A Stochastic Control Approach to Optimally Designing Hierarchical Flash Sets in P300 Communication Prostheses

Rui Ma, Navid Aghasadeghi, Julian Jarzebowski, Timothy Bretl, *Member, IEEE*, and Todd P. Coleman, *Senior Member, IEEE*

***Abstract*—The P300-based speller is a well-established brain–computer interface for communication. It displays a matrix of objects on the computer screen, flashes each object in sequence, and looks for a P300 response induced by flashing the desired object. Most existing P300 spellers uses a fixed set of flash objects. We demonstrate that performance can be significantly improved by sequential selections from a hierarchy of flash sets containing variable number of objects. Theoretically, the optimal hierarchy of flash sets—with respect to a given statistical language model—can be found by solving a stochastic control problem of low computational complexity. Experimentally, statistical analysis demonstrates that the average time per output character at 85% accuracy is reduced by over 50% using our variable-flash-set approach as compared to traditional fixed-flash-set spellers.**

***Index Terms*—Brain–computer interfaces (BCIs), hierarchical flash sets, stochastic dynamic programming, P300.**

I. Introduction

BRAIN–COMPUTER interfaces (BCIs) take measurements of brain activity and use them as inputs to control external devices. By shortcutting the motor system, BCIs have enabled a variety of activities that might otherwise be difficult or impossible for people with impaired sensory-motor function [1].

In this paper, we consider the use of BCIs to restore text communication for people who are unable to operate keyboards [2]–[5]. We focus exclusively on text communication using the P300-based speller [2], [3]. This well-established paradigm takes advantage of the P300 event-related potential (ERP), which is a response to infrequent but anticipated stimuli that can be observed as a deflection in the electroencephalogram (EEG)—often called a *P300 response*—approximately 300 ms after each relevant stimulus. The P300-based speller, in particular, uses visual stimuli to elicit P300 responses. A computer screen displays to the human subject a matrix of possible characters, subsets of which are illuminated in random order. The subject keeps a silent mental count of how many times their desired character has been illuminated. Each time this happens, a P300 response is observed in EEG. By detecting a sequence of P300 responses, the P300-based speller allows users to specify a sequence of characters, one at a time.

Perhaps the most widely used P300-based speller is the one introduced by Farwell and Donchin in 1988 [2]. In this case, 26 English characters and 10 digits are displayed in a matrix of six rows and six columns. Characters are illuminated in groups, one row or column at a time. A P300 response occurs when this row or column includes the desired character. After each row and column has been illuminated for a prespecified number of times, a statistical decision is made about which character was most likely to have been the desired one. We will call this approach the "row-column" (RC) paradigm.

Two decades of work since [2] has explored various ways to improve the performance of P300-based spellers. Most of this work has focused on making the detection of P300 signals more robust, either through better signal processing and classification algorithms, as summarized in [6], or through better ways to elicit P300 responses. For example, in contrast to illuminating the entire rows and columns as in [2], illuminating single characters has been found to improve classification results by eliciting much more robust and distinguishable P300 responses, presumably due to the much lower frequency of illuminating the desired character [7], [8]. We shall term this the "single-character" (SC) paradigm.

However, most P300-based spellers typically need to flash a fixed set of objects, and each object flashing more than once so that its responses can be averaged. It causes a delay which is proportional to the number of objects in a flash set and the number of trials used in averaging. This delay adds up quickly, and can easily approach half a minute for a conventional paradigm to spell a single character.

Recent work has focused on reducing this delay by using a variable number of trials to perform classification on a fixed

Manuscript received May 12, 2011; revised October 09, 2011; accepted December 03, 2011. Date of publication December 23, 2011; date of current version January 25, 2012. This work was partially presented as an invited paper at the IEEE Information Theory WorkshopPorto, Portugal,May2008.

R. Ma is with the Department of Bioengineering, University of California-San Diego, La Jolla, CA 92093 USA (e-mail: r5ma@ucsd.edu).

N. Aghasadeghi is with the Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA (e-mail: aghasad1@illinois.edu).

J. Jarzebowski is with Motorola Inc., Schaumburg, IL 60196 USA (e-mail: julian@jarzebowscy.com).

T. Bretl is with the Department of Aerospace Engineering and the Neuroscience Program, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA (e-mail: tbretl@illinois.edu).

T. P. Coleman is with the Department of Bioengineering and the Institute for Neural Computation, University of California-San Diego, La Jolla, CA 92093 USA (e-mail: tpcoleman@ucsd.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNSRE.2011.2179560

set of flash objects. In [9], the theoretical bit rate was significantly boosted by maintaining a posterior belief on characters and making a classification when that belief is sufficiently strong towards one character. For SC paradigms, Markov decision theory [10] has been employed to determine the optimal pattern of character illuminations in a flash set so that more likely characters are queried more often [11]. However, none of these studies have considered variable flash sets.

The purpose of this paper is to introduce a novel paradigm, called the "dynamic-programming" (DP) paradigm, which sequentially presents a hierarchy of nested flash sets, each of which represents a subset of all possible characters. The optimal hierarchy of flash sets–with respect to a given statistical language model—can be found by solving a dynamic programming problem of low computational complexity [12]. We show that in any optimal strategy, more likely characters appear early on for querying in the hierarchy. For example, in the English language, after the characters "t" and "h", it is much more likely that a vowel will be next as opposed to a consonant. So rather than showing all the possible stimuli, one strategy would be to only include vowels in the next flash set; if no vowel was selected, then a subsequent group of flash sets would contain the remaining constants. In the worst case, the subject might have to navigate through multiple flash sets to select one desired symbol. But given a good statistical language model, this happens rarely. By using a smaller flash set on average, we reduce the expected time to specify a single character.

In Section II, we define our terminology and illustrate it with a simple example. In Section III, we develop the theory underlying the DP paradigm. In Section IV, we describe both our implementation of DP in the BCI2000 framework [13] and our experimental protocol for validation with human subjects. In Section V, we present experimental results and compare the performances. Statistical analysis shows that the average time per character with 85% accuracy is reduced by over 50% using DP paradigm as compared to RC and SC.

II. Definitions

A. General Statement

Throughout this paper, we will be considering a scenario where a subject attempts to spell a sequence of M characters sequentially. To specify a specific character in the SC and RC paradigms, the system randomly illuminates all the characters. The DP paradigm first queries a subset of characters; if the desired one lies in that subset, it is classified. If not, the system queries another subset, and this continues until the desired character is specified. The optimal way of selecting these subsets will be specified in Section III. The purpose of this section is to simultaneously describe all three paradigms—RC, SC, and DP—with unified set-theoretic terminologies. We illustrate these definitions with simple alphabets, examples, and figures.

Set Theory. Denote $\emptyset$ as the empty set. For any set A, denote $|A|$ its size. Denote 2^A as the set of all subsets of the set A. Note that $|2^A| = 2^{|A|}$. For two sets A and B, denote $A \setminus B$ as the set of all elements of A that are not contained in B, $A \setminus B \triangleq \{a' \in A \, s.t. \, a' \notin B\}$. We denote random variables by capital letters (e.g., U) and sample-path realizations by lowercase letters (e.g., u).

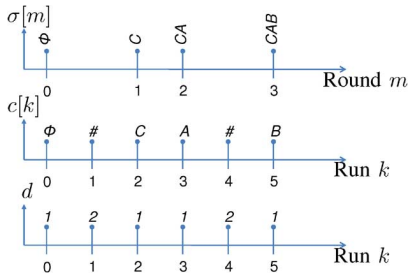

Fig. 1. An example illustrates the relevant time scales for DP paradigm by showing the sequence of events to spell the word "CAB" using the setup in Fig. 2. The runs are the time scale of classifications, while the rounds are the time scale of updating the sequence spelled out so far.

Fig. 2. Example display matrix containing only "A","B", "C", and "♯".

B. Specifics According to Time Scale

In this section we iterate through definitions according to time scales, from broad to narrow.

1) Round: Denote $\mathcal{U}$ as the set of all possible characters in any sentence the subject wants to convey. Throughout the remainder of this paper, we assume $\mathcal{U} = \{\text{"}A\text{"}, \text{"}B\text{"}, \ldots, \text{"}Y\text{"}, \text{"}Z\text{"}, \text{"\_"}\}$, where "_" represents the space character.

