PAPER *Special Section on Discrete Mathematics and Its Applications*

Codeword Set Selection for the Error-Correcting 4b/10b Line Code with Maximum Clique Enumeration

Masayuki TAKEDA[†a)], ***Nonmember and*** **Nobuyuki YAMASAKI**[†], ***Member***

SUMMARY This paper addresses the problem of finding, evaluating, and selecting the best set of codewords for the 4b/10b line code, a dependable line code with forward error correction (FEC) designed for real-time communication. Based on the results of our scheme [1], we formulate codeword search as an instance of the maximum clique problem, and enumerate all candidate codeword sets via maximum clique enumeration as proposed by Eblen et al. [2]. We then measure each set in terms of resistance to bit errors caused by noise and present a canonical set of codewords for the 4b/10b line code. Additionally, we show that maximum clique enumeration is #P-hard.

key words: *4b/10b, NP-hard, maximum clique enumeration*

1. Introduction

Most control systems today are designed as distributed real-time systems. Instances of these systems can be found everywhere: in industrial plants, robots, and spacecraft.

In a real-time system, correctness of tasks are defined not only by the correctness of a given task's output but also by the completion of the task before a given deadline. Meeting deadlines is crucial for these systems as failure to do so may result in catastrophic consequences including loss of life. Thus, tasks and the underlying software are subject to strict requirements and schedulability analysis [3] in order to guarantee deadlines.

Modern real-time systems are usually comprised of many real-time systems interoperating in a distributed fashion. Real-time communication, with guarantees on bounds of end-to-end latency is required for building these systems [4]. Retransmission cannot be tolerated in real-time communication as it introduces unbounded latency in transmission.

Choices and tradeoffs made at various abstraction layers found in modern communication protocol stacks are often ill-suited for real-time communication, a notable example being line codes. Indeed, the recent trend among line codes is to improve throughput by adopting longer and longer codeword lengths in order to reduce overhead introduced by encoding. This is exemplified by modern line codes such as 64b/66b and 128b/132b in contrast to older line codes such as 8b/10b and NRZI. These line codes are used as the underlying encodings for various serial communication protocols such as 100 Gigabit Ethernet [5] and USB 3.1 [6]. However, most line codes are problematic to build upon when devising protocols for communication in real-time systems. In general, line codes such as 8b/10b are designed with three things in mind: achieve DC-balance, where the resulting number of zeros and ones transmitted are the same over time, clock embedding, i.e. the clock may be extracted from the stream of codewords on the receiving end, and error detection. Additional codewords for transmission control may also be desirable.

Unfortunately, line codes with only these three characteristics implicitly assume either a communication channel with little to no noise or lax latency requirements. Real-time systems often operate in environments where the former does not hold, and it is safe to assume that the latter never is the case. One result of this assumption is that conventional line codes do not incorporate any form of FEC, so must rely on either error correction implemented at higher abstraction layers or retransmission. The problem with adding on error correction is that a single bit error occurring at the line code level is translated to a multiple bit error when decoded, thus requiring block-level error correction codes such as Reed-Solomon codes [7]. This necessitates waiting for a whole block of bytes to arrive before performing error correction, introducing latency unacceptable for real-time systems. Retransmission, as noted above, is also not an option as it introduces unbounded latency and, in some cases, makes transmission all but impossible due to exponential decreases in throughput as the bit error rate (BER) increases [8].

The 4b/10b line code, proposed by Suito et al. [9] and standardized in IPSJ-TS 0015:2015 [10] and IEC TR 63094:2017 [11], addresses these issues by adding bit-level error correction at the line code level for single bit error correction and double bit error detection, while preserving DC-balance and facilitating clock recovery. This is achieved by setting several constraints on codewords and searching over the ten bit codeword space in an ad-hoc manner. The 18 codewords found are mapped to four bits and two control characters corresponding to "setup" and "idle".

We have found a set of 20 codewords that satisfy the same constraints on codewords imposed by the protocol [1], and used the two additional codewords to add two more control characters used to implement functional compatibility with 8b/10b. Due to the formulation of code search as a Maximum Clique Problem, we know that the set of codes found is the largest possible, but the question remains whether the particular ones found is the best among all code sets of size 20.

