A Binary Spelling Interface with Random Errors

Jouri Perelmouter and Niels Birbaumer

***Abstract*—An algorithm for design of a spelling interface based on a modified Huffman's algorithm is presented. This algorithm builds a full binary tree that allows to maximize an average probability to reach a leaf where a required character is located when a choice at each node is made with possible errors. A means to correct errors (a delete-function) and an optimization method to build this delete-function into the binary tree are also discussed. Such a spelling interface could be successfully applied to any menu-orientated alternative communication system when a user (typically, a patient with devastating neuromuscular handicap) is not able to express an intended single binary response, either through motor responses or by using of brain–computer interfaces, with an absolute reliability.**

***Index Terms*—Alternative communication, binary tree, Huffman's algorithm, spelling interface.**

I. Introduction

VERBAL communication with a binary signal requires a spelling interface (SI) which transforms a binary code into characters of an alphabet [1]. The same is true for any kind of menu-oriented computer control: in this case menu items can be considered as special letters, and the whole "alphabet" is a set of them. For the overwhelming majority of applications, where longer texts are written, the fastest SI, i.e., the SI with a minimum selection time per letter, may be designed as a full binary tree that is built by using the famous Huffman's algorithm [2], [3]. In this case the binary signal has to be delivered into the SI without error. This assumption may be justified for most of communication systems which are used for alternative communication, i.e., for people with devastating neuromuscular handicaps [4]. However, for more severe neurological patients even simple movements may become unreliable, which leads to the necessity of introducing error terms, i.e., taking into account the probability of a single binary response different from 1. In even more severe cases, the "locked-in" syndrome manifests itself in a complete paralysis of all muscles [5]. For such patients direct brain–computer communication devices are being developed in which patients select different characters or menu-items using their electrical brain activity. The accuracy level for voluntary brain control is also lower than 1 [6]–[8]. By using the Thought Translation Device (TTD) [1], [6], for example, when choosing an intended binary signal, the patient generates an analog signal $s(t)$ (in this case slow cortical potential shift—SCP) whose amplitude must achieve (single selection) or not achieve (single rejection) a certain threshold level S. Obviously, two kinds of errors ("slips of the pen") can take place here: missing the correct selection, because the amplitude of $s(t)$ is too small to achieve S, and false selection instead of correct rejection, because of uncontrolled noise in $s(t)$ which randomly exceeds S ("false alarms").

The use of a binary tree as SI implies that the writing process for one character starts from the origin of the tree and proceeds by single selections or rejections at each internal node which lies on the path from the origin to a leaf where this character is located. We make the convention that each single selection at an internal node means a choice of its left child, and each single rejection—a choice of its right child. The choice of the leaf means an addition of the character from this leaf to the final text. Let p denote the probability of a single correct selection and q that of a single correct rejection, respectively, Ω is the alphabet from which the characters are drawn so that the full binary tree T is used as SI for Ω and thus consists of exactly $n = |\Omega|$ leaves for each character of the alphabet and exactly $n - 1$ internal nodes. For each character c in the alphabet Ω, let $f(c)$ denote the frequency of c in the alphabet (for example, in the natural language), and let $A(c)$ denote the probability of the unerring attainment of character c (the probability of reaching the leaf, where character c is located, without any errors) using a specific sequence of single selections and single rejections.

Since "writing" selection errors can occur at each step of the writing process with nonzero probabilities, and the attempt to write one character can lead to the appearance of some other character in the final text, it is natural as a criterion to build an optimal SI to use the average probability to write one character without any errors

$$\Phi = \sum_{c \in \Omega} f(c) \cdot A(c) \to \max. \tag{1}$$

In the first part of this paper, we present an algorithm that constructs the optimal tree according to criterion (1); this algorithm is a certain generalization of Huffman's algorithm for the case when each parent node of the tree has left and right children with different weights. In the second part we discuss a means to correct possible errors in writing and present a method to build this means into the optimal binary tree.

