INFORMATION AND CONTROL 14, 559–565 (1969)

How Good is Morse Code?

E. N. GILBERT

Bell Telephone Laboratories, Incorporated, Murray Hill, New Jersey 07974

Not bad!

1. INTRODUCTION

A superficial comparison (comparison I below) of international Morse code with codes of information theory suggests that Morse code is slower than necessary. Codes for use by human operators must be designed with other factors beside speed in mind. Comparisons II–V in Secs. 3 and 4 take some of these into account and show finally that Morse code cannot be much improved.

Comparison V considers a tradeoff between code speed and the mean number of hand motions per transmitted letter. This comparison is made for three different keys. Section 5 examines the possibility of reducing hand motions by designing a new key. An "optimal" key is found to be only a slight improvement over present electronic keys.

2. TIMING

In international Morse code the code elements are timed as follows. Take the duration of key closure for a dot as the basic time unit. Then dashes require key closure for 3 time units. Key open times are 1 unit between dots and dashes within a letter, 3 units between letters, and 7 units between words.[1] Thus a letter with d dots and D dashes requires $2 + 2d + 4D$ time units (including letter space). The blank between words, considered as a 27th letter, requires 4 time units.

If the ith letter requires time t_i and occurs with probability p_i then

[1] Different authors disagree on the length of the word space. Kaufman (1957) and Marshall and Wheeler (1966) give figures of 5 and 6 units. The 7 unit space appears in the Telegraph Regulations (Paris revision 1949) annexed to the International Telecommunications Convention (Atlantic City 1947); see Henney (1959) and FCC (1955).

the mean time per letter is

$$T = \sum_i p_i t_i . \tag{1}$$

G. Dewey (1923) gives English letter probabilities p_i. The mean time per letter in Morse code is found to be $T = 8.110$ time units. Here the word separation is treated as a 27th letter while numerals and punctuation marks are ignored.

In the sections which follow, Morse code will be compared with other codes designed to transmit English text letter by letter. These comparisons are somewhat unfair to Morse code because it must transmit languages other than English and must provide numerals and punctuation marks.

3. BINARY CODES

In wire telegraphy, messages are received as sequences of clicks which occur at each key opening or closure. This suggests some comparisons with optimal systems in which binary choices (0 = no click, 1 = click) are made at the rate of one per unit time.

Comparison I. Huffman's coding algorithm gives the fastest letter-by-letter decipherable encoding of English letters into binary digits. The mean time per letter is $T = 4.120$ units (see Gilbert and Moore (1959)), about half the Morse time.

Comparison II. Sending letters by hand in the Huffman code might be very confusing because a letter will sometimes begin with a key opening and sometimes with a key closure. To avoid this difficulty one must require an even number of clicks for each letter. The best code of this sort is not known but a good one can be obtained by adding one more click at the end of each letter which had an odd number of clicks. This increases the mean time per letter to 4 53 units.

Comparison III. When Morse code is used to key a tone generator the letter space acts as a synchronizing mark. Silence of 3 units or more always indicates that a new letter will begin. A binary code with a similar synchronizing feature may be designed as follows. Word space will be 000. All other letters will begin 001, end in 1, have an even number of 1's and have no two consecutive 0's (aside from the first two digits). If $F(k)$ is the number of such binary k-tuples, $F(4)$, $F(5), \cdots, F(10)$ are 1, 1, 1, 3, 4, 6, 11. Also $F(k) = F(k-2) + 2F(k-3) + F(k-4)$. By assigning the shortest k-tuples to the most common letters, I obtain a code with $T = 5.735$ units per letter. An

analogous code in which word space is 0000 and other letters begin 0001 required 7.112 time units per letter.

4. DOT-DASH CODES

When an operator uses the first code in comparison III with a hand key his dots and dashes will be key closures lasting 1 or 2 time units. The spaces between these elements may be either one or two time units long. Morse code, having a longer dash and only a single kind of space within any letter requires less precise timing for readability. In fact, earlier codes (for example, the American Morse code which is still used in wire signaling) did use some dashes and spaces of other lengths, but these peculiar elements were deliberately avoided by the designers of the international Morse code. The comparisons which follow all use codes in which only the dot, dash, and space times of international Morse code are allowed.

Comparison IV. To minimize T one should construct the 26 shortest code groups and assign them to letters in such a way that the coded letter length becomes a decreasing function of the letter probability. This process constructs a code with mean time 7.750 units per letter, a saving of only 4.5%. This code is faster than Morse code because Morse code gives some common letters excessively long code groups. The worst example is the letter O (- - -) which requires 14 time units.