Manuscript received October 22, 2019.
Manuscript revised February 26, 2020.

[†]The authors are with the Department of Information and Computer Science, Faculty of Sciece and Technology, Keio University, Yokohama-shi, 223-8521 Japan.
a) E-mail: mtakeda@ny.ics.keio.ac.jp

DOI: 10.1587/transfun.2019DMP0019

We answer this by improving upon this work by finding all such sets of 20 codewords and presenting the set of codewords which have the best qualitative characteristics for use as a line code. Finding this set corresponds to solving the Maximum Clique Problem and enumerating all such sets correspond to solving the Maximum Clique Enumeration Problem. Additionally, we show that the problem that we are trying to solve is #P-hard.

2. Preliminaries

When considering MCP and related problems, we are concerned with a simple graph $G = (V, E)$ consisting of a set of vertices V and edges $E = \{\{x, y\}|(x, y) \in V \times V\}$. Two vertices x and y are said to be adjacent if $\{x, y\} \in E$. We define $\Gamma(x)$ to be the set of vertices which are adjacent to x. A clique in G is a complete subgraph, i.e. a subgraph of V in which all vertices are adjacent to each other. If no vertices can be added to a given clique to make a larger clique, the clique is said to be maximal, and the largest maximal cliques in a graph are referred to as maximum cliques. The problem of finding a maximum clique in a given graph is referred to as the Maximum Clique Problem (MCP) and the problem of enumerating all maximum cliques in a given graph is referred to as the Maximum Clique Enumeration Problem (MCE).

3. Related Work

Raahemi proposed an error correction scheme for use with the 64b/66b line code [12] which takes error multiplication properties of 64b/66b into account. Carney and Chandler proposed a line code integrated with a BCH code [13]. A line code integrating Reed-Solomon codes, designed for usage in high-energy physics, has been proposed by Papotti [14]. However, these line codes all result in long decode latencies due to large frames or error correction block sizes and are undesirable for use in real-time communication with strict latency requirements.

MCP is known to be an NP-hard problem and there is extensive literature on approaches to solve this problem [15]. While binary code search formulated as MCP has been studied in the relatively limited scope of Hamming and Johnson graphs [16], [17], we are not aware of prior work applying exact algorithms to the problem motivated by finding codes for designing line codes or related encoding schemes.

We make heavy use of the work by Eblen et al. [2] of applying MCP and MCE algorithms to problems in bioinformatics, the only work we are aware of concerning the enumeration of all maximum cliques.

4. The 4b/10b Line Code

We briefly review the 4b/10b line code proposed by Suito et al. [9]. 4b/10b is a line code which encodes four bits of data as ten bit codewords and has the following characteristics:

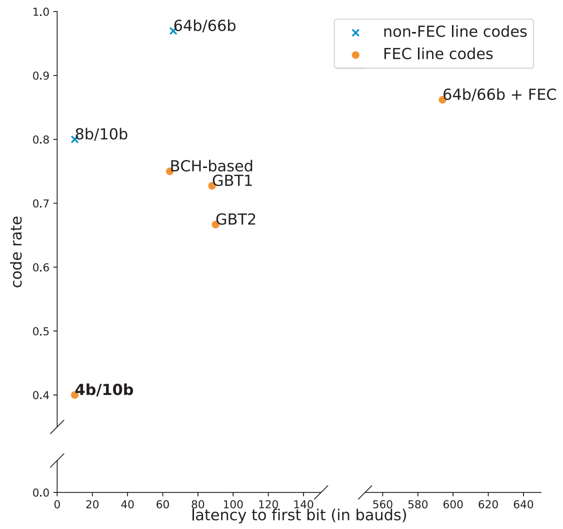

Fig. 1 A comparison of line codes in terms of latency until the first bit is decoded and code rate. In this plot, error-correcting capability is not considered. 4b/10b, the line code which is the focus of this work, is highlighted in bold.

Table 1 Machine specifications.