II. Algorithm

We write the classical Huffman's algorithm as H1 and the modified algorithm as H2. The algorithm H2 also builds the tree T corresponding to the optimal Φ in a bottom-up manner. Like H1, it begins with a set of n leaves and performs a sequence of $n - 1$ merging operations to create the final tree. A priority

Manuscript received November 18, 1999; revised February 15, 2000 and February 25, 2000. This work is supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Society) and the Institut für Grenzgebiete in der Psychologie und Psychohygiene (Freiburg, Germany).

J. Perelmouter is with the Institute of Medical Psychology and Behavioral Neurobiology, University of Tübingen, Germany (e-mail: J.Perelmouter@uni-tuebingen.de).

N. Birbaumer is with the the Institute of Medical Psychology and Behavioral Neurobiology, University of Tübingen, Germany. He is also with the Dipartimento di Psicologia Generale, Università degli Studi, Padova, Italy.

Publisher Item Identifier S 1063-6528(00)04240-3.

queue [3] may be used to identify the two least-frequent objects (characters or sets of characters) to merge together. The result of the merger of two objects is a new object whose frequency is the sum of the frequencies of the two weighted by p and q objects merged, respectively. Let us assume without loss of generality that $p \leq q$, and let a and b denote these merged two objects; if $f(a) \leq f(b)$ then the frequency of the new object is $f_{\text{new}} = p \cdot f(a) + q \cdot f(b)$, otherwise it is $f_{\text{new}} = p \cdot f(b) + q \cdot f(a)$. Thus the algorithm H2 can be described as follows.

1) Let L be a list of the frequencies of the characters corresponding to the leaves of the tree.
2) Take the two smallest frequencies in L, make the corresponding nodes siblings so that the left child's frequency is not greater than the right child's frequency, and generate an intermediate node as their parent.
3) Replace the above-mentioned two frequencies in L with their sum weighted by p and q (p for the left child, q for the right child), associated with the new intermediate node. If the new L contains only one element, then stop, otherwise return to step 2).

The total running time of the algorithm H2 for a set of n characters, if binary heap for the priority queue is implemented, is $O(n \lg n)$ and is the similar to the running time of the algorithm H1 [3]. The proof of correctness of H2 is also almost identical, based on the following lemmas and adduced here just to make the statement complete.

Lemma 1: Let x and y be characters in the alphabet Ω with frequencies $f(x) < f(y)$ that appear in the optimal tree T as sibling leaves. Then if $p < q$, x must appear as a left child; if $p > q$—as a right child.

Proof: Let us assume that $p < q$, and x is a right child and y is a left child of the parent node (x, y). We exchange the positions of x and y to produce a new tree U. Then

$$\begin{aligned}\Phi(T)-\Phi(U) &= \sum f(c)\cdot A_T(c) - \sum f(c)\cdot A_U(c)\\ &= f(x)\cdot q\cdot A(x, y) + f(y)\cdot p\cdot A(x, y) - f(x)\\ &\quad \cdot p\cdot A(x, y) - f(y)\cdot q\cdot A(x, y)\\ &= A(x, y)\cdot[f(x)-f(y)]\cdot(q-p) < 0\end{aligned}$$

where $A(x, y) > 0$ is the probability to reach the parent node (x, y). This last inequality contradicts the optimality of T, so our assumption is incorrect, and x is the left child and y is the right child of the parent node.

Lemma 2: Let x and y be two characters in alphabet Ω having the lowest frequencies. Then in the optimal tree the characters x and y appear as sibling leaves of minimum probabilities $A(x)$ and $A(y)$, and the character with the lowest frequency corresponds to the lowest child.

Proof: Without loss of generality we can assume that $p < q$, $f(x) < f(y)$, and T is the optimal tree. Let also b and c be two characters that are sibling leaves of minimum probabilities $A_T(b)$ and $A_T(c)$ in T, and assume $f(b) < f(c)$. Since $f(x)$ and $f(y)$ are the two lowest leaf frequencies, in order, and $f(b)$ and $f(c)$ are two arbitrary frequencies, in order, we have $f(x) < f(b)$ and $f(y) < f(c)$. We exchange the positions in T of b and x to produce a tree U, and then we exchange the positions in U of c and y to produce a tree V

$$\begin{aligned}\Phi(T) - \Phi(U) &= \sum f(c)\cdot A_T(c) - \sum f(c)\cdot A_U(c)\\ &= f(x)\cdot A_T(x) + f(b)\cdot A_T(b) - f(x)\\ &\quad \cdot A_U(x) - f(b)\cdot A_U(b)\\ &= f(x)\cdot A_T(x) + f(b)\cdot A_T(b) - f(x)\\ &\quad \cdot A_T(b) - f(b)\cdot A_T(x)\\ &= [f(b)-f(x)]\cdot[A_T(b) - A_T(x)] \leq 0.\end{aligned}$$