A round is indexed by m and pertains to the time scale over which a subject sequentially spells a character $U_m \in \mathcal{U}$. The spelled sequence at the end of round m is denoted by $\sigma[m] = (U_1, \ldots, U_m)$. See Fig. 1 (top), where $\sigma[1] = (\text{"}C\text{"})$ and $\sigma[2] = (\text{"}C\text{"}, \text{"}A\text{"})$.

2) Run: A run is indexed by k and pertains to the time scale over which classifications occur.

Control Signals. Denote $\mathcal{C}$ as a set of additional symbols, that convey control-like signals in the event of misclassification or sequentially navigating subsets of $\mathcal{U}$. In real implementations, $\mathcal{C}_{\text{RC}} = \mathcal{C}_{\text{SC}} = \{\text{" } \Leftarrow \text{"}\}$, denoting the backspace character to delete the previously selected symbol in the event of a misclassification. In real implementations for the DP paradigm, $\mathcal{C}_{\text{DP}} = \{\text{"}\sharp\text{"}, \text{" } \Leftarrow \text{"}\}$. If U_m lies in the subset being queried, the subject selects it. Otherwise, the subject selects the symbol "♯" and the next subset of $\mathcal{U}$ is queried after this classification.

Alphabet and Display Matrix. The alphabet containing all possible symbols that can be classified during a run is given by $\bar{\mathcal{U}} = \mathcal{U} \cup \mathcal{C}$. Denote $\mathcal{D}$ to be a display matrix in predetermined configuration to present $\bar{\mathcal{U}}$. For a simple illustration limited to this section, let $\mathcal{U} = \{\text{"}A\text{"}, \text{"}B\text{"}, \text{"}C\text{"}\}$ and $\mathcal{C} = \{\text{"}\sharp\text{"}\}$, and they are arranged in the 2×2 display matrix in Fig. 2.

Query Subsets and Policies. The query subset, $\mathcal{G} \subseteq \bar{\mathcal{U}}$, is the subset of symbols in $\bar{\mathcal{U}}$ that are illuminated during a run. For the RC and SC paradigms, $\mathcal{G} = \bar{\mathcal{U}}$. For the DP paradigm, $\mathcal{G}$ on any run is allowed to be a subset of $\bar{\mathcal{U}}$. Any sequence $\pi_{\text{DP}} = (\mathcal{G}_1, \ldots, \mathcal{G}_D)$ for which $\bigcup_{d=1}^{D} \mathcal{G}_d = \bar{\mathcal{U}}$ will be termed a **policy**. In run k, the system first queries a subset of symbols at **depth**

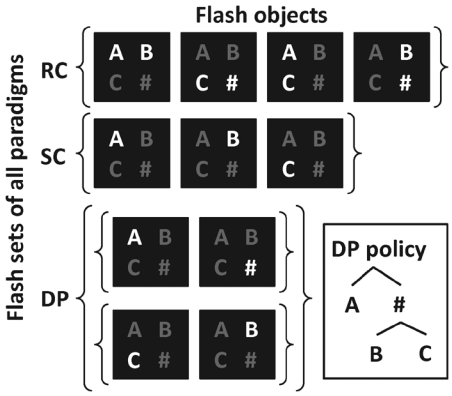

Fig. 3. Examples illustrating the definitions of flash objects, flash sets, and tree of policy.

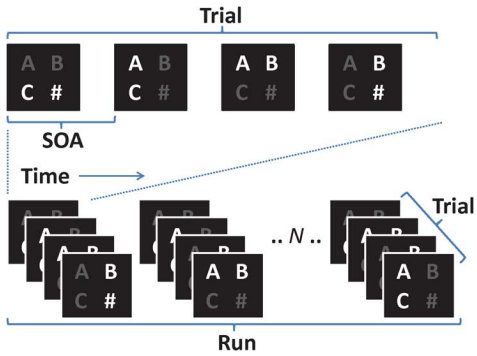

Fig. 4. Examples illustrates trial, run and SOA in RC. It is straightforward to generalize to SC and DP.

$d = 1$, termed $\mathcal{G}_1$. If the desired character U_m lies in $\mathcal{G}_1$, it is classified. If not, then the "other" symbol denoted by "♯" is classified. At the next run $k+1$ the depth becomes 2 and the system queries the subset at depth $d = 2$, $\mathcal{G}_2$. This continues until U_m is classified. Denote $c[k] \in \bar{\mathcal{U}}$ as the symbol classified after run k (see Fig. 1 middle). Note that $\pi_{\rm DP}$ can be specified as a tree where leafs lie in $\mathcal{U}$ and only "♯" among all siblings in the same level d can have children. Based on the setup of Fig. 2, the example DP paradigm in the inset of Fig. 3 has a tree of two flash sets, $\pi = (\mathcal{G}_1, \mathcal{G}_2)$ where

$$\mathcal{G}_1 = \{\text{“}A\text{"}, \text{“}\sharp\text{"}\}, \qquad \mathcal{G}_2 = \{\text{“}B\text{"}, \text{“}C\text{"}\}$$

Note that RC and SC can be viewed as having a policy $\pi = (\mathcal{G}_1 = \bar{\mathcal{U}})$, as shown in Fig. 3 for an example. Run k and depth d evolve on the same time scale. For the RC and SC paradigms, it is always the case that $d = 1$ and $c[k] \in \mathcal{U}$; thus run $k+1$ always pertains to a new round $m+1$. For the DP paradigm, if $c[k] \in \mathcal{U}$ or $c[k] =$ " $\Leftarrow$ ", then at time $k+1$ the depth d becomes 1; otherwise it becomes $d+1$ (see Fig. 1 bottom).

Flash Object and Flash sets. During any run, the detailed illumination of all possible characters in $\mathcal{G}$ depends upon the paradigms RC, SC, and DP. In general, subsets of characters of $\mathcal{G}$, termed **flash objects** $f \subset \mathcal{G}$, are simultaneously illuminated in a random order until a classification is made. For any query subset $\mathcal{G}$, the set of all possible flash objects is called a **flash set** and is denoted by $\mathcal{F}(\mathcal{G})$. As illustrated by the example in Fig. 3, for RC $\mathcal{G} = \bar{\mathcal{U}}$; moreover, flash objects always pertain to either rows or columns

$$\mathcal{F}_{\rm RC}(\bar{\mathcal{U}}) = \{\{\text{“}A\text{"}, \text{“}B\text{"}\}, \{\text{“}C\text{"}, \text{“}\sharp\text{"}\}, \ldots \{\text{“}A\text{"}, \text{“}C\text{"}\}, \{\text{“}B\text{"}, \text{“}\sharp\text{"}\}\}.$$

For both the SC and DP paradigms, flash objects always consist of singleton sets, because no two characters are ever simultaneously illuminated. One small difference between SC and DP, however, is that the symbol "♯" can only be illuminated in DP. For the example in Fig. 3, the SC has

$$\mathcal{F}_{\rm SC}(\bar{\mathcal{U}}) = \{\{\text{“}A\text{"}\}, \{\text{“}B\text{"}\}, \{\text{“}C\text{"}\}\}.$$

while the DP has

$$\mathcal{F}_{\rm DP}(\{\text{“}A\text{"}, \text{“}\sharp\text{"}\}) = \{\{\text{“}A\text{"}\}, \{\text{“}\sharp\text{"}\}\}$$
$$\mathcal{F}_{\rm DP}(\{\text{“}B\text{"}, \text{“}C\text{"}\}) = \{\{\text{“}B\text{"}\}, \{\text{“}C.\text{"}\}\}$$

Trials. Any run consists of N **trials**. Each trial is the random sequence of illuminating each flash object f in a given flash set $\mathcal{F}$. The time between the illumination onsets of two successive flash objects is called stimulus onset asynchrony (SOA). This is illustrated in upper half of Fig. 4.

To be more precise and succinct, the process of sequential spelling under all three paradigms is provided in unified pseudo-code below.