Processor	Intel Core i7-7600U @ 2.80 GHz
Memory	8x2 GB 2133 MHz DDR4 SODIMM
Operating System	Linux 4.19.80, NixOS 19.03 (Koi)
Nixpkgs Channel	19.03.173677.daf861a810d (Koi)
Rust Compiler	rustc 1.42.0-nightly (3e0a1c091 2019-12-26)

- DC-balance,
- clock recovery,
- single bit error correction and double bit error detection,
- a few control characters for transmission control (two in the case of the original standard, four in the case of our improved version [1]).

This is achieved by selecting codewords which obey the following constraints; for all ten bit codewords x,

1. The Hamming weight of x must be five.
2. The run length of x must be less than or equal to five.
3. The run length of x, with a single bit flip in an arbitrary location, must be less than or equal to five.

and for all two codewords x and y where $x \neq y$,

1. The Hamming distance between x and y must be greater or equal to four.
2. The run length of xy must be less than or equal to five.
3. The run length of xy, with two bit flips at least four bits apart, must be less than or equal to five.

where xy is the concatenation of x and y, and the run length of x is defined as the length of the longest string of consecutive 0s or 1s in x. For example, the run length of the codeword 1010100011 is three. The error pattern 0000100001 will cause two bit flips four bits apart, changing, for example, the codeword 1010100011 to 1010000010. This error pattern fulfills the criteria for Constraint 3. An example

Table 2 Codewords appearing in maximum cliques found (continued in Table 3).

vertex number	codeword
0	0010110101
1	0010110110
2	0010111001
3	0010111010
4	0011001101
5	0011001110
6	0011010011
7	0011100101
8	0011100110
9	0011101001
10	0011101010
11	0100101101
12	0100101110
13	0100110011
14	0100111001
15	0100111010
16	0101001011
17	0101010101
18	0101010110
19	0101011001
20	0101011010
21	0101011100
22	0101100011
23	0101100101
24	0101100110
25	0101101001
26	0101101010
27	0101110001
28	0101110010
29	0101110100
30	0110001011
31	0110001101
32	0110001110
33	0110010101
34	0110010110
35	0110011001
36	0110011010
37	0110011100
38	0110100011
39	0110100101
40	0110100110
41	0110101001
42	0110101010
43	0110110100
44	0111000101
45	0111000110
46	0111001100
47	0111010001
48	0111010010
49	1000101101

Table 3 Codewords appearing in maximum cliques found (continued from Table 2).

vertex number	codeword
50	1000101110
51	1000110011
52	1000111001
53	1000111010
54	1001001011
55	1001010101
56	1001010110
57	1001011001
58	1001011010
59	1001011100
60	1001100011
61	1001100101
62	1001100110
63	1001101001
64	1001101010
65	1001110001
66	1001110010
67	1001110100
68	1010001011
69	1010001101
70	1010001110
71	1010010101
72	1010010110
73	1010011001
74	1010011010
75	1010011100
76	1010100011
77	1010100101
78	1010100110
79	1010101001
80	1010101010
81	1010110100
82	1011000101
83	1011000110
84	1011001100
85	1011010001
86	1011010010
87	1100010101
88	1100010110
89	1100011001
90	1100011010
91	1100101100
92	1100110001
93	1100110010
94	1101000101
95	1101000110
96	1101001001
97	1101001010

of an error pattern that does not is 0000010001, as the bit flips it will cause are only three bits apart. Constraint 1 enables single bit error correction and double bit error detection. Constraints 2 and 3 enable clock recovery by ensuring a timely bit transition.

We show a comparison between 4b/10b and other line codes in Fig. 1. Note that while 4b/10b has a significantly lower code rate when compared to other error-correcting line codes, it is the only line code with a decode latency on par with 8b/10b.

