Convolutional Code Search for Good Permutation Trellis Codes using M-ary FSK

Theo G. Swart*, A. J. Han Vinck† and Hendrik C. Ferreira*
*Department of Electrical and Electronic Engineering Science,
University of Johannesburg, Auckland Park, 2006, South Africa
Email: ts@ing.rau.ac.za, hcferreira@uj.ac.za
†Institute for Experimental Mathematics,
University Duisburg-Essen, Essen 45326, Germany
Email: vinck@iem.uni-due.de

***Abstract*— Permutation trellis codes combined with M-ary FSK are considered with the aim of combating additive noise, impulse noise and permanent frequency disturbances. New permutation trellis codes are presented that were found by exhaustive computer search, considering the free distance and the distance spectrum of the codes. Until now the best binary convolutional codes were considered to be the best candidates to use in mappings for trellis codes, however the results will show that this is not always the case. Simulation results are also presented, showing that in most cases the new permutation trellis codes are performing better than the previous codes, although the gains attained are small.**

***Keywords*— Channel coding, convolutional codes, frequency-shift keying (FSK), interference suppression.**

I. Introduction

PERMUTATION codes in conjunction with M-ary frequency shift keying [1] possess error correcting capabilities suitable for the noise types encountered in power-line communications. The time-diversity and frequency-diversity of permutation codes ensure that additive noise, impulse noise and permanent frequency disturbances can be corrected. By using distance-preserving mappings (DPMs), binary convolutional codes and permutation codes can be combined to form permutation trellis codes [2], [3]. The mapping is used to map from the binary convolutional codes' output bits to the codewords of a permutation code, guaranteeing that the distance between binary outputs will be preserved between the corresponding permutation codewords.

Much research has been done on finding mapping algorithms [4]–[8] and even optimum DPMs [9], [10], but little has been done in finding the optimum convolutional codes for these mappings. Using an exhaustive computer search the best permutation trellis codes for various mappings are found, in some cases these codes are obtained for binary codes that are much weaker than the best binary convolutional code for the same parameters.

In Section II we provide an introduction to permutation trellis codes and the system used. Section III discusses the exhaustive computer search used, as well as presenting the new codes that were found. Section IV presents simulation results, showing the performance of the previous and new codes when used in a noisy PLC environment.

II. Permutation trellis codes and M-ary FSK

A brief description of permutation trellis codes with M-FSK will be provided. For a more detailed description the reader can refer to [3] or [11]. Because of the low overall rate of these codes, it is envisaged that it could be used when high throughput is not necessary and absolute reliable communication is required, as in control systems or security systems.

A permutation code of length M is used, where each of the M symbols is assigned to one of the M frequencies in the M-FSK modulator. Each symbol is then sent in time as the corresponding frequency, giving the transmitted signal a constant envelope. A modified envelope detector is used for each frequency, outputting 1 when the signal envelope is above a certain threshold and 0 otherwise. This results in an $M \times M$ matrix which is used in the Viterbi decoding to compare with the branches of the trellis. In Section IV we describe how this matrix is affected by the noise on the channel.

As stated earlier, a permutation trellis code is created when the n-length outputs of a binary convolutional code is mapped to M-length permutation codewords. If x_i is the i-th binary output and y_i is the corresponding i-th permutation codeword in the mapping, then we require that $d_H(x_i, x_j) \leq d_H(y_i, y_j) + \delta$, $\forall i \neq j$, where d_H is the Hamming distance and δ is an integer determining the mapping type. Distance-conserving mappings are obtained for $\delta = 0$, distance-increasing mappings for $\delta > 0$ and distance-reducing mappings for $\delta < 0$. As an example, a mapping with $n = 2$, $M = 3$ and $\delta = 1$ is $\{00, 01, 10, 11\} \rightarrow \{123, 132, 213, 231\}$. Using this mapping in conjunction with an $R = 1/2$, $m = 2$, $d_{\text{free}} = 5$, convolutional code, a permutation trellis code with $d'_{\text{free}} = 8$ is obtained. Fig. 1 shows the state diagrams for the binary convolutional code and the permutation trellis code.