Similarly, because exchanging y and c does not increase Φ, $\Phi(U)-\Phi(V)$ is nonpositive. Thus V is an optimal tree in which x and y appear as sibling leaves of minimum probabilities, from which the lemma follows.

Lemmas 1 and 2 represent the greedy-choice property: a globally optimal solution can be arrived at by making a locally optimal (greedy) choice [3].

Lemma 3: Let T be a full binary tree representing the optimal solution of the problem (1) over the alphabet Ω, where frequency $f(c)$ is defined for each character $c \in \Omega$. Consider any two characters x and y that appear as sibling leaves in T, and let (x, y) be their parent. Then considering (x, y) as a character with frequency $f(x, y) = pf(x) + qf(y)$ or $f(x, y) = qf(x) + pf(y)$, the tree $U = T - \{x, y\}$ represents the solution of this problem for the alphabet $\Xi = \Omega - \{x, y\} \cup \{(x, y)\}$.

Proof: We first show that $\Phi(T)$ of tree T can be expressed in terms of $\Phi(U)$ of tree U. For each $c \in C - \{x, y\}$, we have $A_T(c) = A_U(c)$, and hence $f(c)A_T(c) = f(c)A_U(c)$. Since $A_T(x) = pA_U(x, y)$ and $A_T(y) = qA_U(x, y)$ or $A_T(x) = qA_U(x, y)$ and $A_T(y) = pA_U(x, y)$, we have $f(x)\cdot A_T(x) + f(y)\cdot A_T(y) = A_U(x, y)\cdot[p\cdot f(x) + q\cdot f(x)] = f(x, y)\cdot A_U(x, y)$, from which we conclude that $\Phi(T) = \Phi(U)$. If U represents a nonoptimal tree for the alphabet Ξ then there exists a tree V whose leaves are characters in Ξ such that $\Phi(V) > \Phi(U)$. Since (x, y) is treated as a character in Ξ, it appears as a leaf in V. If we add x and y as children of (x, y) in V, then we obtain a tree, whose leaves are characters in Ω with

$$\Phi = \Phi(V) > \Phi(U) = \Phi(T)$$

contradicting the optimality of T. Thus, U must be optimal for the alphabet Ξ.

Lemma 3 represents the optimal-substructure property: an optimal solution to the problem contains within it optimal solution to subproblems [3].

Theorem: Procedure of the algorithm H2 produces an optimal binary tree for the problem (1) over the alphabet Ω.

Proof: This theorem immediately follows from lemmas 1, 2, and 3.

Fig. 1 illustrates an example of an application of H2 in the case when $p = 0.7, q = 0.9$ to the alphabet $\Lambda = \{\text{A}(0.4), \text{B}(0.3), \text{C}(0.2), \text{D}(0.1)\}$ which consists of four letters: A, B, C, and D. Here and further we put in brackets after each character its occurrence frequency in the alphabet. Fig. 2 shows, as another example, optimal binary trees for the alphabet $\text{M} = \{\text{A}(0.35), \text{B}(0.3), \text{C}(0.2), \text{D}(0.15)\}$ when $p = q = 1$ (Huffman's tree); $p = q = 0.9$, $p = q = 0.8$, and $p = 0.6$, $q = 0.7$. One could see that optimal tree-structures,