5. Codeword Search as a Maximum Clique Problem

We are interested in finding the largest set of bit strings, or codewords, that obey a set of constraints which give rise to desirable characteristics of the 4b/10b line code such as error correction, DC-balance, and clock embedding. Following our scheme [1], we consider the generalized version of this problem by considering arbitrary constraints on individual codewords and arbitrary constraints on pairs of codewords which must hold for all pairs in a candidate set of codewords. This problem can then be formally described in

Table 4 All maximum cliques found. Refer to Tables 2 and 3 for actual codewords. Vertices 6 and 91, highlighted in bold, form the smallest maximum clique cover. Set 8 is the codeword set specified in [1] and set 7 is the set that we base the proposed canonical 4b/10b translation table upon, both highlighted in bold.

set number	vertices
0	0 5 **6** 9 11 13 17 20 24 34 35 42 50 52 54 56 61 69 74 76
1	0 5 **6** 9 11 15 16 17 24 34 35 38 51 56 57 61 64 69 74 78
2	0 5 **6** 9 12 14 16 18 23 31 36 38 49 51 55 58 62 72 73 80
3	0 5 **6** 9 13 18 19 23 26 31 36 40 49 53 54 55 62 72 73 76
4	1 4 **6** 10 11 15 16 17 24 32 35 38 50 51 56 57 61 71 74 79
5	1 4 **6** 10 12 13 18 19 23 33 36 41 49 53 54 55 62 70 73 76
6	1 4 **6** 10 12 14 16 18 23 33 36 38 51 55 58 62 63 70 73 77
7	1 4 **6** 10 13 17 20 24 25 32 35 39 50 52 54 56 61 71 74 76
8	2 5 **6** 7 11 15 18 19 22 30 33 40 51 55 58 62 63 69 72 80
9	2 5 **6** 7 13 17 20 24 25 31 34 42 49 53 56 57 60 68 71 78
10	3 4 **6** 8 12 14 17 20 22 30 34 39 51 56 57 61 64 70 71 79
11	3 4 **6** 8 13 18 19 23 26 32 33 41 50 52 55 58 60 68 72 77
12	17 24 25 35 39 42 46 48 59 61 66 74 79 81 83 85 88 **91** 92 97
13	17 25 28 34 35 39 42 46 57 64 67 75 78 79 82 86 90 **91** 92 95
14	18 23 26 36 40 41 46 47 59 62 65 73 80 81 82 86 87 **91** 93 96
15	18 26 27 33 36 40 41 46 58 63 67 75 77 80 83 85 89 **91** 93 94
16	19 23 28 33 36 40 41 46 59 62 63 73 80 81 82 86 88 **91** 92 97
17	19 26 29 37 40 41 44 48 55 63 66 72 73 77 80 84 90 **91** 92 95
18	20 24 27 34 35 39 42 46 59 61 64 74 79 81 83 85 87 **91** 93 96
19	20 25 29 37 39 42 45 47 56 64 65 71 74 78 79 84 89 **91** 93 94
20	21 23 26 36 41 43 45 47 58 62 65 72 73 77 80 84 87 **91** 93 96
21	21 23 28 36 41 43 45 47 55 62 63 73 77 80 84 86 88 **91** 92 97
22	21 24 25 35 42 43 44 48 57 61 66 71 74 78 79 84 88 **91** 92 97
23	21 24 27 35 42 43 44 48 56 61 64 74 78 79 84 85 87 **91** 93 96

the following way: given a unary relation $A \subseteq V$ and a binary symmetric relation $B \subseteq V \times V$ where the set $V = \{0, 1\}^n$ is the set of all bit strings of length n,

$$\begin{aligned} &\underset{S}{\text{maximize}} \quad |S| \\ &\text{subject to} \quad S \subseteq A, S \times S \subseteq B. \end{aligned} \tag{1}$$

One solution to this problem can be found by solving MCP on a corresponding graph. By setting $V = A$ and $E = \{\{x, y\} | (x, y) \in B\}$, a maximum clique S in G will yield a