$k = 1, m = 1, d = 1, \sigma[0] = \emptyset$
while $\sigma[m] \neq U^M$ **do**
  randomly illuminate flash set $\mathcal{F}(\mathcal{G}_d)$
  $c[k] \leftarrow$ symbol classified from $\mathcal{G}_d$
  if $c[k] \in \mathcal{U}$ **then**
    $m \leftarrow m + 1$
    $\sigma[m] \leftarrow (\sigma[m-1], c[k])$
    $d \leftarrow 1$
    re-calculate $\mathcal{G}_1, \mathcal{G}_2, \ldots$
  else if $c[k] =$ " $\Leftarrow$ " **then**
    remove last character from $\sigma[m]$
    $d \leftarrow 1$
  else
    $d \leftarrow d + 1$
  end if
  $k \leftarrow k + 1$
end while

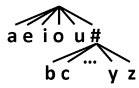

Fig. 5. Tree representation of the policy π pertaining to first illuminating vowels, followed by consonants.

III. Theory for the DP Paradigm

In this section we consider the DP paradigm where variable-sized flash sets can be adaptively chosen. We also consider the following idealized assumption:

Assumption 1:

a) Classification is 100% accurate.
b) $\mathcal{C} = \{\text{“}\sharp\text{”}\}$.
c) $N = 1$.

Note that Assumption 1b and 1c follow naturally as a consequence of Assumption 1a. This leads to the following proposition.

Proposition 1: For each round m, $\sigma[m] = (U_1, \ldots, U_m)$. During run k with associated query subset $\mathcal{G}$, the time taken to spell $c[k]$ is proportional to the size of the flash set, $|\mathcal{F}(\mathcal{G})|$.

Suppose the language, denoted by $\mathcal{U}$, is composed of the 26 characters of the English alphabet and presented by a 5×6 display matrix. Denote $U_m \in \mathcal{U}$ as the mth character in the sentence the subject would like to spell. If all characters in $\mathcal{U}$ were illuminated, the time required to successfully classify U_m is 11 time units for RC and 26 for SC (given the implementation in Fig. 8 of Section IV-B, ignoring any symbols other than the $\bar{\mathcal{U}}$ specified here), assuming each flash set is illuminated once. Suppose $U_1 =$ “t" and $U_2 =$ “h". Consider first illuminating a flash set that only has the characters {“a", “e", “i", “o", “u", “$\sharp$"}, where “$\sharp$” is focused on by the subject if U_3 is one of the 21 consonants. If the subject indeed was interested in a vowel, then U_3 is classified, only $W(1) = 6$ time units were required, and the system can continue to query about U_4. If U_3 is a consonant, then the subject focuses on “$\sharp$” and then after classification a subsequent flash set must be illuminated. Suppose the second flash displays the $W(2) = 21$ consonants and then U_3 is classified. In this case, the total time required to convey U_3 was $W(1) + W(2) = 6 + 21 = 27$ time units. Given $U_1 =$ “t" and $U_2 =$ “h", the average amount of time required is given by

$$6 \times \mathbb{P}(U_3 \text{ is a vowel } | U_1 = \text{“}t\text{"}, U_2 = \text{“}h\text{"}) + 27 \times \mathbb{P}(U_3 \text{ is a consonant } | U_1 = \text{“}t\text{"}, U_2 = \text{“}h\text{"}).$$

If this value is less than $|\mathcal{F}_{\mathrm{RC}}| = 11$ or $|\mathcal{F}_{\mathrm{SC}}| = 26$, then this is a more advantageous approach on average.

Note that any policy π (refer to Section II) is in one-to-one correspondence with a tree of flash sets on $\bar{\mathcal{U}}$, where at each depth, only one node (“$\sharp$”) has descendants. See Fig. 5 for the example with vowels and consonants. The $W_\pi(1)$ nodes at depth 1 in the tree, including “$\sharp$”, are the symbols that are illuminated in the first flash set. “$\sharp$” has descendants that correspond to the set of $W_\pi(2)$ characters that are illuminated in the second flash set if the first classification is “$\sharp$”. This extends until the final depth of the tree. Denote $\Pi(\mathcal{U})$ as the space of all possible trees on $\bar{\mathcal{U}}$ for which one node at any depth has descendants. The time required to classify $u \in \bar{\mathcal{U}}$ for policy $\pi \in \Pi(\mathcal{U})$ is denoted by $L(\pi, u)$

$$L(\pi, u) = \sum_{d=1}^{D_\pi(u)} W_\pi(d) \tag{1}$$

where $D_\pi(u)$ is the depth in the tree that symbol u resides and $W_\pi(d)$ is the total number of symbols in the dth flash set. For the example above, $|\mathcal{U}| = 26$, $d = 1$ pertains to vowels and the # character with $W_\pi(1) = 6$, and $d = 2$ pertains to consonants with $W_\pi(2) = 21$.

Denote the space of all probability distributions on $\mathcal{U}$ as $\mathcal{P}(\mathcal{U}) = [Q : \mathcal{U} \to [0,1] s.t. \sum_{u \in \mathcal{U}} Q(u) = 1]$. We now define the average time to spell a character with distribution Q under policy π as $\bar{L}(\pi, Q)$ and the optimal policy as $\pi^*(Q)$

$$\bar{L}(\pi, Q) \triangleq \sum_{u \in \mathcal{U}} Q(u) L(\pi, u) \tag{2}$$

$$\pi^*(Q) \triangleq \arg\min_{\pi \in \Pi(\mathcal{U})} \bar{L}(\pi, Q). \tag{3}$$

Assuming a statistical model for language, for each U_m we define

$$Q_m(u) = \mathbb{P}(U_m = u | U^{m-1}). \tag{4}$$

Then the minimal expected time to convey M characters $U^M \triangleq (U_1, \ldots, U_M)$ is given by

$$\min_{\pi_1, \ldots, \pi_M} \sum_{m=1}^{M} \mathbb{E}[L(\pi_m, U_m)] = \min_{\pi_1, \ldots, \pi_M} \sum_{m=1}^{M} \mathbb{E}\left[\bar{L}(\pi_m, Q_m)\right] \tag{5}$$

$$= \sum_{m=1}^{M} \mathbb{E}\left[\min_{\pi \in \Pi} \bar{L}(\pi, Q_m)\right] \tag{6}$$

$$= \sum_{m=1}^{M} \mathbb{E}\left[\bar{L}(\pi^*(Q_m), Q_m)\right] \tag{7}$$

where (5) follows from the law of iterated expectation; (6) follows from 1) the statistics of $(U_1, \ldots, U_M)$ are unaffected by any policy π, and 2) Assumption 1a which implies that U^{m-1}, and thus Q_m, is known at the beginning of round m; and (7) follows from (2) and (3). A consequence of (7) is that the batch optimization problem for minimizing expected total time can be done sequentially given any algorithm that solves (3) for any $Q \in \mathcal{P}(\mathcal{U})$. This leads to the algorithm of our adaptive DP paradigm that performs (5) sequentially

for $m = 1$ to M **do**
  Construct $Q_m \in \mathcal{P}(\mathcal{U})$ using (4).
  Find $\pi_m = \pi^*(Q_m) = (\mathcal{G}_1, \ldots, \mathcal{G}_D)$ by (3).
  $d \leftarrow 1$
  while Classifier output is “$\sharp$” **do**
    Illuminate $\mathcal{F}_{\mathrm{DP}}(\mathcal{G}_d)$.
    Find the output of the classifier.
    $d \leftarrow d + 1$.
  end while
  Choose $U_m \in \mathcal{U}$ as the classifier output.
end for

The statistical language model for Q_m is a variable-order Markov model [14] that will be discussed in more detail in Section IV. Below, we discuss how (3) can be solved with dynamic programming.

A. Finding an Optimal Policy for the Variable-Sized Flash set Paradigm Using Dynamic Programming

We now demonstrate that solving (3) for $U \in \mathcal{U}$ with distribution $Q \in \mathcal{P}(\mathcal{U})$ can be cast as a stochastic shortest path problem, which falls within the realm of dynamic programming [10]. We define the control problem as follows.