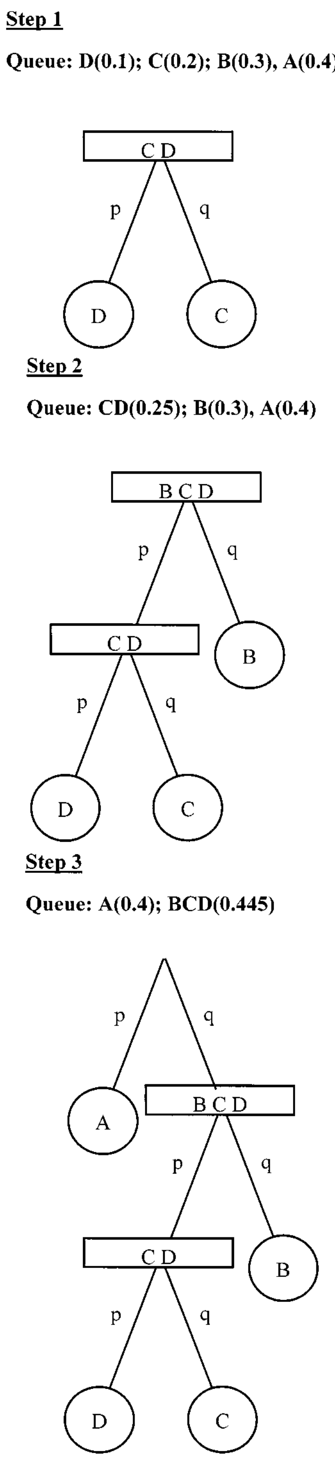
Step 1
Queue: D(0.1); C(0.2); B(0.3), A(0.4)
CD
p
q
D
C
Step 2
Queue: CD(0.25); B(0.3), A(0.4)
BCD
CD
B
Step 3
Queue: A(0.4); BCD(0.445)
A

Fig. 1. The steps of the algorithm H2 for the alphabet $\Lambda = \{A(0.4), B(0.3), C(0.2), D(0.1)\}$ when $p = 0.7$ and $q = 0.9$. Each part shows the contents of the queue sorted into increasing order by frequency. At each step, the two trees with lowest frequencies are merged.

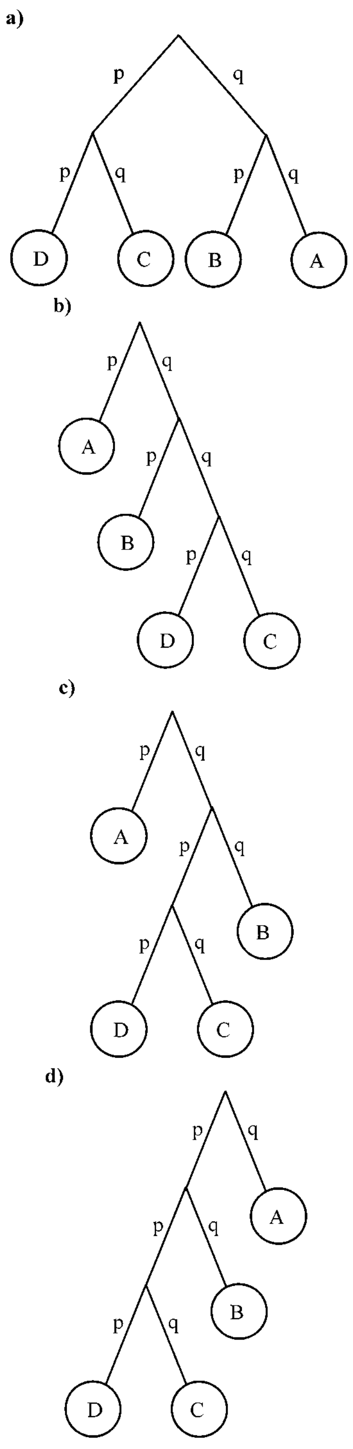

Fig. 2. Optimal binary trees for the alphabet $M = \{A(0.35), B(0.3), C(0.2), D(0.15)\}$: (a) one of possible Huffman's trees $p = q = 1$, (b) $p = q - 0.9$, (c) $p = q = 0.8$, and (d) $p = 0.6$, $q = 0.7$.

which are generated for different probability values (p, q), can be absolutely different.

