ordering		
	0 ms	5 ms	10 ms	0 ms	5 ms	10 ms
Image 1:						
$k = 2$	160196	199922	232725	159072	198811	231720
$k = 4$	146431	186202	219729	144999	184795	218433
$k = \infty$	132156	171939	206220	130257	170088	204542
Image 2:						
$k = 2$	140564	171076	187869	140778	171328	188192
$k = 4$	111380	142486	162718	111284	142446	162959
$k = \infty$	82405	114134	137590	81777	113597	137564
Image 3:						
$k = 2$	289670	320459	336055	289130	320005	335721
k = 4	242191	273657	292645	242606	274195	293293
k = ∞	195198	227315	249424	195792	228084	250465
Image 4:						
$k = 2$	578680	609552	623370	573570	604490	618202
$k = 4$	540474	571256	585478	535034	565900	580016
$k = \infty$	503548	534304	549021	496238	527082	541684
Image 5:						
$k = 2$	309949	340302	354977	308335	338726	353319
$k = 4$	267979	299010	316769	265914	297040	314716
$k = \infty$	224792	256533	277408	222687	254598	275230
Image 6:						
$k = 2$	217758	248646	263847	217713	248664	263591
$k = 4$	167956	198841	215302	167673	198689	214851
$k = \infty$	118565	149499	167399	118460	149596	167137
Image 7:						
$k = 2$	632662	661074	669233	634047	662572	670568
$k = 4$	587417	615803	623959	590930	619562	627638
$k = \infty$	542808	571182	579348	548127	576883	585092
Image 8:						
$k = 2$	294094	321536	329539	298915	324360	332304
$k = 4$	225967	253686	263471	231735	259451	269049
$k = \infty$	157859	185779	197170	166726	194666	205892
	Average Total Bits per Document					
$k = 2$	327947	359071	374702	327695	358620	374202
$k = 4$	286224	317618	335009	286272	317760	335119
$k = \infty$	244666	276336	295448	245008	276824	295951
	Average Transmission Time per Document (in Seconds)					
$k = 2$	68.32	74.81	78.06	68.27	74.71	77.96
$k = 4$	59.63	66.17	69.79	59.64	66.20	69.82
$k = \infty$	50.97	57.57	61.55	51.04	57.67	61.66

The least bits are required for the ordering scheme with $k = \infty$, 0-ms transmission time constraint, and adaptive ordering. This results in 55.13 seconds of transmission time, on the average, which is about 41 percent less than that required for one-dimensional run length with the modified Huffman code. However, a reasonable alternative, considering the effect of transmission errors, would be for $k = 4$; and, in this case, the average transmission time is 64.02 seconds, which is 31 percent less than the one-dimensional run-length code. It should be pointed out that the effect of transmission errors has not been evaluated.

Table VI—Compression results for forward, reverse, and adaptive ordering

Each case does not include bits necessary for the end-of-line code (2128 × 12) but does include bits necessary for start and end of message. There is no constraint on the minimum transmission time for each line. Adaptive ordering results include the bits necessary for specifying the direction of the ordering. The first sequence of the form 0,0,0,· · ·,0,1 is not dropped for the kth line ($k = 2, 4$).