- $\mathcal{S} = 2^{\mathcal{U}}$. $s_d \in \mathcal{S}$ denotes the "state" of the system, which is the set of characters not yet illuminated. Initially, $s_1 = \mathcal{U}$.
- The action space $\mathcal{A}(s)$ is the set of allowable actions when the system is in s. An action $a \in \mathcal{A}(s)$ is a subset of characters for the flash set that have not been displayed yet

$$\mathcal{A}(s) = \begin{cases} \emptyset, & s = \emptyset \\ 2^s \setminus \emptyset, & s \neq \emptyset \end{cases}.$$

- A policy π maps states to actions. $a = \pi(s)$ specifies what set of symbols in $\mathcal{U}$ to illuminate in the current flash set symbols in s have not been illuminated yet. Π is the set of all possible mappings; note that each π pertains to a tree representation (see the above discussion and Fig. 5).
- Under Assumption 1a, $s_{d+1} \subset s_d$ and the system reaches a termination state $\emptyset$. The state update equation can be succinctly described as

$$\begin{aligned} s_{d+1} &= f(s_d, a_d, u) \\ &= \begin{cases} \emptyset, & u \in a_d \text{ or } s_d = \emptyset \\ s_d \setminus a_d, & \text{otherwise.} \end{cases} \end{aligned} \tag{8}$$

- $g(s, a)$ denotes the cost incurred while in state s and taking action a. The time to classify a character is proportional to the current flash set size—so we have that

$$g(s,a) = \begin{cases} |a| + 1, & s \neq a \\ |a|, & s = a \end{cases} \tag{9}$$

  where the additional 1 in (9) pertains to ♯—which is not in $\mathcal{S}$ or $\mathcal{A}$ but adds to the time to illuminate—except when in the final flash set.

Note clearly from (8) and (9) that

$$L(\pi, u) = \sum_{d \geq 1} g\left(s_d, \pi(s_d)\right) \tag{10}$$

where $S_1 = \mathcal{U}$ by definition. $(S_d : d \geq 1)$ is a controlled Markov chain with the following transition law:

$$\begin{aligned} P_{s,s'}(a) &\triangleq \mathbb{P}(S_{d+1} = s' | S_d = s, A_d = a) \\ &= \begin{cases} \frac{\mathbb{P}(U \in s')}{\mathbb{P}(U \in s)}, & s \neq \emptyset, s' = s \setminus a \\ \frac{\mathbb{P}(U \in a)}{\mathbb{P}(U \in s)}, & s \neq \emptyset, s' = \emptyset \\ 1, & s = \emptyset, s' = \emptyset \\ 0, & \text{otherwise} \end{cases} \end{aligned} \tag{11}$$

where (11) follows from (8). Note that for any policy $\pi \in \Pi$, $J_\pi(\emptyset) = 0$. For any $s \neq \emptyset$, we have that [10]

$$\begin{aligned} J_\pi(s) &= g\left(s, \pi(s)\right) + \sum_{s' \in \mathcal{S}} P_{s,s'}\left(\pi(s)\right) J_\pi(s') \\ &= g\left(s, \pi(s)\right) + P_{s, s \setminus \pi(s)}\left(\pi(s)\right) J_\pi\left(s \setminus \pi(s)\right) \end{aligned} \tag{12}$$

where (12) follows from (11) since $\bar{L}(\pi, Q) = J_\pi(\mathcal{U})$.

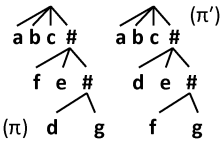

Fig. 6. Representing policy π (left) and π' (right) as Huffman trees.

Finding the optimal policy is a stochastic shortest path problem which can be solved using dynamic programming [10, p. 365]. The optimal cost function $(J_0(s) : s \in \mathcal{S} \setminus \{\emptyset\})$ satisfies Bellman's equation

$$\begin{aligned} J^*(s) &= \min_{a \in \mathcal{A}(s)} \left[g(s,a) + \sum_{s' \in \mathcal{S}} P_{s,s'}(a) J^*(s') \right] \\ &= \min_{a \in \mathcal{A}(s)} \left[g(s,a) + P_{s, s \setminus a}(a) J^*(s \setminus a) \right] \end{aligned}$$

and as mentioned above, $\bar{L}(\pi^*(Q), Q) = J^*(\mathcal{U})$. Unfortunately, the state space is of size $2^{|\mathcal{U}|}$ and the action space in general also has exponential size: $|\mathcal{A}(s_d)| = 2^{|s_d|} - 1$. As such, using a standard approach (e.g., policy iteration or value iteration [10]) to solve Bellman's equation requires exponential complexity. We next illustrate how we can reduce this search complexity from exponential to linear by making an argument analogous to the proof of optimality for Huffman coding [15], [16].

B. Connection to Lossless Data Compression and Reduction from Exponential to Linear Complexity

Consider the scenario where $\mathcal{U} = \{\text{“}a\text{”}, \text{“}b\text{”}, \text{“}c\text{”}, \text{“}d\text{”}, \text{“}e\text{”}, \text{“}f\text{”}, \text{“}g\text{”}\}$. Assume without loss of generality that

$$Q(\text{“}a\text{”}) \geq Q(\text{“}b\text{”}) \geq \ldots \geq Q(\text{“}g\text{”}). \tag{13}$$

Consider the policy π which consists of displaying first $\{\text{“}a\text{”}, \text{“}b\text{”}, \text{“}c\text{”}, \sharp\}$, then $\{\text{“}f\text{”}, \text{“}e\text{”}, \text{“}\sharp\text{”}\}$ if the subject selects “♯”, and finally $\{\text{“}d\text{”}, \text{“}g\text{”}\}$ if the subject did not select “f” nor “e.” See Fig. 6 for a description of π as a tree. Then from (1) and (2) we have

$$J_\pi(\mathcal{U}) = \bar{L}(\pi, Q) = \sum_{u \in \mathcal{U}} Q(u) \sum_{d=1}^{D_\pi(u)} W_\pi(d). \tag{14}$$

This new way to look at the cost leads us to a Huffman code-like problem [15]. The main difference is that the size of the alphabet of the output code, the cost incurred at time d

$$W_\pi(d) = g(s_d, a_d) \tag{15}$$

can change at any node in the tree (for example, see left panel of Fig. 6). Moreover, from (1), we see that the cost function for any symbol $u \in \mathcal{U}$ is **not** just a function of the depth of $D_\pi(u)$ in the tree. Nonetheless, we can state the following theorem and still prove with the same Huffman method that the most likely characters appear first.

Theorem 1: For any optimal policy π^* that satisfies (3)

$$If\ Q(u_1) < Q(u_2) \text{ then } D_{\pi^*}(u_1) \geq D_{\pi^*}(u_2). \tag{16}$$

Proof: We use a proof by contradiction. Suppose that the policy π is optimal but there exists a u_1 and a u_2 for which $D_\pi(u_1) < D_\pi(u_2)$ and $Q_\pi(u_1) < Q_\pi(u_2)$. For example, when $\mathcal{U} = \{\text{“}a\text{"}, \text{“}b\text{"}, \text{“}c\text{"}, \text{“}d\text{"}, \text{“}e\text{"}, \text{“}f\text{"}, \text{“}g\text{"}\}$ and (13) holds, see left panel of Fig. 6 and let $u_1 = \text{“}f\text{"}$ and $u_2 = \text{“}d\text{"}$. Construct another policy π' whose tree representation is equivalent to π except for the roles of u_1 and u_2 are reversed (see right panel of Fig. 6). Note that π and π' have the same topology

$$W_\pi(d) = W_{\pi'}(d) = W(d) \text{ for all}, d. \tag{17}$$

As a consequence, for any $u \in \mathcal{U} \setminus \{u_1, u_2\}$

$$\sum_{d=1}^{D_\pi(u)} W(d) = \sum_{d=1}^{D_{\pi'}(u)} W(d). \tag{18}$$

Therefore

$$\begin{aligned}
&\bar{L}(\pi, Q) - \bar{L}(\pi', Q) \\
&= \sum_{u \in \mathcal{U}} Q(u) \left[\sum_{d=1}^{D_\pi(u)} W(d) - \sum_{d=1}^{D_{\pi'}(u)} W(d) \right] && (19) \\
&= \sum_{\{u_1, u_2\}} Q(u) \left[\sum_{d=1}^{D_\pi(u)} W(d) - \sum_{d=1}^{D_{\pi'}(u)} W(d) \right] && (20) \\
&= \underbrace{(Q(u_2) - Q(u_1))}_{>0} \underbrace{\left(\sum_{d=D_\pi(u_1)+1}^{D_\pi(u_2)} W(d) \right)}_{>0} \\
&> 0 && (21)
\end{aligned}$$

where (19) follows from (14); (20) follows from (18); and (21) follows from the assumption that $D_\pi(u_1) < D_\pi(u_2)$ and $Q(u_1) < Q(u_2)$. Then π' has strictly smaller expected cost, and thus this serves as a contradiction. ∎

Assume without loss of generality that

$$Q(u_1) \geq Q(u_2) \geq \ldots \geq Q\left(u_{|\mathcal{U}|}\right).$$

The state space $\mathcal{S}$ can be collapsed to $\tilde{\mathcal{S}}$ without loss of optimality so that $|\tilde{\mathcal{S}}| \propto |\mathcal{U}|$. The dynamic programming problem can now be solved in less than $|\mathcal{U}|$ iterations using the value iteration algorithm [10, p. 373].