In many instances in a mapping one will find that $d_H(x_i, x_j) < d_H(y_i, y_j) + \delta$ for some $i \neq j$, resulting in the distance exceeding the minimum requirement. How this additional distance affect the mappings has been investigated in [9]. However, until now it has not been investigated how this additional distance affect the overall permutation trellis code.

1-4244-1090-8/07/$25.00 ©2007 IEEE.

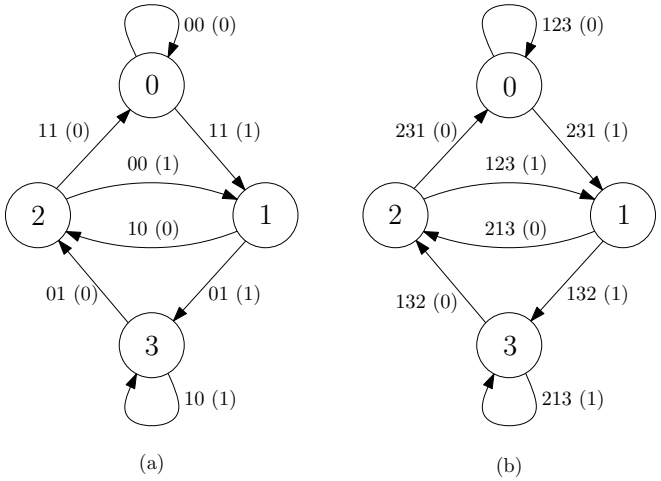

Fig. 1. State systems for (a) binary convolutional base code and (b) permutation trellis codes

III. Exhaustive computer search

The free distance of the trellis code is used as criterion to select better codes. A motivation why Hamming distance is a suitable metric for this setup can be found in [3].

To find better permutation trellis codes, all the possible connections in the encoder are investigated, for a certain set of parameters (any set of connections that lead to a catastrophic code is discarded) and the free distance and the first few components of the distance spectrum is determined. Since the permutation trellis code is not linear, as in the case of a binary convolutional code, an exhaustive search must be done of the entire trellis, i.e. paths leaving every possible state must be compared with all possible paths merging in any other state. For binary convolutional codes only paths leaving and merging with the all-zeros path have to be compared. Several efficient algorithms are also known to calculate these values for binary codes, such as the FAST algorithm [12]. It is an open question whether similar algorithms would be possible for the permutation trellis codes.

To simplify the search, the permutation trellis code's free distance and distance spectrum is first calculated using the best known binary convolutional code. The free distance and distance spectrum obtained for this case is then used as a benchmark in the exhaustive search. Any code that does worse is immediately discarded and the search proceeds to the next candidate. The best code is chosen based on the code with highest free distance. If there are several codes with the same highest free distance, the code with the lowest first component of the distance spectrum is chosen. If necessary, the second component is considered, and so forth. However, the results will show that this method is too simplistic, especially when there is an order magnitude difference in any of the later components. This will be discussed further in the next section.

A preliminary investigation into further simplifying the search has shown that only comparing paths starting from one state and ending in any other state produces the same codes as a search done when considering all states. A complete study of the structure and possible symmetry of the trellis would be necessary to conclude whether this would result in the best code for all cases. For large trellises this is a viable alternative, as it reduces the number of paths to compare by 2^m, where m is the constraint length of the encoder.

In Table I, Table II, Table III and Table IV we present the previous codes and the new codes found for $R = 1/2$, $R = 2/3$, $R = 1/3$ and $R = 1/4$ binary convolutional codes respectively. All generator sequences are given in octal format. For comparison we present the free distance of the trellis code, d'_{free}, as well as the free distance, d_{free}, of the binary convolutional code that is used. The number of paths with a distance d, when compared to any other path, is represented by N_d. Note that this spectrum is for the simplified case where all the paths starting from one state is considered. The codes that are best according to the search parameters are marked, but as stated earlier, these do not always produce the best results. Specific codes will be discussed in the next section.

The full mappings will not be presented, because of space constraints, but these can be found in the references:

- $Q(2, 3, 1)$, $n = 2 \rightarrow M = 3$ increasing mapping [3],
- $Q(3, 4, 1)$, $n = 3 \rightarrow M = 4$ increasing mapping [3],
- $\mathcal{M}(3, 4, 1)$, $n = 3 \rightarrow M = 4$ increasing mapping [10],
- $Q(4, 4, 0)$, $n = 4 \rightarrow M = 4$ conserving mapping [3],
- $Q'_3(4, 4, 0)$, $n = 4 \rightarrow M = 4$ conserving mapping [3].