Comparison V. Some Morse code groups are easier to send than others. The difficulty in sending a letter might be measured by the number of hand motions involved. Thus, if an ordinary straight key is used, the letter O requires 3 motions while the shorter letter B (- · · ·) requires 4. If the kth letter requires m_k motions the mean number of motions per letter is

$$M = \sum p_k m_k . \tag{2}$$

The code which minimizes T does not also minimize M. One can construct a family of codes in which each code minimizes T for a given value of M.

Actually three different kinds of keys are in common use. Each key will have its own family of minimizing codes. The simplest key, the *straight key* requires one motion for each dot or dash. The *bug key* contains a mechanical vibrator which can make any sequence of consecutive dots, all with one motion. Finally, various *electronic key* circuits are used to produce runs of dashes as well as runs of dots with one motion.

Then m_k is the number of dots and dashes with a straight key, the number of dashes and runs of dots with a bug, and the number of runs of both kinds with an electronic key. For example the question mark ($\cdot\cdot--\cdot\cdot$) takes 6, 4, or 3 motions with straight, bug, or electronic keys. The mean number of motions per letter (counting word space as no motions) for Morse code is

$$\begin{aligned} M &= 2.055 \quad \text{(straight key)} \\ M &= 1.604 \quad \text{(bug)} \\ M &= 1.393 \quad \text{(electronic key)}. \end{aligned} \tag{3}$$

The numbers in (3) will be useful for comparing different codes using the same key. Such numbers do not give a fair comparison of different keys with the same code because the motions involved are so different. For example, all straight key motions include both a downward hand movement and a return movement; some bug and electronic key motions require only a single sidewise hand movement. I have not counted return movements as extra motions because the returns are aided by spring forces. Electronic keys have an automatic "self-completing" feature which ensures that each dot or dash, once started, has accurate duration. Thus, the timing of electronic key motions is less critical than those of a straight key or bug. These factors make the electronic key even more superior to the other keys than one might conclude from (3).

To find a code which minimizes T for a given value of M in (2), one may introduce a positive Lagrange multiplier λ and minimize

$$U = \sum p_k(t_k + \lambda m_k).$$

If λ is so chosen that the code which minimizes U also satisfies (2), then this code minimizes T subject to (2). To minimize U one assigns code groups to letters in such a way that $t_k + \lambda m_k$ becomes a decreasing function of p_k.

Figure 1 shows the minimum T as a function of the allowed value of M. The three curves correspond to the three keys types; these may be compared with the three isolated points which represent the Morse code. The curves cover the entire range of interest, $\lambda = 0$ (unconditional minimization of T) to $\lambda = \infty$ (minimization of M). The curve for the straight key is very short because m_k and t_k are closely related for this key. By contrast, it is possible to produce arbitrarily long code groups with a single motion of a bug or electronic key; then codes with small M are obtainable at the expense of a very large T. Figure 1 shows that

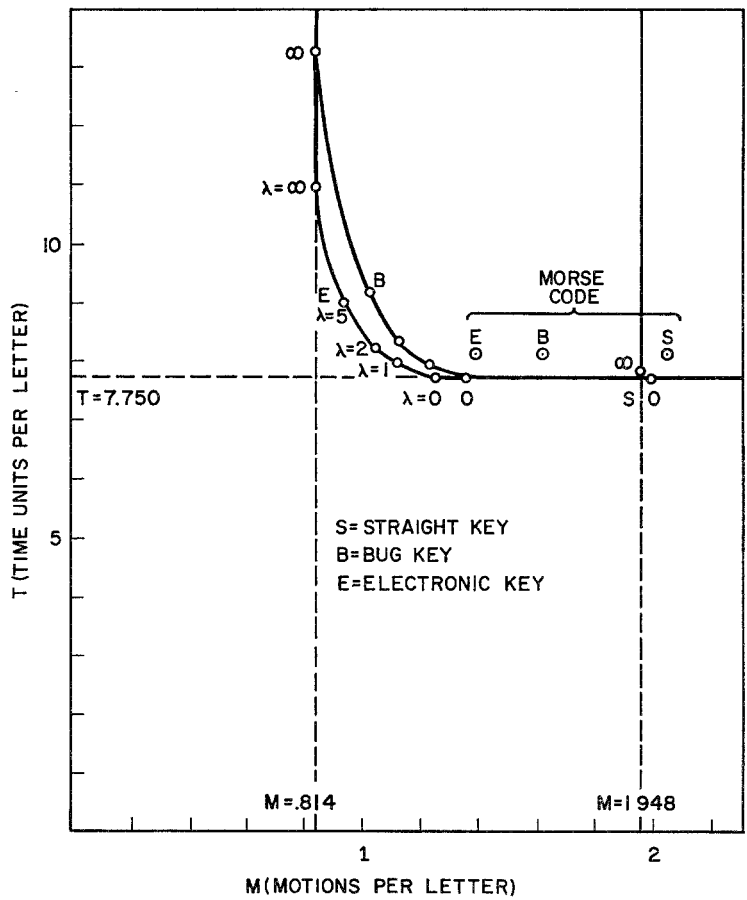

FIG. 1. Comparison between Morse code and codes which minimize mean time per letter for a fixed mean number of hand motions per letter.