C. Motivation for Experiments

A few issues remain unresolved by theoretical discussion, and demand experimental verifications. First of all, in Assumption 1b there are no classification errors and so no “⇐” was considered. In practice, however, due to classification errors, $\mathcal{C}$ need to include “⇐”.

Secondly, due to the sequential nature of the DP paradigm, classification errors could lead to “error propagation,” where subjects go through multiple flash sets to correct for a single error. Our theoretical discussion is unable to predict how the number of trials per run will affect the classification accuracy for RC, SC, and DP.

Thirdly, DP is expected to outperforms RC and SC when a statistically “typical” sentence is spelled. In this regard, the negative logarithm of the probability of a sequence divided by the number of characters in that sequence is denoted by NLL/Char. Typical sentences have an NLL/Char close to that of the language model. When the target sentence is statistically unlikely (large NLL/Char), DP might under-perform RC or SC. This is demonstrated by simulating the performances in Fig. 7. Details of the language model are in Section IV.

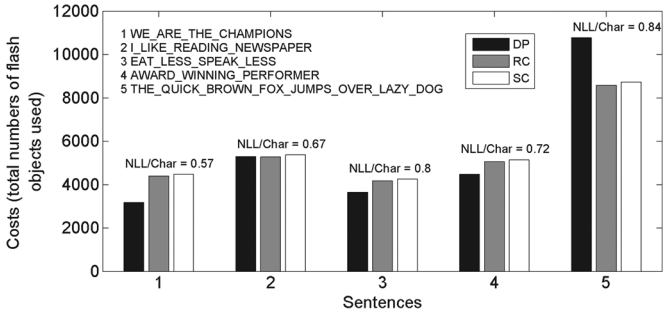

Fig. 7. The performances of the three paradigms were simulated for 5 example sentences, with corresponding NLL/Char labeled above. Costs are measured by the total number of flash objects needed to spell the sentences assuming 100% classification accuracy. DP (black) under-performed RC (gray) and SC (white) when spelling an atypical sentence (number 5).

For these reasons, the theoretically optimal DP speller need not always outperform the RC and SC paradigms. The practical performance of all three may depend on the number of trials N per run in a complicated manner. This motivates the need to perform experiments to determine the effects of these competing factors in terms of aggregate system performance.

IV. Methods

A. Subjects

This study has been approved by the Institutional Review Board of the University of Illinois, Urbana-Champaign. Data were collected from 12 healthy adults (six females) between the age of 18–30. All subjects had normal or corrected-to-normal vision, and had no prior experience with P300-based BCI.

Each of our subjects performed six experiments, of which two were using the DP, two using RC, and two using SC. The sequence of experiencing the three types of paradigm was randomized over all subjects in each gender group, in such a way that each of the six permutations of the paradigm orderings were assigned to one male and one female subject.

B. Visual Displays

In general, all setup is done according to the definitions given in Section II. Fig. 8 shows the same 5×6 display matrix $\mathcal{D} = \bar{\mathcal{U}} \cup \{\text{“}, \text{"}\}$ for all the three paradigms.[1]Note that, in light of how we defined $\mathcal{C}$, we have that

$$\begin{aligned}
\bar{\mathcal{U}}_{\text{RC}} &= \mathcal{U} \cup \mathcal{C}_{\text{RC}} = \mathcal{U} \cup \{\text{“} \Leftarrow \text{"}\} \\
\bar{\mathcal{U}}_{\text{SC}} &= \mathcal{U} \cup \mathcal{C}_{\text{SC}} = \mathcal{U} \cup \{\text{“} \Leftarrow \text{"}\} \\
\bar{\mathcal{U}}_{\text{DP}} &= \mathcal{U} \cup \mathcal{C}_{\text{DP}} = \mathcal{U} \cup \{\text{“} \Leftarrow \text{", “}\sharp\text{"}\}
\end{aligned}$$

The number of trials per run, N, was set to 25 for DP, 20 for RC and 8 for SC. The reasons for empirically choosing these

[1]The comma “,” is merely a placeholder never used.

TABLE I
SPELLING PARADIGM PROPERTIES

Paradigms	Flash Object Description	Flash set Sizes	Flash set Description	Hierarchy Sizes	Number of Trials per Run (N)
RC	Rows and Columns	11	Fixed size containing all characters	1	20
SC	Individual symbols	28	Fixed size containing all characters	1	8
DP	Individual symbols	Variable-sized	A hierarchy of flash sets	>1	25

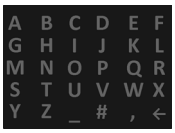

Fig. 8. Display matrix $\mathcal{D}$.

values were: 1) to control the length of the experiments to finish within 10 min, so that the participants do not get too mentally exhausted; 2) to ensure accurate classifications. DP achieved better empirical performance even though it needed more flashes per character than RC or SC, as will be shown in Section V.

Specifically, the flash set for RC, $\mathcal{F}_{\mathrm{RC}}$ contains five rows and six columns as flash objects, and flash set size $|\mathcal{F}_{\mathrm{RC}}| = 11$. The flash set for SC contains 28 individual symbols as flash objects, $\mathcal{F}_{\mathrm{SC}} = \mathcal{D}\backslash\{$“,", “♯"$\}$, and flash set size $|\mathcal{F}_{\mathrm{SC}}| = 28$. The important properties of all three paradigms are summarized in Table I to highlight the uniqueness of DP.

C. Experimental Paradigm

Each experiment was composed of a preparation, a training session and a spelling session.

1) Preparation: The subjects were given instructions about the experiments and were told to sit still and relaxed while trying to avoid blinking during the flash of the target characters. Electrode impedances are reduced to less than $10k\Omega$.

2) Training: The subjects fixated on the training sequence “ATHEOI,” one character at a time. The prompt of target character was given in a pair of parenthesis following the training sequence, displayed on top of the display matrix. The subjects were instructed to how many times the target character was flashed. The duration of flash was set to 100 ms, and the SOA was set to 125 ms for all three types of paradigm. The query subsets in each paradigm is briefly highlighted for 500 ms before each run, and there was a 500 ms pause at the end of each run. At the end of the session, the coefficients of a classifier were estimated by stepwise linear discriminant analysis (SWLDA) [17] with the training data.

3) Spelling and Statistical Language Model: In the spelling session, the subjects were required to spell a predetermined sequence of characters: “HELLO_WORLD” as seen in Fig. 9. The DP programs were blind to this information. Using a variable order hidden Markov model for English language [14], the NLL/Char of this sentence is 0.68, which is typical of this model.

For DP, every flash set was highlighted for 500 ms before each run for subjects to locate the desired character, see Fig. 9 for a series of snapshots of this period. the equivalent periods for RC and SC were blank resting. The classified letter was appended to the feedback sequence and displayed for 500 ms at the end of each run. All other settings were the same as Training. The subjects were told to correct any errors using the backspace “⇐.”

The statistical model of the English language for DP was a variable order Markov model, and the associated prediction by partial matching (PPM) algorithm was implemented in Java [14]. According to Section III and as shown in Fig. 9, DP displayed an optimal flash set each run according to the statistical model of the immediate character it was inquiring, U_m, which is drawn according to $Q_m(\cdot) = P(U_m = \cdot|U^{m-1})$ calculated by the PPM.