	Forward Ordering	Reverse Ordering	Adaptive Ordering
Image 1:			
$k = 2$	163476	162432	163252
$k = 4$	148027	146630	146747
$k = \infty$	132156	130257	129683
Image 2:			
$k = 2$	147603	147247	145915
$k = 4$	114940	114530	111501
$k = \infty$	82405	81777	77120
Image 3:			
$k = 2$	293211	293067	292352
$k = 4$	243960	244562	241912
$k = \infty$	195198	195792	191804
Image 4:			
$k = 2$	583396	579372	578165
$k = 4$	542996	537947	534577
$k = \infty$	503548	496238	491533
Image 5:			
$k = 2$	315514	314772	313800
$k = 4$	270788	269107	266576
$k = \infty$	224792	222687	218542
Image 6:			
$k = 2$	223402	223495	222072
$k = 4$	170847	170636	167751
$k = \infty$	118565	118460	113778
Image 7:			
$k = 2$	638451	641133	635789
$k = 4$	590304	594525	585217
$k = \infty$	542808	548127	535221
Image 8:			
$k = 2$	297418	300717	295936
$k = 4$	227666	233657	225470
$k = \infty$	157859	166726	155089
	Average Total Bits per Document		
$k = 2$	332809	332779	330910
$k = 4$	288691	288949	284969
$k = \infty$	244666	245008	239096
	Average Transmission Time per Document (in Seconds)		
$k = 2$	69.34	69.33	68.94
$k = 4$	60.14	60.20	59.37
$k = \infty$	50.97	51.04	49.81

Table VII—Compression results with adaptive ordering

The first run of the form 0,0,0,· · ·,0,1 of each line is dropped. The adaptive ordering includes the bits necessary to specify the direction of the ordering. In the case of 0 ms, bits necessary for end-of-line codes are not included.

	0 ms	5 ms	10 ms
	Adaptive Ordering, Total Bits		
Image 1:			
$k = 2$	159426	197893	230952
$k = 4$	144889	183421	217286
$k = \infty$	129683	168244	202963
Image 2:			
$k = 2$	137856	168016	185731
$k = 4$	107433	138182	159785
$k = \infty$	77120	108503	133731
Image 3:			
$k = 2$	286949	317318	333089
$k = 4$	239228	270222	289416
$k = \infty$	191804	223424	245902
Image 4:			
$k = 2$	571272	601723	615573
$k = 4$	531166	561568	575811
$k = \infty$	491533	521523	536679
Image 5:			
$k = 2$	306399	336372	351292
$k = 4$	262874	293520	311645
$k = \infty$	218542	249897	271152
Image 6:			
$k = 2$	215791	246276	261758
$k = 4$	164517	195059	211925
$k = \infty$	113778	144418	162820
Image 7:			
$k = 2$	627383	655650	663814
$k = 4$	580995	609353	617502
$k = \infty$	535221	563682	571841
Image 8:			
$k = 2$	291264	318521	326718
$k = 4$	223086	250612	260626
$k = \infty$	155089	182816	194551
	Average Number of Bits per Image		
All Images:			
$k = 2$	324543	355221	371116
$k = 4$	281774	312742	330500
$k = \infty$	239096	270313	289955
	Average Transmission Times (in Seconds)		
All Images:			
$k = 2$	67.61	74.00	77.32
$k = 4$	58.70	65.15	68.85
$k = \infty$	49.81	56.32	60.41

REFERENCES

1. A. N. Netravali, F. W. Mounts, and E. G. Bowen, "Ordering Techniques for Coding of Two-Tone Facsimile Pictures," B.S.T.J., *55*, No. 10 (December 1976), pp. 1539–1552.
2. A. N. Netravali, F. W. Mounts, and J.-D. Beyer, "Techniques for Coding Dithered Two-Level Pictures," B.S.T.J., *56*, No. 5 (May–June 1977), pp. 809–819.
3. F. W. Mounts, A. N. Netravali, and K. A. Walsh, "Some Extensions of the Ordering Techniques for Compression of Two-Level Facsimile Pictures," B.S.T.J., *57*, No. 8 (October 1978), pp. 3057–3067.

4. D. Preuss, "Comparison of Two-Dimensional Facsimile Coding Schemes," International Conference on Communications, June 1975, pp. 7.12–7.16.
5. Y. Yasuda and K. Arai, "Facsimile Data Compression by Rearranging Picture Elements," 1977 Picture Coding Symposium, Tokyo, Japan.
6. Y. Ueno, F. Ono, T. Semasa, S. Tomita, and R. Ohnishi, "Comparison of Facsimile Data Compression Methods," ICC '78 Conference Record, *3*, June 4–7, 1978, pp. 48.4.1–48.4.6.