Since the mapping algorithms from [4]–[8] are only for conserving mappings, and only start to differ from the above ones for $M > 5$, these were not considered.

In the tables we compare $Q(3, 4, 1)$ to an optimized $\mathcal{M}(3, 4, 1)$ and $Q(4, 4, 0)$ to an optimized $Q'_3(4, 4, 0)$, using the sum of the Hamming distances in the mapping as the optimization criterion [10]. In all cases the optimized mappings result in permutation trellis codes with better free distances than those using the other mapping. Because of this we only consider the optimized mappings in the simulation results.

The search also revealed that the symmetry that exists in binary convolutional codes does not exist in the permutation trellis codes, e.g. a binary encoder with generator $(3\ 3\ 1)$ produce the same code as an encoder with generator $(1\ 3\ 3)$. The non-linearity of the mappings causes the permutation trellis codes to lose this property. This can cause different generators for the same good binary code to produce permutation trellis codes with different error correcting capabilities.

IV. Simulation results

We will use the same simple error model that was used previously [11] to evaluate mappings. Errors are generated in the received matrix according to certain error parameters, as follows:

- *background noise* – each element in the received matrix has a probability, p_b, of being in error, i.e. a zero is changed to a one, or vice versa,
- *impulse noise* – each column in the received matrix has a probability, p_i, of resulting in an impulse noise, where the entire column's elements is set to ones,

- *permanent frequency disturbance* – all received matrices will have ones in the row that corresponds to the frequency disturbance.

The error parameters were assumed to be equal for all frequency sub-bands.

In all the results a single permanent frequency disturbance is placed on the second frequency and impulse noise with a probability $p_i = 10^{-1}$ is present. The background noise probability is then varied and the bit error rate is determined for each case. In Figs. 2, 3, 4 and 5 we present the performance results for the $R = 1/2$, $R = 1/3$, $R = 1/4$ and $R = 2/3$ codes respectively.

In the $R = 1/2$, $m = 1$ and $m = 2$ case, the search resulted in the same generator sequence as that of the best binary convolutional code. For the $m = 3$ case, a new code with $d'_{\text{free}} = 10$ was found and a first distance component that is lower than the previous code, with $N_{10} = 14$ compared to $N_{10} = 30$. The results in Fig. 2 show that the new code is performing much worse than the previous one, and only slightly better than the $m = 2$ code. A possible reason for this is the high second distance component, with $N_{11} = 472$ compared to $N_{11} = 14$. The bit error rate is affected by the number of paths with a certain distance and the probability of more errors occurring than that distance can correct. The larger the number of paths, the greater the possibility is of decoding incorrectly. Also, the higher distance components have a greater effect with high error probabilities, as in the case of our simulation results. For very low error probabilities the effect of the higher distance components diminishes and the effect of the first component will be more dominant. For $m = 4$ a new code with a higher free distance was found, but in this case it was found that error propagation causes it to perform slightly worse than the previous code.

Fig. 3 shows all the new codes using $R = 1/3$ codes performing better than the previous ones. In the $m = 1$ and $m = 2$ cases the previous codes start to perform better as the error probability lowers. Note in Table III that the previous and new code for $\mathcal{M}(3, 4, 1)$ and $m = 3$ both have $d'_{\text{free}} = 16$, but that the binary convolutional codes used as base codes have $d_{\text{free}} = 10$ and $d_{\text{free}} = 7$ respectively. The additional distance obtained from the mapping resulted in a weaker binary convolutional code producing a better permutation trellis code.

The new codes using $R = 1/4$ codes in Fig. 4 show a slight improvement as the error probability lowers, except the new $m = 1$ code that starts to coincide with the previous code.

Finally, Fig. 5 shows the results for the trellis codes using $R = 2/3$ codes. A new code was not found for $m = 1$ and the difference between the $m = 2$ codes are negligible. For $m = 3$, the previous code and new code have free distances of 12 and 14 respectively. However, the new code is performing much worse than the previous code. It was again found in this case that error propagation causes this to happen.