D. Data Acquisition

EEG signals were collected by an Electro-Cap through eight channels (Fp1, Fz, Pz, P3, P4, P7, P8, and Oz). Fpz was used as ground, and Cz the reference. Although the standard reference for P300 is the mastoids or earlobe [18], we used Cz instead. Judging from Figs. 12 and 13, classification accuracy and P300 morphologies were not hurt. The subjects monitored their own eye movements (Fp1) to avoid blinking, no further artifact removal techniques were employed. Voltages were amplified by a low noise amplifier (James-Long Co.), low-pass filtered to 100 Hz (analog), and sampled at 400 Hz by an IOtech Personal Daq 3000 A/D converter. A notch filter at 60 Hz was applied in software to eliminate power line noise.

E. Software Setup

The software was adapted from BCI2000 framework Version 3.0 [13]. The visual stimuli were configured to display in a 800 × 600 pixel area on a 21-in LCD screen, 50 cm away in front of the subjects. For DP, the conditional probability distributions over all possible next symbols are calculated by PPM (implemented in Java [14]) and relayed by a Matlab script to the BCI2000 program.

The SWLDA coefficients were fitted at the end of the training session [17], using epochs of 600 ms following the onset of the flashes from all eight channels and down-sampled before concatenating into a single feature vector. During the spelling session, feature vectors time-locked to the flash of each stimulus were generated again, and the dot-product between these vectors and the SWLDA coefficients are computed as the classification scores. The scores from multiple flashes for each stimulus were summed up, and the stimulus with the maximum summed-score was classified as the attended target.

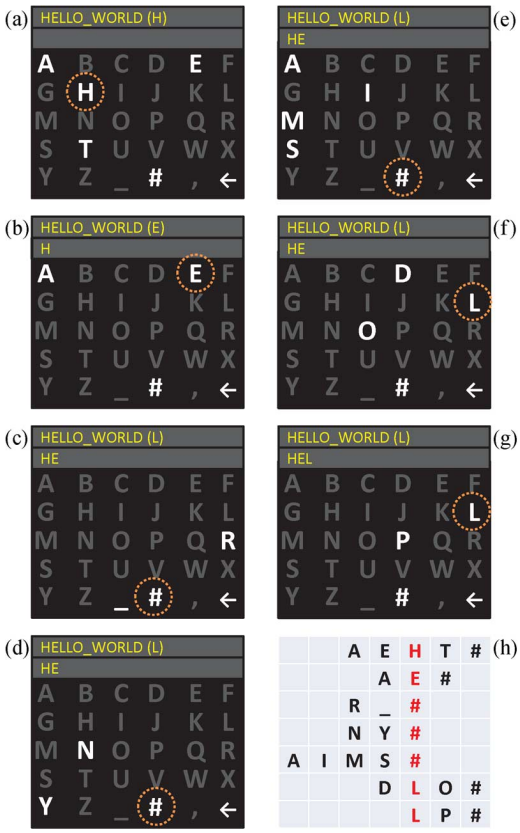

Fig. 9. The flash set of each run are highlighted simultaneously for 500 ms at the beginning of that run, and then they are flashed individually in random sequence. (a)–(g) demonstrate how the flash sets evolve across runs in spelling the first four letters in "HELLO_WORLD". The first text box in each plot shows the target sentence, which ends with a parenthesis indicating the target character of that round. The second text box shows the sequence already spelled out before that run. The dashed circles indicate the symbols that the subject should choose at that run. (h) lists the query subsets of all the runs in (a)–(g), each row corresponding to a run.

F. Statistical Analysis by Bootstrapping

In order to statistically characterize the performance of DP, RC, and SC beyond our specific settings of N, we need a performance metric that is independent of N, therefore we estimated accuracy curves as a function of actual time by bootstrapping analysis [19, p. 436] in accordance with well-established procedures [2], [3].

Random re-sampling with replacement was done 400 times on the feature vectors for each character in "HELLO_WORLD," using only the data from spelling sessions. The fraction of correct classifications from the resamplings is reported as the accuracy.

The average time to spell each actual character was calculated for each paradigm using 22, although it was done a bit differently for DP

$$\begin{aligned} T_{\mathrm{RC}} &= N \times 11 \times \mathrm{SOA} + 1000 \\ T_{\mathrm{SC}} &= N \times 28 \times \mathrm{SOA} + 1000 \\ T_{\mathrm{DP}} &= \frac{16}{11} \times (N \times 4.375 \times \mathrm{SOA} + 1000). \end{aligned} \tag{22}$$

Trials that corresponded to choosing the symbol "♯" were also included in the resampling. However, "♯" was an extra step unique to DP, it was not considered as a successful spelling of an actual character but only added to the cost. Moreover, a 1000 ms pause was also added between characters.

The analysis of average time per character was equivalent to that of the overall time for the whole sentence.[2]

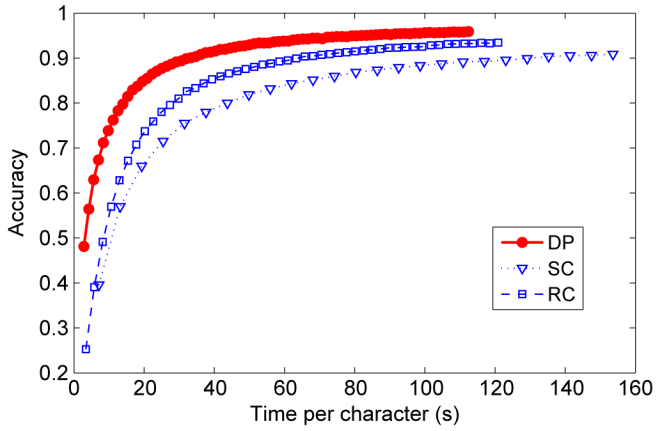

Fig. 10. Accuracy estimated by bootstrapping as a function of the time needed to spell each character, averaged across all subjects. DP (dot) is at least 50% faster than RC (square) or SC (triangle) at 85% accuracy.

V. Experimental Results

A. Spelling Performance Using Bootstrapping

For all paradigms, the spelling speed were obtained from the accuracy curves plotted as a function of the actual time consumed per character, including pause between selections, in Fig. 10. The details of bootstrap analysis were discussed in Section IV. It can be observed from Fig. 10 that, at 85% accuracy threshold, the DP paradigm (dots) has enabled far superior performance (20 s/char) than RC (squares, 39 s/char, p-value= 0.009) and SC (triangles, 68 s/char, p-value= 0.259).

The information transfer rate (ITR) is another popular criterion for performance [20]. ITR is defined as $R = B/T_{\mathrm{sym}}$, where $B = \log C + P \log P + (1-P)\log(1 - P/C - 1)$ for an alphabet size of $C = 27$, and the time per character T_{sym} (min/char) at 85% accuracy threshold is obtained from Fig. 10, i.e., $P = 0.85$. The values of T_{sym} for DP, RC, and SC are 20, 39, and 68 s, respectively (including the time for selecting error-correcting actions and the pause between selections). Therefore, DP achieved 11.3 bits/min (bpm), RC 5.8 bpm, and SC 3.3 bpm. This amounts to an improvement of at least 50% for DP over other conventional approaches.

B. Spelling Performance Using Empirical Data

The best empirical time to completion out of the two experiments done by each subject has been plotted for every type of paradigm in Fig. 11. The mean completion time (± standard deviation), averaged across all subjects, was 6.93 (±0.50) minutes using DP, compared with 9.30 (±0.55) minutes using

[2]To spell a sentence with m characters using N trials per run, DP needs to use $k(\geq m)$ total runs; the flash sets pertaining to runs will be of size $|\mathcal{F}_{\mathrm{DP},1}|, \ldots, |\mathcal{F}_{\mathrm{DP},k}|$. RC needs m flash sets of size $|\mathcal{F}_{\mathrm{RC}}|$, while SC needs m flash sets of size $|\mathcal{F}_{\mathrm{SC}}|$. The total/averaged costs (total/averaged number of flash objects needed) are obviously only different by a constant factor $1/m$: $\mathrm{TOTAL}_{\mathrm{DP}} = (|\mathcal{F}_{\mathrm{DP},1}| + \ldots + |\mathcal{F}_{\mathrm{DP},k}|)N, \mathrm{AVG}_{\mathrm{DP}} = (|\mathcal{F}_{\mathrm{DP},1}| + \ldots + |\mathcal{F}_{\mathrm{DP},k}|)N/m; \mathrm{TOTAL}_{\mathrm{RC}} = |\mathcal{F}_{\mathrm{RC}}|mN, \mathrm{AVG}_{\mathrm{RC}} = |\mathcal{F}_{\mathrm{RC}}|mN/m; \mathrm{TOTAL}_{\mathrm{SC}} = |\mathcal{F}_{\mathrm{SC}}|mN, \mathrm{AVG}_{\mathrm{SC}} = |\mathcal{F}_{\mathrm{SC}}|mN/m.$