Morse code is nearly the best code for straight key operation. For the other two keys the minimizing codes with the same T as the Morse code require roughly 20 % fewer motions per letter.

5. OTHER KEYS

The values of M in (3) show a significant advantage of automatic keys over a simple straight key. This section compares these keys with an optimal key.

In defining "optimal key" I want to consider only devices which are

controlled by a single paddle such as is used with electronic keys. Electronic keys are sometimes controlled by two independent paddles held between thumb and forefinger. This arrangement permits an additional kind of motion (squeezing the two paddles together) which will be forbidden here.

The allowed single paddle has three positions; center, left, and right. The center position is a rest position reserved for silent periods between letters. Any change of position to left or right counts as a motion; the final return to center does not count.

The following kind of switching circuit is allowed to convert the paddle motions into Morse elements. A *state* of the circuit is determined by the sequence of dots and dashes produced since the beginning of the letter. Thus, ϕ (the initial state), ·, ·-, ·--, and ·--· are the states encountered while the key forms the letter P. In state ϕ the operator has the option of a motion to left for a dot or to right for a dash. In later states the operator can make a motion (which moves the paddle to the opposite side), no motion, or (at the end of the letter) a return to ϕ. The circuit achieves a new state by producing a dot, a dash, or a letter space; the code element produced is a function of both the paddle movement, if any, and of the present state. In the *optimal key* this function is specified so as to minimize M.

The following very simple example, while not optimal, is an illustration. Suppose, for all states other than ϕ, that a motion is required for a dash but no motion is required for a dot. This key has $M = 1.376$, a small improvement.

To minimize M write M as a sum over states S

$$M = \sum_S p(S)m(S).$$

Here, $p(S)$ is the probability that state S is encountered in forming a random letter; $m(S)$ is the conditional probability at S that the letter in question requires a motion to pass from S to the next state. The circuit design has no influence on $p(S)$, for $p(S)$ is just the combined probability of all letters having S as a prefix. To minimize $m(S)$, and hence M, the next state immediately following S must be determined in such a way that no motion is required for the most likely next state while a motion is required for the other possible next state. For instance, consider the state S = - - and its possible next states - -· and - - -;

$$p(\text{- -·}) = \text{prob } (G, Z, \text{or } Q) = .0165$$

$$p(\text{- - -}) = \text{prob } (\text{O}) = .0632.$$

In state S one would require a motion to produce a dot, no motion for a dash (a return movement would also be allowed for letter M). In this way one obtains an optimal key with $M = 1.256$.

Note that the optimal key has been defined in such a way as to rule out some reasonable possibilities. Only the simplest motion was allowed. Even motions of a bug key (from dash to center and back to dash again) were forbidden. To allow this would give the operator a choice of two motions instead of one at each state. With two motions I can design a key, somewhat like the above optimal key, having $M = 1.1080$. Since its motions are more complicated it is not clear that this key is a real improvement over the optimal key. Both keys might be very confusing to an operator because they relate hand motions to dots and dashes in a complicated way. Another kind of forbidden key is one with typewriter keyboard and buffer storage; for an early (1915) mechanical key of this kind see Habig (1963). Although one might assign such a key the value $M = 1$, it is probably a much more convenient key than the electronic key or its generalizations. The comparison between M values might be more sensible if the typist were restricted to use one finger.

Starting with Morse code and an electronic key, one reduces M by about 10% by adopting the optimal key. Alternatively, one could retain the electronic key but redesign the code to achieve the same T and a 22% reduction in M. An interesting unsolved problem is to find a code and key when both are optimized together to minimize T subject to a constraint on M.

RECEIVED: April 30, 1969

REFERENCES

DEWEY, G. (1923). "Relative Frequency of English Speech Sounds," p. 185. Cambridge University Press, Cambridge.

FCC (1955). "FCC Study Guide and Reference Material for Commercial Radio Operators Examinations," p. 261 U. S. Govt. Printing Office, Washington.

GILBERT, E. N. AND MOORE, E. F. (1959). Variable-length binary encodings. *BSTJ* **38**, 933–968.

HABIG, H. A. (1963). A telegraph key with a memory. *QST* **47**, 70–71.

HENNEY, K. (1959). "Radio Engineers Handbook," p. 24–3. McGraw Hill, New York.

KAUFMAN, M. (1957). "Radio Operators License Question and Answer Manual," p. 443. John F. Rider Publ., New York.

MARSHALL, W. P. AND WHEELER, L. K. (1966) Telegraph. In *Encyclopedia Britannica.*