The additional distance in mappings increases as M increases, suggesting that better codes should be found for larger M values. Also, for convolutional codes with longer constraint lengths, this additional distance contributes significantly as it takes longer for paths to merge, as in the case of the $R = 1/3$, $m = 3, 4$ and $R = 1/4$, $m = 3$ codes. Both of these cases where additional distance can be obtained however result in increased complexity in the trellis, making exhaustive searches very time consuming.

V. Conclusion

By using an exhaustive computer search, we have found permutation trellis codes that have better error correcting performance than previous codes in some cases, and our simulation results have corroborated this. However, in other cases the performance difference between the previous code and the new code is negligible, or error propagation causes the new code to perform worse. Also, the distance spectra found differs from those of binary codes, causing the search criteria to be too simplistic. However, these results warrant a further study in which improved search criteria are used and error propagation is taken into account.

Finding good permutation trellis codes thus far consisted of finding good mappings for a specific binary convolutional code, or, as we did in this paper, finding good binary convolutional codes for a specific mapping. Finding the best code would thus involve an exhaustive search of all possible mappings combined with all possible convolutional codes, which is impractical. The next step would be to design permutation trellis codes directly, without using a binary convolutional code as an intermediate step, to see if better codes can be obtained.

References

[1] A. J. H. Vinck, "Coded modulation for powerline communications," *Proc. Int. J. Elec. Commun.*, vol. 54, no. 1, pp. 45–49, 2000.

[2] H. C. Ferreira and A. J. H. Vinck, "Interference cancellation with permutation trellis codes," in *Proc. IEEE Veh. Technol. Conf. Fall 2000*, Boston, MA, Sep. 2000, pp. 2401–2407.

[3] H. C. Ferreira, A. J. H. Vinck, T. G. Swart and I. de Beer, "Permutation trellis codes," *IEEE Trans. Commun.*, vol. 53, no. 11, p. 1782–1789, Nov. 2005.

[4] J.-C. Chang, R.-J. Chen, T. Kløve and S.-C. Tsai, "Distance-preserving mappings from binary vectors to permutations," *IEEE Trans. Inf. Theory*, vol. 49, no. 4, pp. 1054–1059, Apr. 2003.

[5] K. Lee, "New distance-preserving mappings of odd length," *IEEE Trans. Inf. Theory*, vol. 50, no. 10, pp. 2539–2543, Oct. 2004.

[6] J.-C. Chang, "Distance-increasing mappings from binary vectors to permutations," *IEEE Trans. Inf. Theory*, vol. 51, no. 1, pp. 359–363, Jan. 2005.

[7] K. Lee, "Cyclic constructions of distance-preserving maps," *IEEE Trans. Inf. Theory*, vol. 51, no. 12, pp. 4392–4396, Dec. 2005.

[8] K. Lee, "Distance-increasing mappings of all lengths by simple mapping algorithms," *IEEE Trans. Inf. Theory*, vol. 52, no. 7, pp. 3344–3348, Jul. 2006.

[9] T. G. Swart, I. de Beer and H. C. Ferreira, "On the distance optimality of permutation mappings," in *Proc. Int. Symp. Inf. Theory*, Adelaide, Australia, Sep. 4–9, 2005, pp. 1068–1072.

[10] T. G. Swart and H. C. Ferreira, "A generalized upper bound and a multilevel construction for distance-preserving mappings," *IEEE Trans. Inf. Theory*, vol. 52, no. 8, pp. 3685–3695, Aug. 2006.

[11] T. G. Swart, I. de Beer, H. C. Ferreira and A. J. H. Vinck, "Simulation results for permutation trellis codes using M-ary FSK," in *Proc. Int. Symp. on Power Line Commun. and its Applications*, Vancouver, BC, Canada, Apr. 6-8, 2005, pp. 317-321.

[12] M. Cedervall and R. Johannesson, "A Fast algorithm for computing distance spectrum of convolutional codes," *IEEE Trans. Inf. Theory*, vol. 35, no. 6, pp. 1146–1159, Nov. 1989.