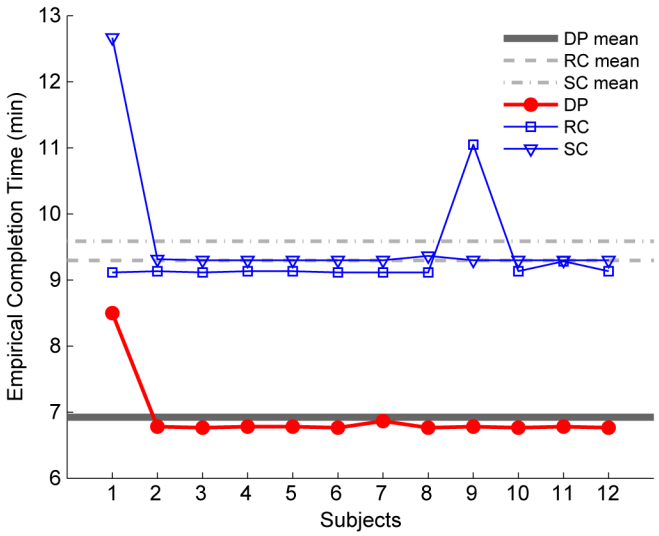

Fig. 11. The empirical time to completion is plotted for every subject. DP (dot) enabled all subjects to spell significantly faster than RC (square) and SC (triangle).

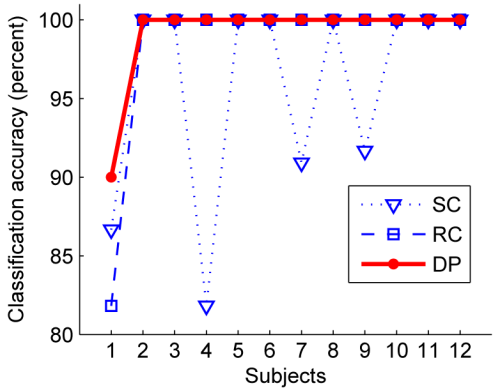

Fig. 12. The empirical accuracy of spelling is plotted for every subject using each of the three paradigms. Although not statistically significant, DP (dot) enabled equal or higher accuracy than RC (square) and SC (triangle) across all subjects.

RC and 9.59 (± 0.97) using SC. Welch's t-test verifies that DP enables the subjects to spell significantly faster than both RC ($t(11) = -11.1, p < 10^{-4}, d.f. = 21.7$) and SC ($t(11) = -8.5, p < 10^{-4}, d.f. = 16.4$). The DP paradigm has reduced the time to completion by about 25.5% over the RC paradigm, and by 27.7% over the SC paradigm. It should be emphasized that even though the number of trials N for DP was unfavorably set to its disadvantage (25) as compared to the both RC (20) and SC (8), it still performed the best in terms of completion time, *uniformly* across individual subjects (Fig. 11).

The DP paradigm also seemed to have helped improving empirical classification accuracy (Fig. 12), which is defined as the ratio between the number of correctly classified target characters and the number of total classifications incurred during spelling. Although Welch's t-test did not show significance over RC ($t(11) = 0.39, p = 0.70, d.f. = 17.1$) or SC ($t(11) = 1.59, p = 0.13, d.f. = 15.2$), note that the DP enabled equal or higher accuracy *uniformly* across all subjects.

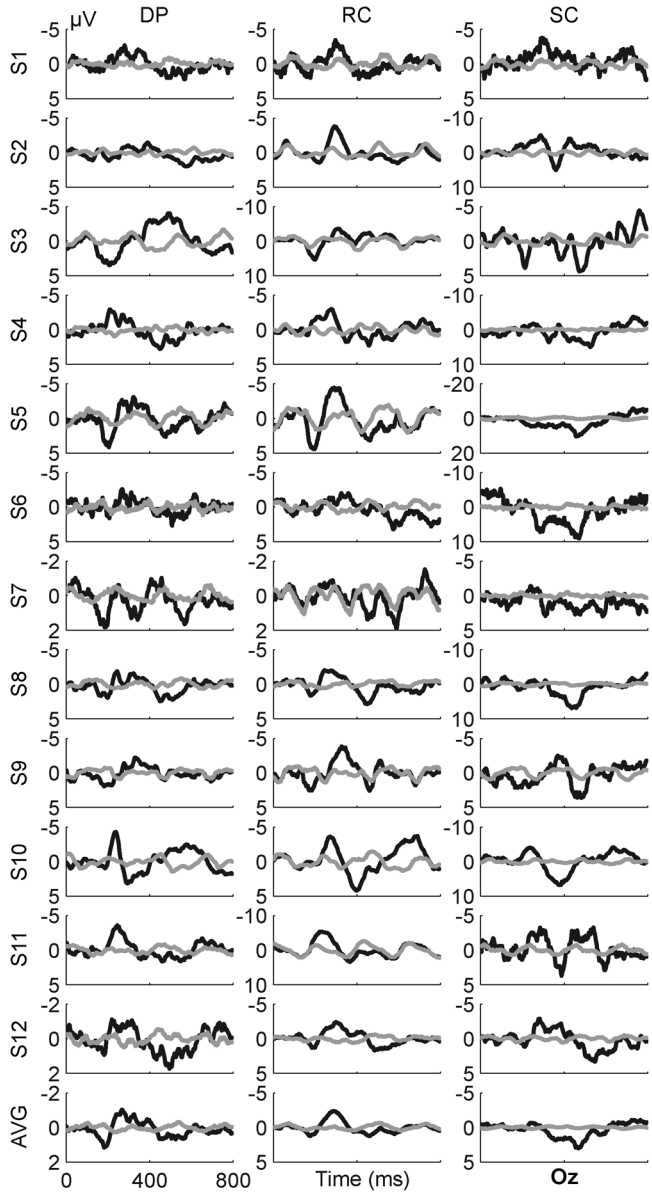

Fig. 13. P300 waveforms recorded from Oz are plotted for all subjects (S1–S12) using each paradigm. The waveform averaged across all subjects (AVG) are plotted at the bottom. Target and null stimuli are in dark and light colors respectively.

C. Physiological Data

Fig. 13 plots the P300 waveforms recorded from Oz for all subjects (S1–S12) using DP, RC, and SC, and the waveforms averaged across subjects (AVG) are at the bottom. All subjects showed clear distinctions between the responses to the targets and the null stimuli. Each subject displayed different waveforms using different paradigms. Overall, the morphologies closely resembled those in other work [21].

VI. Discussion and Future Work

In short, we have developed an efficiently solvable stochastic control approach to design hierarchical flash set strategies that exploit the statistical structure of language. Dynamically varying the illuminated flash set, in accordance with a statistical

model of language, was cast within the framework of a stochastic shortest path problem. In order to efficiently solve the problem, we proved that any optimal scheme displays queries more likely symbols first. This reduced the complexity of the problem from exponential to linear in alphabet size.

In order to compensate for its higher frequency of illuminating the intended letter which weakened P300 responses, we set the number of trials per run for DP to be larger than that of RC and SC [2], [7]. Nonetheless, the empirical time to complete sentences was the smallest for DP *uniformly* across all subjects. The superior performance of DP is due to its ability to reduce the averaged flash set sizes in the long run—not by taking advantage of artificially manipulating N. To elucidate a performance metric that is independent of trials per run N, we performed a bootstrap analysis on experimental data with human subjects that characterized the average time to spell a sentence as a function of the actual completion time per character. According to this metric, the DP speller again out-performed the other fixed-flash set paradigms by over 50% with a 85% classification accuracy.

The method presented is independent of the classification technique, the alphabet for the language, and the statistical model of the language. We also expect considerable increase in performance by using more sophisticated statistical models of language—for example, use Google word-completion.