III. Delete-Function

As it was shown above, the algorithm H2 builds an optimal tree as a base of an SI corresponding to any acceptable pair (p, q): $0 \leq p$, $q \leq 1$, and in the case of $p = q = 1$ produces the Huffman's tree. Clearly, if at least one of the values p and q is not equal to 1, then an error (writing a false character) is possible. The average probability of making an error is $1 - \Phi$. As a means to correct the final text we can use a *delete-function* or a *backspace-function*—a special character whose choice erases the last character of the final text similar to the action of a backspace-key on a computer keyboard. This special character must be added to the original alphabet and can be chosen with the same procedure (a certain sequence of single selections/rejections) as any other character of the alphabet.

Since the delete-function is represented in the binary tree as a special character, a reasonable question arises: how is it possible to produce the optimal binary tree over the alphabet Ξ which contains the original alphabet Ω as well as the special delete-character which should appear *as frequently as the user requires it?* It seems reasonable that the probability to select this character should be not less than the average probability of making an error in writing one character from the alphabet Ω. To answer this question let us consider the addition of this new delete-character χ with an "additional" frequency δ to the alphabet Ω so that the new frequency of each $c \in \Omega \subset \Xi$ over the alphabet $\Xi = \Omega \cup \{\chi\}$ is

$$f(c) = (1-\delta)f(c).$$

The criterion (1) "the average probability to select one character without any errors" over the alphabet Ξ looks as

$$\Phi = (1-\delta)\sum_{c\in\Omega} f(c)\cdot A(c) + \delta\cdot A(\chi) \to \max \quad (2)$$

where $A(\chi)$ is the probability of the unerring attainment character χ. Note that with the help of H2 we can obtain the binary tree $T(\delta)$ which is optimal in the sense of (2) for each $\delta \in (0, 1)$.

The probability of making an error when selecting of a character from the alphabet Ω is

$$(1-\delta)\cdot\sum_{c\in\Omega} f(c)\cdot(1-A(c)) = (1-\delta)\cdot\left(1-\sum_{c\in\Omega} f(c)\cdot A(c)\right)$$

but only in $n-1$ out of n cases this leads to the writing of a false character into the final text because the character χ is also included in the tree. So the requirement that the probability to select character χ should be not less than the average probability of making an error in writing one character from the alphabet Ω could be expressed as

$$\delta\cdot A(\chi) \geq (n-1)\cdot(1-\delta)\cdot\left(1-\sum_{c\in\Omega} f(c)\cdot A(c)\right)\Big/ n. \quad (3)$$

Criterion (2) defines the objective "to maximize the probability to select any character from the alphabet Ξ" including the delete character χ. But the actual objective of optimal verbal (or menu-oriented) communication is to select "usual" characters from the alphabet Ω preferably, and to use this special delete character χ only when it is needed to correct a false choice. This means that in our case the delete character's frequency of occurrence δ must be as small as possible. So the problem to build the optimal binary SI-tree, which contains the alphabet Ω and the special delete-character, can be solved as an optimization problem

$$\delta \to \min \quad (4)$$

with the constrains $\delta \in (0, 1)$ and (3), where all the probabilities $A(c)$ and $A(\chi)$ must correspond to the optimal binary tree $T(\delta)$ which is built according to criterion (2). This can be realized as a successive increase of a δ-value from zero with a permanent monitoring the constrain (3). In this way the first value of δ which is met with and satisfies inequality (3) is the required solution. At any step of this procedure (for any considered value of δ) the algorithm H2 should be used to build the tree $T(\delta)$ and to calculate the probabilities $A(c)$ and $A(\chi)$.

Fig. 3 illustrates different optimal SI-trees with the delete-function for the above-mentioned alphabet Λ. It can be seen how the χ-leaf, i.e., the leaf where the delete-character χ is located, moves from deep tree levels to a top tree level as the single choice probability decreases. Such a decrease depresses the value of the average probability to select one "usual" (not delete) character without any errors

$$S = \sum_{c\in\Omega} f(c)\cdot A(c)$$

and thus increases the required delete-function's frequency δ. The graph on Fig. 4 demonstrates this relationship for the English alphabet E $=$ {SPACE(0.1905), E(0.1005), T(0.0721),..., Z(0.0008)}when values of selection/rejection probabilities are equal ($p = q$). It is interesting to note that in this case the χ-leaf is placed at the top level of the tree whenever this probability value is less than 0.952, which means that the delete-function becomes the most frequent character in the alphabet $E \cup \{\chi\}$.