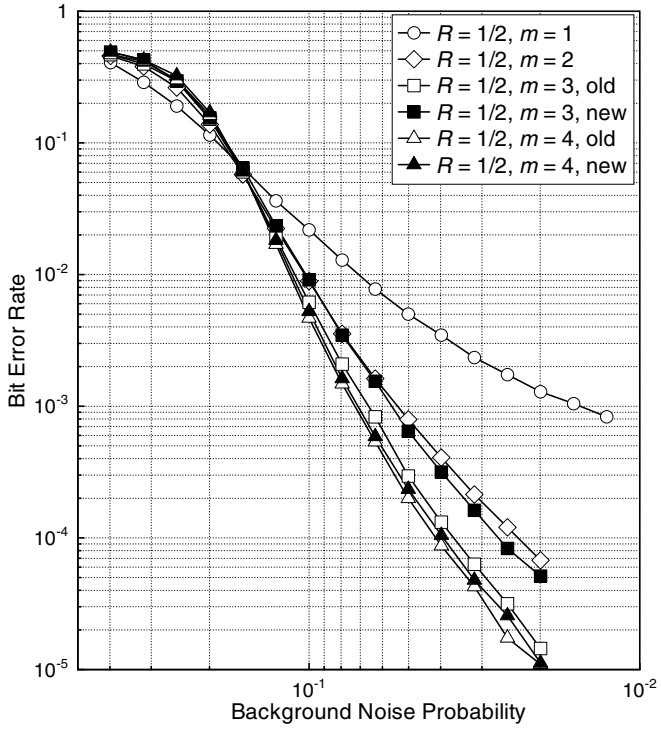

Fig. 2. Simulation results for previous and new $Q(2,3,1)$ trellis codes, using $R = 1/2$ base codes. Impulse noise with $p_i = 10^{-1}$ and one permanent frequency disturbance is present, in addition to the background noise.

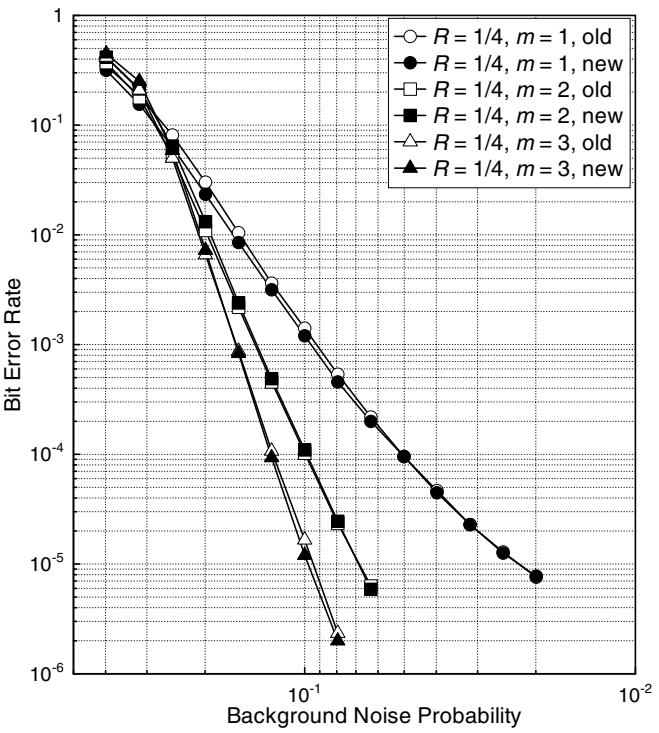

Fig. 4. Simulation results for previous and new $Q_3(4,4,0)$ trellis codes, using $R = 1/4$ base codes. Impulse noise with $p_i = 10^{-1}$ and one permanent frequency disturbance is present, in addition to the background noise.