Algorithm 1 4b/10b Codeword Set Enumeration

```
function ENUMERATE4B10B
    $V, E \leftarrow$ formulate graph from 4b/10b constraints
    $Q_{max} \leftarrow \text{MCS}(V)$
    $V' \leftarrow \text{GetDegeneracyOrdering}(V)$
    $C \leftarrow V'$
    $N \leftarrow \emptyset$
    for $c \in C$ do
        EXTEND($Q_{max}$, $\{c\}$, $C \cap \Gamma(c)$, $N$)
        $C \leftarrow C \setminus \{c\}$
        $N \leftarrow N \cup \{c\}$
    end for
end function
function EXTEND($Q_{max}$, $Q$, $C$, $N$)
    if $|Q| + |C| < |Q_{max}|$ then return
    if $C = \emptyset$ then
        if $N = \emptyset$ then output $Q$
        return
    end if
    $p \leftarrow \arg\min_{p \in C \cup N} |C \setminus \Gamma(p)|$
    for $c \in C \setminus \Gamma(p)$ do
        EXTEND($Q \cup \{c\}$, $C \cap \Gamma(c)$, $N \cap \Gamma(c)$)
        $C \leftarrow C \setminus \{c\}$
        $N \leftarrow N \cup \{c\}$
    end for
end function
```

solution to the codeword search problem above, as by definition, no clique S' such that $|S'| > |S|$ can exist.

One question that may arise is whether there are more efficient algorithms for solving this problem. Unfortunately, there is strong evidence that this is most likely not the case. We have shown that this problem is NP-hard [1], meaning that if this problem could be solved in its general form in polynomial time, then P=NP. Indeed, the algorithm used to find the original set of codewords specified in the standard is a greedy heuristic which, while being polynomial in complexity, has no guarantees of finding the maximum set and found a set of only 18 codewords [18].

6. Codeword Enumeration as a Maximum Clique Enumeration Problem

Given a set of codewords found via some MCP algorithm, a question that arises is whether such set is the only set of its size and if not, whether there are others which are more suitable for use as a line code. Multiple maximum cliques may exist; the problem of enumerating all of them is referred to as the Maximum Clique Enumeration Problem (MCE). In order to solve MCE, we use the Intelligent Backtracking algorithm, proposed by Eblen et al. [2] with the MCS algorithm, proposed by Tomita et al. [19], instead of using MaximumCliqueFinder as the maximum clique algorithm. Additionally, we apply a degeneracy ordering of the vertices as a preprocessing step as proposed by Eppstein and Strash [20]. We show the full algorithm in Algorithm 1. The algorithm was implemented in the Rust programming language, executed on a machine specified in Table 1, and took 7.11 seconds.

We present the sets of codewords found in Tables 2,

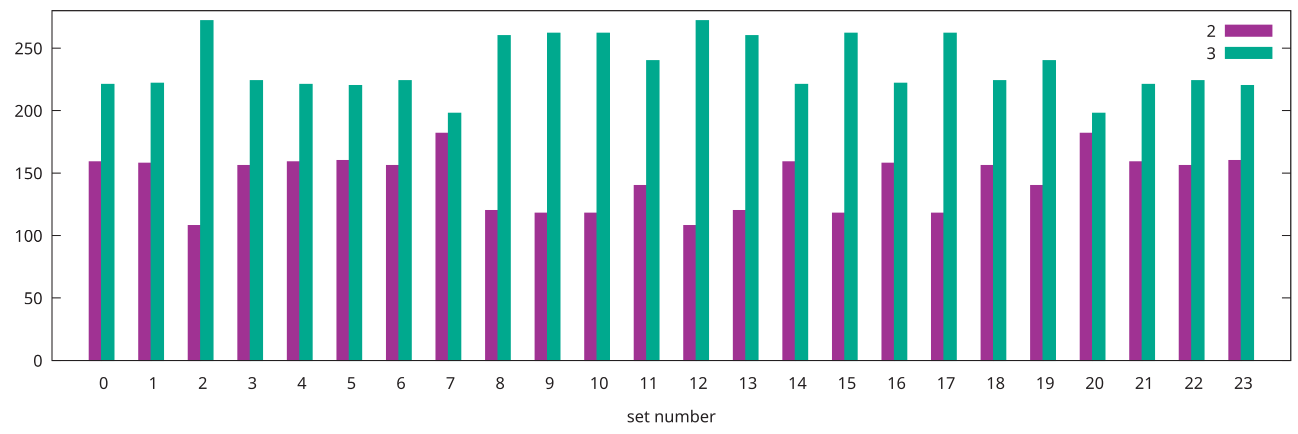

Fig. 2 The distribution of the run length of concatenated codewords. The 4b/10b line code specifies that this value must be equal or less than five for all pairs of codewords in order to enable clock recovery.