It should also be noted that the delete-function is not the only possible means to correct communication errors in binary SI. The other type of correction—a back-up function, which allows a return to previous tree-levels—is discussed in detail in [1]. With only binary signals it is relatively difficult to find efficient means of error correction; any kind of "confirmations," "response verification" [7] etc. are destined not for the error-correction but for improvement of the response-accuracy. Many more possible means of correction result from using of k-ary responses ($k > 2$) [7], [9] which also need an optimal k-ary tree design.

IV. Discussion

The proposed algorithm and method may be useful for any type of communication system, that is controlled through a binary (YES/NO) response and where errors are possible. Even with the use of commercial systems for alternative communication with motor response control, a patient very often is not able to produce an absolutely reliable single binary response. A typical patient's estimation "almost absolutely reliable—not more than 5–10% of errors" is equivalent to the choice-probability value 0.90–0.95, which leads, as shown above, to a considerable modification of the corresponding optimal binary SI-tree. Algorithms such as the one described become even more useful with the use of Brain-Computer-Interfaces where response-accuracy values of 0.7–0.9 are not rare [1], [6]–[8].

As an example of the method's application we can consider our TTD-LSP (Language Support Program) system [1], [6]. In the used paradigm, patients sit in wheelchairs or beds and view a PC monitor on which two rectangles (goals) and a small moving object (ball) are displayed. A vertical ball coordinate on the PC screen is calculated as a linear function of SCP shift, therefore controlling their own SCP shifts patients can move the ball

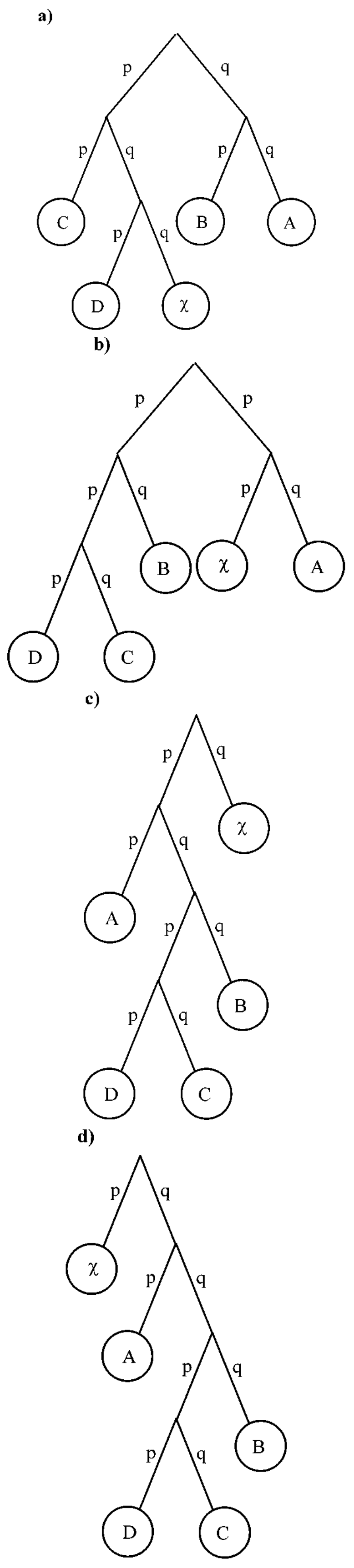

Fig. 3. Optimal binary trees with the delete-function (character χ) for the alphabet $\Lambda = \{A(0.4), B(0.3), C(0.2), D(0.1)\}$: (a) $p = 0.9$, $q = 1$, (b) $p = 0.8$, $q = 1$, (c) $p = 0.7$, $q = 0.9$, and (d) $p = 0.6$, $q = 0.7$.