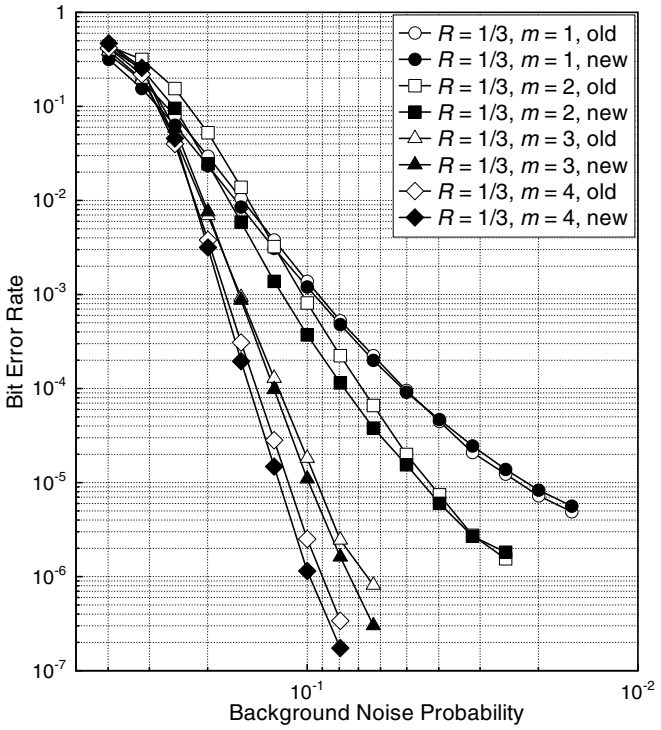

Fig. 3. Simulation results for previous and new $\mathcal{M}(3,4,1)$ trellis codes, using $R = 1/3$ base codes. Impulse noise with $p_i = 10^{-1}$ and one permanent frequency disturbance is present, in addition to the background noise.

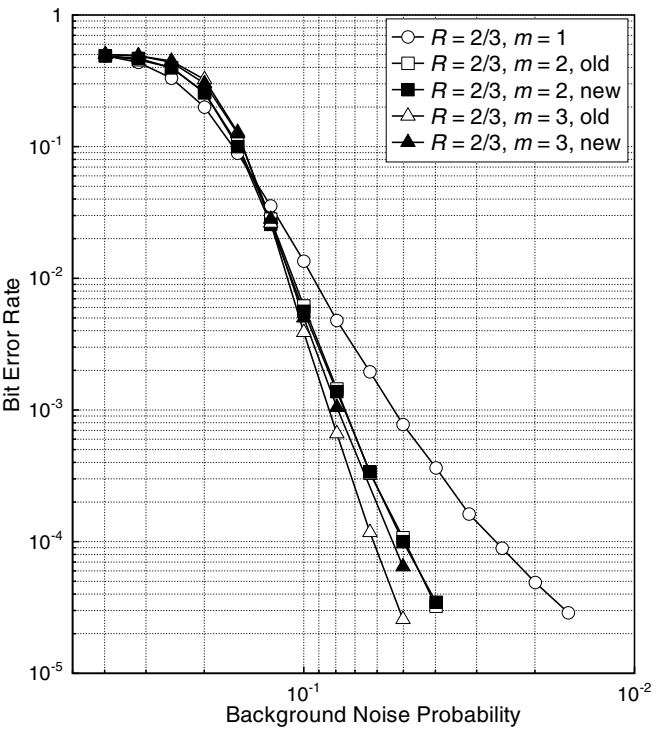

Fig. 5. Simulation results for previous and new $\mathcal{M}(3,4,1)$ trellis codes, using $R = 2/3$ base codes. Impulse noise with $p_i = 10^{-1}$ and one permanent frequency disturbance is present, in addition to the background noise.

TABLE I

NEW PERMUTATION TRELLIS CODES WITH $R = 1/2$ BASE CODE

Code	m	d'_{free}	d_{free}	$N_{d'_{\text{free}}+i}$ $i=0$	$i=1$	$i=2$	$i=3$	$i=4$	Generator sequence
$Q(2,3,1)$	1	5	3	2	0	6	0	14	$(1 \quad 3)^{*+}$
$Q(2,3,1)$	2	8	5	6	0	44	0	280	$(5 \quad 7)^{*+}$
$Q(2,3,1)$	3	10	6	30	14	188	62	1400	$(13 \quad 17)^{*}$
$Q(2,3,1)$	3	10	6	14	472	0	62	2168	$(7 \quad 15)^{+}$
$Q(2,3,1)$	4	11	7	30	126	316	1276	2556	$(23 \quad 33)^{*}$
$Q(2,3,1)$	4	12	6	92	506	254	2042	2554	$(13 \quad 36)^{+}$