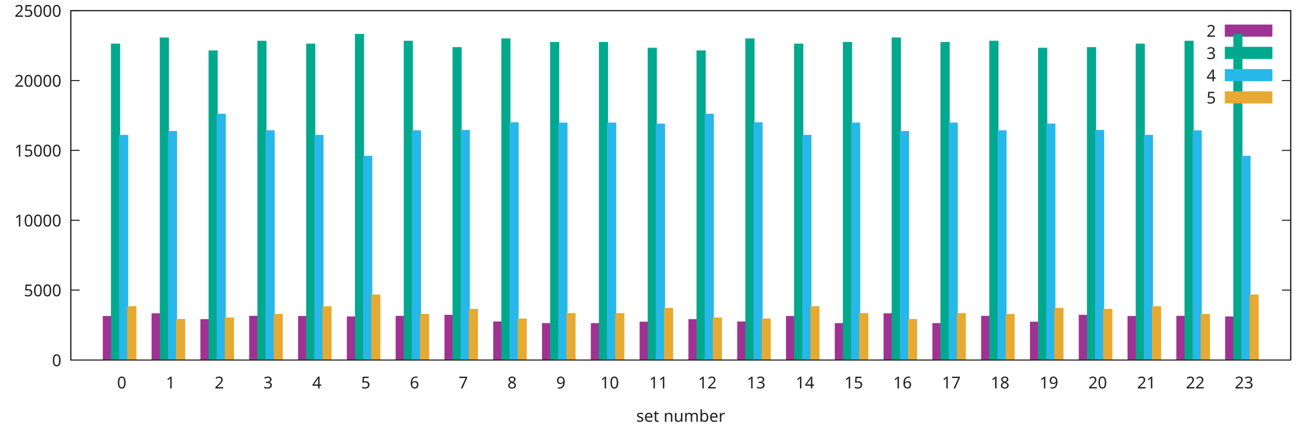

Fig. 3 The distribution of the run length of concatenated codewords, when two bits at least four bits apart, have been flipped. The 4b/10b line code specifies that this value must be equal or less than five for all pairs of codewords in order to preserve clock recovery in the presence of noise. Ideally, this should hold for all double bit flips, but strengthening this constraint further will reduce the size of candidate codewords to the point where it becomes impossible to map four bits to them.

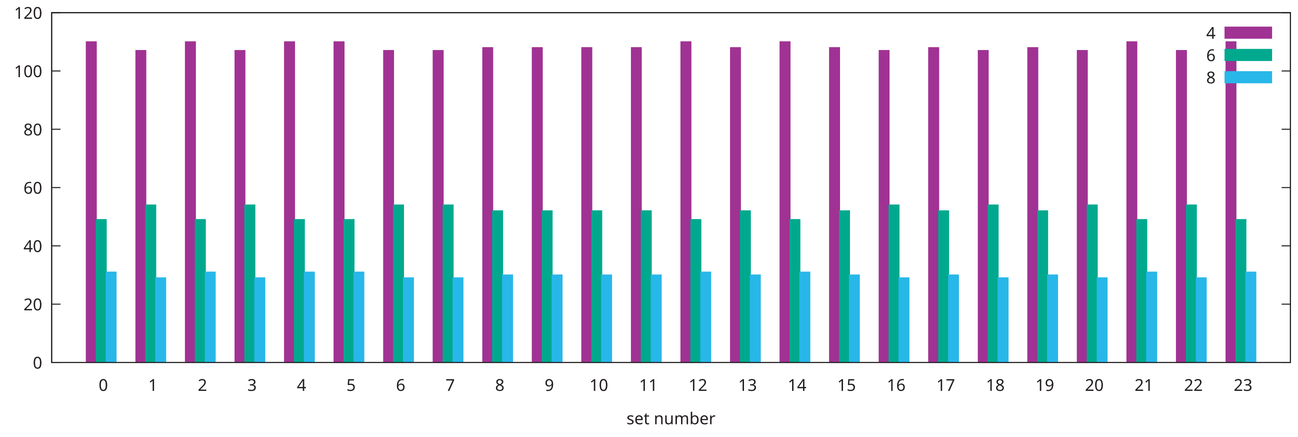

Fig. 4 The distribution of the hamming distance between codewords. The 4b/10b line code specifies this value must be greater or equal than four in order to implement error correction. As codewords have the same number of zeros and ones, hamming distances only take even values.