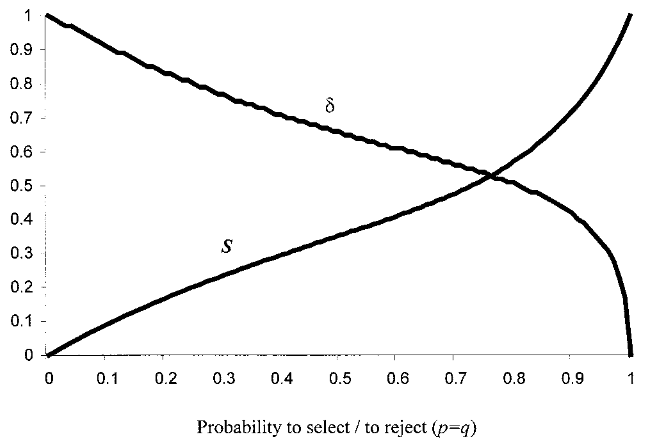

Fig. 4. Probability S to write one English character without any error and frequency δ of the delete-function depend on the probability to select/to reject when $p = q$.

across the computer screen. Two alternated tones of different pitch, which follow each other with an interval of 2 s hold the patient to the rhythm of the session. During the first 2-s phase between the high-pitch tone and low-pitch tone (baseline phase) the ball remains in the center of the screen. It can move only during the second phase (active phase) between the low-pitch tone and the next high-pitch one. During the baseline phase patients are presented letters, and in the active phase, they can select these letters by using the corresponding brain response (SCP): the patient produces a signal whose amplitude must achieve (single selection) or not achieve (single rejection) a certain threshold level. Since it is very difficult to control both cortical negativity and cortical positivity simultaneously, and not every person can do it with the same reliability, we use only one direction as a single selection response—either cortical negativity or cortical positivity; the other direction is counted (together with small amplitude values) as an absence of the selection response, i.e., as the single rejection response.

In order to implement such a binary SI, at each writing step (at the origin of the tree or at a corresponding internal node of the tree) letters, which are attainable from this particular origin or internal tree node by the rejection response (the right sub-tree), can be presented to the subject in the upper goal. A set of letters, which are attainable by the selection response (the left sub-tree), can be presented in the opposite goal. While using the TTD-LSP system during the baseline phase, the subject has to consider in which node of the tree he/she is at the moment (a navigational prompt) and which kind of response (selection or rejection) he/she want to produce during the following active phase. After some leaf of the tree is reached, the corresponding letter is written in the text field of the screen (or the last written letter is deleted), and the process starts again from the origin of the tree.

As letters can be used either "genuine" letters of an alphabet (as the rule, we use in the LSP a set of 32 symbols—all German letters and three punctuation marks) or items of some menu (one of our patients, for example, communicates by means of some complete sentences, expressing his everyday needs such as "switch on TV-set" or "open window"). In the same manner the proposed method (or rather an optimized with its help binary tree) can be applied to any augmentative communication device which uses the binary kind of responses—for example, to "head-switch" communication devices.

It is important to emphasize that the method is designed not to reduce the redundancy of an alphabet in use. For example, in order to design the optimal binary SI which uses the traditional Morse code (four symbols—"dot," "dash," "end of character," and "delete") we are not interested in the redundancy of this code, which has to be considered as a given alphabet, but only in frequencies of occurrence of the first three symbols during writing of long texts and in the user's ability to make a correct response—the value of probability of a single correct selection and that of a single correct rejection. The first kind of data could be obtained from a direct statistical analysis of relatively long texts, the second one—from permanent monitoring. For example, for each of our patients we always have the actual values of their probabilities of correct responses since monitoring continuously takes place—before every writing session each patient has to copy some sequence of single responses presented by his/her trainer.

To reduce redundancy of an alphabet, there could be different approaches to this important problem. One of them might be to supply the system with a vocabulary support or with an automatic spell checker. We use such an approach for one of our completely locked-in patients with amyotrophic lateral sclerosis, who has his actual error probability values about 0.05–0.15 and can communicate words by means of the TTD-LSP (he uses the SI described in [1]). The system suggests to him a word from his individual vocabulary after some letters have been written. The patient can either confirm this word or not pay attention to this word and continue writing a word different from the suggested one. The different approach might be the using of Markov-features of natural languages (so-called bigramm/trigramm/k-gramm probabilities). Here a proper utilization of the fact, that frequencies of letters' occurrence must be efficiently corrected depending on an already written text, could reduce a redundancy of an alphabet in use. A detailed discussion of such a redundancy reduction is outside the scope of this paper. However, it should be noted that almost every reduction approach can be easily combined with the suggested method.