Other future directions consist of including explicitly the probability of error in the variable-length flash set optimization. In principle, we could apply our technique in conjunction with previously discussed variable-trials-per-run paradigms [9], [11] to elicit further performance improvement.

VII. Conclusion

This paper has exploited the statistical structure of language to design a multiple-flash-set perusal strategy to convey a character in a language using a P300 communication prosthesis. Boostrap analysis demonstrated that for 85% classification accuracy, average completion time is reduced by at least 50% using this dynamic approach as compared to traditional fixed-flash-set paradigms. It is clear that the design of BCIs can benefit greatly from employing fundamental principles of stochastic control and statistical language models systematically.

Acknowledgment

The authors would like to thank L. Srinivasan for the seminal discussion on P300 spellers with adaptive flash sets; C. Omar and D. Steines for software developments; E. Maclin, M. Johnson and L. Varshney for useful discussions; and K. Snell, S. Liou, R. Frasca, D. Reitz, M. Kasraie, D. Shin, G. R. Gorantla, and S. Umunna for helping with administering experiments.

References

[1] G. Dornhege, *Toward Brain-Computer Interfacing*. Cambridge, U.K.: MIT Press, 2007.

[2] L. Farwell and E. Donchin, "Talking off the top of your head: Toward a mental prosthesis utilizing event-related brain potentials," *Electroencephalogr. Clin. Neurophysiol.*, vol. 70, no. 6, pp. 510–523, 1988.

[3] E. Donchin, K. M. Spencer, and R. Wijesinghe, "The mental prosthesis: Assessing the speed of a P300-based brain-computer interface," *IEEE Trans. Neural Syst. Rehab.*, vol. 8, no. 2, pp. 174–179, Jun. 2000.

[4] S. A. Wills and D. J. C. Mackay, "DASHER-an efficient writing system for brain-computer interfaces," *IEEE Trans. Neural Syst. Rehabil. Eng*, vol. 14, no. 2, pp. 244–246, Jun. 2006.

[5] C. Omar, A. Akce, M. Johnson, T. Bretl, R. Ma, E. Maclin, M. McCormick, and T. P. Coleman, "A feedback information—Theoretic approach to the design of brain-computer interfaces," *Int. J. Human-Computer Interaction*, vol. 27, no. 1, pp. 5–23, Jan. 2011.

[6] D. J. Krusienski, E. W. Sellers, F. Cabestaing, S. Bayoudh, D. J. McFarland, T. M. Vaughan, and J. R. Wolpaw, "A comparison of classification techniques for the P300 speller," *J. Neural Eng.*, vol. 3, pp. 299–305, 2006.

[7] C. Guan, M. Thulasidas, and J. Wu, "High performance P300 speller for brain-computer interface," in *IEEE Int. Workshop Biomed. Circuits Syst.*, Dec. 2004.

[8] C. Guger, S. Daban, E. Sellers, C. Holzner, G. Krausz, R. Carabalona, F. Gramatica, and G. Edlinger, "How many people are able to control a P300-based brain-computer interface (BCI)?," *Neurosci. Lett.*, vol. 462, no. 1, pp. 94–98, 2009.

[9] A. Lenhardt, M. Kaper, and H. J. Ritter, "An adaptive P300-based online brain-computer interface," *IEEE Trans. Neural Syst. Rehab.*, vol. 16, no. 2, pp. 121–130, Apr., 2008.

[10] D. P. Bertsekas, *Dynamic Programming and Optimal Control*. Nashua, NH: Athena Scientific, 1995.

[11] J. Park, K. E. Kim, and S. Jo, "A POMDP approach to P300-based brain-computer interfaces," in *Proc. 14th Int. Conf. Intell. User Interfaces*, 2010, pp. 1–10.

[12] J. Jarzebowski, L. Srinivasan, and T. P. Coleman, "Using stochastic control with data compression perspectives to enhance P300 neural communication prostheses," in *IEEE Inf. Theory Workshop*, Porto, Portugal, 2008, pp. 109–113.

[13] G. Schalk and J. Mellinger, *A Practical Guide to Brain-Computer Interfacing With BCI2000*. New York: Springer, 2010.

[14] R. Begleiter, R. El-Yaniv, and G. Yona, "On prediction using variable order Markov models," *J. Artif. Intell. Res. (JAIR)*, vol. 22, pp. 385–421, 2004.

[15] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proceedings of the IRE*, 1952.

[16] T. M. Cover and J. Thomas, *Elements of Information Theory*, 2nd ed. New York: Wiley, 1991.

[17] N. R. Draper and H. Smith, *Applied Regression Analysis*. New York: Wiley, 1966.

[18] C. C. Duncan, R. J. Barry, J. F. Connolly, C. Fischer, P. T. Michie, R. Näätänen, J. Polich, I. Reinvang, and C. Van Petten, "Eventrelated potentials in clinical research: Guidelines for eliciting, recording, and quantifying mismatch negativity, P300, and N400," *Clin. Neurophysiol.*, vol. 120, no. 11, pp. 1883–1908, 2009.

[19] B. Efron, R. Tibshirani, and R. J. Tibshirani, *An Introduction to the Bootstrap*. London, U.K.: Chapman & Hall/CRC, 1993.

[20] J. R. Wolpaw, N. Birbaumer, W. J. Heetderks, D. J. McFarland, P. H. Peckham, G. Schalk, E. Donchin, L. A. Quatrano, C. J. Robinson, and T. M. Vaughan, "Brain-computer interface technology: A review of the first international meeting," *IEEE Trans. Rehabil. Eng.*, vol. 8, no. 2, pp. 164–173, Jun. 2000.

[21] D. B. Ryan, G. E. Frye, G. Townsend, D. R. Berry, S. Mesa-G, N. A. Gates, and E. W. Sellers, "Predictive spelling with a P300-based brain-computer interface: Increasing the rate of communication," *Int. J. Human-Computer Interact.*, vol. 27, no. 1, pp. 69–84, 2010.

Rui Ma received the B.S. degree in neurobiology and biophysics from the University of Science and Technology of China, Hefei, China, in 2004, and the M.S. degree in applied mathematics and the Ph.D. degree in neuroscience, both in 2008, from the University of Illinois Urbana-Champaign. He has been a postdoctoral scholar in Prof. T. Coleman's lab since 2008.

His research interests involve brain–computer interfaces in human and in animals, as well as flexible electronics.

Navid Aghasadeghi graduated with a B.S. in electrical and computer engineering from the University of Texas at Austin in 2008. He obtained his M.S. in electrical and computer engineering in 2010 from the University of Illinois at Urbana-Champaign, where he is currently pursuing his Ph.D. under the supervision of Prof. T. Bretl.

His research interests include optimal control, inverse optimal control and random processes applied to problems in motor control, and design of brain-machine interfaces

Julian Jarzebowski received the B.S. degree in mathematics and physics from the Ecole Stanislas, in 2004, and the M.S. degree in computer engineering from the École Supérieure d'Électricité, in 2006. Then he joined Prof. T. Coleman's lab in Fall 2006, and received the M.S. degree in electrical and computer engineering, in 2008.

He is currently a Senior Systems Engineer at Motorola Solutions.

Timothy Bretl (M'05) received his B.S. in engineering and B.A. in mathematics from Swarthmore College in 1999, and his M.S. in 2000 and Ph.D. in 2005 both in aeronautics and astronautics from Stanford University. Subsequently, he was a Postdoctoral Fellow in the Department of Computer Science, also at Stanford University.

Since 2006, he has been with the University of Illinois at Urbana-Champaign, where he is an Assistant Professor of Aerospace Engineering and an Affiliate of the Neuroscience Program.

Dr. Bretl was a recipient of the National Science Foundation Faculty Early Career Development Award, in 2010.

Todd P. Coleman (SM'11) received the B.S. degree in electrical engineering and computer engineering from the University of Michigan, Ann Arbor, in 2000, along with the M.S. and Ph.D. degrees in electrical engineering from the Massachusetts Institute of Technology (MIT), Cambridge, in 2002 and 2005. He was a postdoctoral scholar at MIT and Massachusetts General Hospital during 2005–2006.

From July 2006 to June 2011, he was an Assistant Professor of Electrical and Computer Engineering and Neuroscience at the University of Illinois, Urbana-Champaign. Since July 2011, he is an Associate Professor in the department of Bioengineering at the University of California, San Diego.