3, and 4. As there was significant overlap among the sets, we number and list the union of all codewords appearing in maximum cliques found in Tables 2 and 3, and refer to codewords by number when listing them in Table 4.

Table 5 The canonical 4b/10b translation table based on codeword set 7.

input	output
0000	0010110110
0001	0011001101
0010	0011010011
0011	0011101010
0100	0100110011
0101	0101010101
0110	0101011010
0111	0101100110
1000	0101101001
1001	0110001110
1010	0110011001
1011	0110100101
1100	1000101110
1101	1000111001
1110	1001001011
1111	1001010110
setup	1001100101
idle	1010010101
ack	1010011010
command	1010100011

Eblen et al. [2] empirically observes that there exists a nonempty set of "essential" vertices shared by all maximum cliques for most graphs and proposes a pruning method based on this fact for use prior to running MCE algorithms. Notably, this is not the case for our graph, despite significant overlap among maximum cliques found, with vertices 6 and 91 both being shared among 12 cliques and coming the closest to being "essential". Furthermore, these two vertices form the smallest instance of a maximum clique cover $V' \subseteq V$ where all maximum cliques in G contain a vertex in V'. This can be explained by the constraints being agnostic to order and inversion; that is, given a set of codewords conformant to these constraints, the set of codewords obtained by reversing or inverting all codewords will also result in a conformant codeword set. Thus, if a codeword set is not equal to itself under either reversion or inversion, the essential set will be empty. This can be confirmed by observing that the codeword corresponding to vertex 6 is both the inverted and reversed version of the codeword corresponding to vertex 91.

7. Evaluation and Selection

We evaluate the sets found based on the 4b/10b line code's constraints with respect to each other mentioned in Sect. 4 The distribution of these values for all candidate sets are shown in Figs. 2, 3, and 4. As Fig. 2 shows, set 7 and 20 have the shortest run lengths on average, with 182 instances out of a total 380 having a run length of 2. These two sets are similarly favorable across all remaining measures, with negligible differences between cliques as seen in Figs. 3 and 4. Thus, we present set 7, somewhat arbitrarily, as the canonical 4b/10b line code in Table 5. Notably, the codeword set found in [1] corresponds to set 8, which has a worse run length distribution when compared to set 7 as shown in Fig. 2.

8. #P-hardness of Maximum Clique Enumeration

As MCP is NP-hard, MCE is clearly also NP-hard, but it remains to be seen whether it can be classified as an instance of a more specific complexity class. Here, we show the following:

Theorem 1. *Maximum Clique Enumeration is #P-hard.*

Proof. We show this by reduction from #3SAT. Given an instance of 3SAT, under Karp's formulation as a graph [21] for the proof of the NP-completeness of the clique decision problem, enumerating all maximum cliques will enumerate all assignments that satisfy the formula. Thus, counting the number of enumerated maximum cliques will yield the answer to #3SAT. As this reduction can be performed in polynomial time, MCE is #P-hard. □

9. Conclusion

We have presented all maximum sets of codewords possible for the 4b/10b line code found via formulation of codeword search and enumeration as instances of maximum clique problems, and chosen a canonical set based on qualitative evaluations. The 4b/10b line code has substantial application in real-time communication due to its ease of implementation due to its simplicity and codeword-level compatibility with 8b/10b enabling switching between line codes depending on the use case. Additionally, we have shown that maximum clique enumeration belongs to the #P-hard complexity class. We believe that this approach of finding and enumerating codewords under various constraints has wide applications and deserves further study.