* Best binary convolutional code
+ Best $Q(2,3,1)$ trellis code

TABLE II

NEW PERMUTATION TRELLIS CODES WITH $R = 2/3$ BASE CODE

Code	m	d'_{free}	d_{free}	$N_{d'_{\text{free}}+i}$ $i=0$	$i=1$	$i=2$	$i=3$	$i=4$	Generator sequence
$Q(3,4,1)$	1	6	3	12	84	8	400	2244	$\begin{pmatrix}0 & 2 & 3\\ 3 & 1 & 2\end{pmatrix}^{*\diamond}$
$\mathcal{M}(3,4,1)$	1	8	3	104	0	2286	0	46800	$\begin{pmatrix}0 & 2 & 3\\ 3 & 1 & 2\end{pmatrix}^{*+}$
$Q(3,4,1)$	2	9	5	217	477	0	23392	45579	$\begin{pmatrix}6 & 5 & 1\\ 7 & 2 & 5\end{pmatrix}^{*}$
$Q(3,4,1)$	2	9	5	14	504	176	35168	46188	$\begin{pmatrix}1 & 3 & 5\\ 4 & 6 & 7\end{pmatrix}^{\diamond}$
$Q(3,4,1)$	2	9	5	203	602	655	16361	33258	$\begin{pmatrix}0 & 7 & 5\\ 7 & 5 & 2\end{pmatrix}$
$\mathcal{M}(3,4,1)$	2	10	5	124	0	6962	0	102880	$\begin{pmatrix}6 & 5 & 1\\ 7 & 2 & 5\end{pmatrix}^{*}$
$\mathcal{M}(3,4,1)$	2	10	5	316	0	1654	0	159440	$\begin{pmatrix}1 & 3 & 5\\ 4 & 6 & 7\end{pmatrix}$
$\mathcal{M}(3,4,1)$	2	10	5	62	0	13162	0	47078	$\begin{pmatrix}0 & 7 & 5\\ 7 & 5 & 2\end{pmatrix}^{+}$
$Q(3,4,1)$	3	10	7	62	128	36914	24058	4094	$\begin{pmatrix}06 & 13 & 13\\ 13 & 06 & 17\end{pmatrix}^{*}$
$Q(3,4,1)$	3	12	6	31	2327	3420	43076	134380	$\begin{pmatrix}07 & 17 & 11\\ 12 & 10 & 07\end{pmatrix}^{\diamond}$
$Q(3,4,1)$	3	12	5	3003	34103	21432	65964	126580	$\begin{pmatrix}02 & 14 & 17\\ 11 & 03 & 16\end{pmatrix}$
$\mathcal{M}(3,4,1)$	3	12	7	254	0	26868	0	131300	$\begin{pmatrix}06 & 13 & 13\\ 13 & 06 & 17\end{pmatrix}^{*}$
$\mathcal{M}(3,4,1)$	3	14	6	19448	0	114400	0	207830	$\begin{pmatrix}07 & 17 & 11\\ 12 & 10 & 07\end{pmatrix}$
$\mathcal{M}(3,4,1)$	3	14	5	6392	0	172500	0	145380	$\begin{pmatrix}02 & 14 & 17\\ 11 & 03 & 16\end{pmatrix}^{+}$

* Best binary convolutional code
◇ Best $Q(3,4,1)$ trellis code
+ Best $\mathcal{M}(3,4,1)$ trellis code