References

[1] J. Perelmouter, B. Kotchoubey, A. Kübler, E. Taub, and N. Birbaumer, "A language support program for thought translation devices," *Automedica*, vol. 18, pp. 67–84, 1999.

[2] D. A. Huffman, "A method for the construction of minimum redundancy codes," *Proc. IRE*, vol. 40, pp. 1098–1101, 1952.

[3] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*. Cambridge, MA: The MIT Press, 1997, ch. 17, pp. 337–343.

[4] D. Beukelman and P. Mirenda, *Augmentative and Alternative Communication: Management of Severe Communication Disorders in Children and Adults*. Baltimore, MD: Paul H. Brooks, 1992.

[5] A. Oksenberg, N. Soroker, P. Solzi, and I. Reider-Groswasser, "Polysomography in locked-in syndrome," *Electroencephalogr. Clin. Neurophysiol.*, vol. 78, pp. 314–317, 1991.

[6] N. Birbaumer, N. Ghanayim, T. Hinterberger, I. Iversen, B. Kotchoubey, A. Kübler, J. Perelmouter, E. Taub, and H. Flor, "A spelling device for the paralyzed," *Nature*, vol. 398, pp. 297–298, Mar. 1999.

[7] J. R. Wolpaw, H. Ramoser, D. J. McFarland, and G. Pfurtscheller, "EEG-based communication: Improved accuracy by response verification," *IEEE Trans. Rehab. Eng.*, vol. 6, pp. 326–333, Sept. 1998.

[8] G. Pfurtscheller, C. Neuper, A. Schlögl, and K. Lugger, "Separability of EEG signals recorded during right and left motor imagery using Adaptive autoregressive parameters," *IEEE Trans. Rehab. Eng.*, vol. 6, pp. 316–325, Sept. 1998.

[9] J. Perelmouter, D. J. McFarland, N. Birbaumer, L. A. Miner, and J. R. Wolpaw, "A spelling interface for an EEG-based communication system," in *Abstr. 28th Ann. Meet. Soc. Neurosc.*, 1998, p. 656.

Jouri Perelmouter received the M.Sc. degree in mechanics and mathematics from Moscow State University, Russia, in 1971 and the Ph.D. degree in engineering from Moscow Engineering Institute, Russia, in 1979.

Between 1971 and 1993, he worked as a Senior Researcher and as a Head of Laboratory on numerous projects in applied mathematics in industry and environmental protection in Russia. Since 1994, he has been working in medical physics and psychology in Germany. He is currently a Senior Researcher at the Institute of Medical Psychology and Behavioural Neurobiology, the University of Tuebingen, Germany, where he has been involved in the research program Thought Translation Device since 1996.

Dr, Perelmouter was elected to an Academician of the International Informatization Academy (IIA) in 1997.

Niels Birbaumer was born in 1945 and received the Ph.D. degree in psychology and statistics from the University of Vienna, Austria, in 1969.

In 1975, he was a Full Professor of Clinical and Physiological Psychology, University of Tuebingen, Germany. From 1986 to 1988, he was a Full Professor of Psychology, Pennsylvania State University, University Park. Since 1993, he has been a Professor of Medical Psychology and Behavioral Neurobiology at the Faculty of Medicine of the University of Tuebingen, and Professor of Clinical Psychophysiology, University of Padova, Italy. His research interests are in neuronal basis of learning and plasticity, neurophysiology and psychophysiology of pain, behavioral medicine, and neurorehabilitation. He has authored/coauthored more than 400 publications in peer-reviewed journals and authored/coauthored 12 books.

Dr. Birbaumer is a member of the German Academy of Science and Literature, president of the European Association of Behavior Therapy, Fellow of the American Psychological Association, Society of Behavioral Medicine, American Association of Applied Psychophysiology. He is the recipient of the Leibniz-Award from the German Research Society (DFG).