References

[1] M. Takeda and N. Yamasaki, "Finding the maximum number of symbols for the 4b/10b line code with error correction," 2019 Seventh International Symposium on Computing and Networking Workshops, 2019.

[2] J.D. Eblen, C.A. Phillips, G.L. Rogers, and M.A. Langston, "The maximum clique enumeration problem: Algorithms, applications, and implementations," BMC Bioinform., vol.13, p.S5, BioMed Central, 2012.

[3] G.C. Buttazzo, Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications, Springer Science & Business Media, 2011.

[4] K. Tindell, A. Burns, and A.J. Wellings, "Analysis of hard real-time communications," Real-Time Syst., vol.9, no.2, pp.147–171, 1995.

[5] "802.3-2018 - IEEE Standard for Ethernet," 2018.

[6] "Universal Serial Bus 3.1 Specification," 2013.

[7] I.S. Reed and G. Solomon, "Polynomial codes over certain finite fields," J. Soc. Ind. Appl. Math., vol.8, no.2, pp.300–304, 1960.

[8] K. Suito, R. Ueda, K. Fujii, T. Kogo, H. Matsutani, and N. Yamasaki, "The dependable responsive multithreaded processor for distributed real-time systems," IEEE Micro, vol.32, no.6, pp.52–61, 2012.

[9] K. Suito, T. Kogo, H. Matsutani, and N. Yamasaki, "Design and implementation of dependable communication mechanism on responsive link for distributed real-time systems," IPSJ Journal of Information Processing, vol.53, no.12, pp.2728–2739, 2012 (in Japanese).

[10] "Dependable Line Code with Error Correction Capability: 4b/10b," IPSJ-TS 0006:2003.
[11] "Multimedia systems and equipment - Multimedia signal transmission - Dependable line code with error correction," IEC TR 63094:2017.
[12] B. Raahemi, "Error correction on 64/66 bit encoded Links," Canadian Conference on Electrical and Computer Engineering, 2005, pp.412–416, IEEE, 2005.
[13] D.T. Carney and E.W. Chandler, "Error correction coding for a serial digital multi-gigabit communication system," 2004 IEEE Electro/Information Technology Conference, pp.33–41, Aug. 2004.
[14] G. Papotti, "Architectural studies of a radiation-hard transceiver asic in 0.13 um cmos for digital optical links in high energy physics applications," Technical Report, CERN, 2007.
[15] Q. Wu and J.K. Hao, "A review on algorithms for maximum clique problems," Eur. J. Oper. Res., vol.242, no.3, pp.693–709, 2015.
[16] T. Etzion and P.R. Ostergard, "Greedy and heuristic algorithms for codes and colorings," IEEE Trans. Inf. Theory, vol.44, no.1, pp.382–388, 1998.
[17] I.M. Bomze, M. Budinich, P.M. Pardalos, and M. Pelillo, "The maximum clique problem," Handbook of Combinatorial Optimization, pp.1–74, Springer, 1999.
[18] N. Yamasaki, "Conversion method of transmission line code for digital communication, and code generation method for the conversion method," Japanese Patent #P5900850, 2016.
[19] E. Tomita, Y. Sutani, T. Higashi, S. Takahashi, and M. Wakatsuki, "A simple and faster branch-and-bound algorithm for finding a maximum clique," International Workshop on Algorithms and Computation, pp.191–203, Springer, 2010.
[20] D. Eppstein and D. Strash, "Listing all maximal cliques in large sparse real-world graphs," International Symposium on Experimental Algorithms, pp.364–375, Springer, 2011.
[21] R.M. Karp, "Reducibility among combinatorial problems," Complexity of Computer Computations, pp.85–103, Springer, 1972.

Masayuki Takeda is an undergraduate student at the Department of Information and Computer Science at Keio University, expected to receive his B.S. degree in 2020. His research interest include real-time systems and clique problems.

Nobuyuki Yamasaki is a Professor in the Department of Information and Computer Science at Keio University. He received his B.S., M.S., and Ph.D. degrees from Keio University in 1991, 1993, and 1996, respectively. His research interests include processor architecture, parallel and distributed systems, real-time systems, real-time communication, System-on-Chip (SoC), and robotics.