TABLE III

NEW PERMUTATION TRELLIS CODES WITH $R = 1/3$ BASE CODE

Code	m	d'_{free}	d_{free}	$N_{d'_{\text{free}}+i}$ $i=0$	$i=1$	$i=2$	$i=3$	$i=4$	Generator sequence
$Q(3,4,1)$	1	7	5	2	0	0	6	0	$(1\ 3\ 3)^{*\diamond}$
$Q(3,4,1)$	1	6	3	2	0	0	0	6	$(1\ 1\ 2)$
$\mathcal{M}(3,4,1)$	1	8	5	2	0	6	0	14	$(1\ 3\ 3)^{*}$
$\mathcal{M}(3,4,1)$	1	8	3	2	0	0	0	6	$(1\ 1\ 2)^{+}$
$Q(3,4,1)$	2	11	8	6	0	0	44	0	$(5\ 7\ 7)^{*\diamond}$
$Q(3,4,1)$	2	9	5	6	0	0	0	14	$(2\ 3\ 5)$
$\mathcal{M}(3,4,1)$	2	12	8	20	0	0	0	310	$(5\ 7\ 7)^{*}$
$\mathcal{M}(3,4,1)$	2	12	5	6	0	0	0	14	$(2\ 3\ 5)^{+}$
$Q(3,4,1)$	3	14	10	14	0	0	14	16	$(13\ 15\ 17)^{*}$
$Q(3,4,1)$	3	14	8	7	7	0	15	15	$(11\ 15\ 13)^{\diamond}$
$Q(3,4,1)$	3	13	7	14	0	0	316	188	$(05\ 05\ 13)$
$\mathcal{M}(3,4,1)$	3	16	10	44	0	0	0	124	$(13\ 15\ 17)^{*}$
$\mathcal{M}(3,4,1)$	3	14	8	14	0	0	0	0	$(11\ 15\ 13)$
$\mathcal{M}(3,4,1)$	3	16	7	14	0	0	0	534	$(05\ 05\ 13)^{+}$
$Q(3,4,1)$	4	17	12	30	0	0	141	31	$(25\ 33\ 37)^{*}$
$Q(3,4,1)$	4	17	10	7	16	7	63	94	$(23\ 33\ 25)^{\diamond}$
$Q(3,4,1)$	4	16	10	14	16	44	222	176	$(13\ 15\ 36)$
$\mathcal{M}(3,4,1)$	4	20	12	218	0	0	0	890	$(25\ 33\ 37)^{*}$
$\mathcal{M}(3,4,1)$	4	18	10	30	0	0	0	380	$(23\ 33\ 25)$
$\mathcal{M}(3,4,1)$	4	20	10	30	0	62	0	506	$(13\ 15\ 36)^{+}$

* Best binary convolutional code
◇ Best $Q(3,4,1)$ trellis code
+ Best $\mathcal{M}(3,4,1)$ trellis code

TABLE IV

NEW PERMUTATION TRELLIS CODES WITH $R = 1/4$ BASE CODE

Code	m	d'_{free}	d_{free}	$N_{d'_{\text{free}}+i}$ $i=0$	$i=1$	$i=2$	$i=3$	$i=4$	Generator sequence
$Q(4,4,0)$	1	7	7	2	0	0	6	0	$(3\ 2\ 3\ 3)^{\diamond}$
$Q(4,4,0)$	1	6	6	2	0	4	0	6	$(1\ 1\ 3\ 3)$
$Q'_3(4,4,0)$	1	8	7	2	0	6	0	14	$(3\ 2\ 3\ 3)$
$Q'_3(4,4,0)$	1	8	6	2	0	0	0	6	$(1\ 1\ 3\ 3)^{+}$
$Q(4,4,0)$	2	11	10	6	4	0	37	39	$(5\ 5\ 7\ 7)^{*}$
$Q(4,4,0)$	2	12	10	6	0	14	0	30	$(7\ 5\ 7\ 7)^{\diamond}$
$Q(4,4,0)$	2	10	9	3	0	3	0	8	$(3\ 7\ 5\ 5)$
$Q'_3(4,4,0)$	2	12	10	6	0	0	0	44	$(5\ 5\ 7\ 7)^{*}$
$Q'_3(4,4,0)$	2	12	10	20	0	0	0	310	$(7\ 5\ 7\ 7)$
$Q'_3(4,4,0)$	2	12	9	6	0	0	0	14	$(3\ 7\ 5\ 5)^{+}$
$Q(4,4,0)$	3	15	13	30	32	14	30	95	$(13\ 13\ 15\ 17)^{*}$
$Q(4,4,0)$	3	16	10	28	0	62	0	250	$(17\ 12\ 7\ 17)^{\diamond}$
$Q(4,4,0)$	3	13	12	40	48	117	94	333	$(7\ 7\ 15\ 15)$
$Q'_3(4,4,0)$	3	16	13	14	0	92	0	0	$(13\ 13\ 15\ 17)^{*}$
$Q'_3(4,4,0)$	3	15	10	30	14	0	62	0	$(17\ 12\ 7\ 17)$
$Q'_3(4,4,0)$	3	16	12	14	0	0	0	534	$(7\ 7\ 15\ 15)^{+}$

* Best binary convolutional code
◇ Best $Q(4,4,0)$ trellis code
+ Best $Q'_3(4,4,0)$ trellis code