WORKING NOTE D27

Alan Creak
2000 October 13

A BINARY SELECTION INTERFACE

This note records a sequence of suggestions stimulated by a paper by Perelmouter and Birbaumer (□P&B□) on a sequential binary selection interface[1]. A Huffman-algorithm design is proposed, guaranteeing the minimum-length selection sequence for each letter consistent with a maximally efficient set of letter codes. (□THINKS□ haven't they just reinvented Morse code□□) An interesting extension caters for errors in selection; an additional "letter" is included to delete the previous letter, and the position of this letter in the selection tree is determined according to the probability of single errors, moving closer to the rot as the error probability increases. At each step of the selection, two groups of letters are presented, and the correct group must be selected. I suppose that's selection, but the selection objects change at each step. They do not describe in any detail how the letters are presented, though some other references are given.

There is some information about the interface. In the main article[1], they write□ "... at each writing step ... letters, which are attainable from this ... node by the rejection response ..., can be presented to the subject in the upper goal. A set of letters, which are attainable by the selection response ..., can be presented in the opposite goal.". In another article[2], apparently about the same device, they write□ "... the alphabet is split into halves (□etter-banks□) which are presented successively at the bottom of the screen for several seconds. If the subject selects the letter-bank being shown ... the letter-bank is then split into two new halves, and so on ...".

I think that means that half the available letters are presented at any time, and to choose that half you must actively select it. If you don't, you see the other half, which you must actively select. Presumably the two halves alternate until you select one. The principle there isn't silly, because the method is intended for use by people who have great difficulty in producing any signal whatever, and what they can produce is noisy. The practice seems improvable□ wouldn't it be better to show *all* the letters each time, in alphabetical order, with those selected distinguished in some way□□ then you'd know more or less where to look for the letter of interest, and wouldn't have to search curious sets of letters to the end to determine that your letter wasn't there. Also, once you'd found the letter you wouldn't have to search at all□ you could just watch it change state.

This bare description of the interface struck me as fairly dreadful, though I have to make it clear that I haven't followed up their other references. I thought I might be able to do better, and that set me on an interesting track.

WHAT'S WRONG WITH PP&B□

I use "PP&B", standing for "perceived P&B", to denote my imagined version of P&B's interface. It conforms to the description above, and has no further refinements. There are two obvious features which are likely to be unhelpful□

1□ Much searching□ At each step you have to find the letter you want among others in a set. Doubtless you'll learn the sets before very long, so this might not be a serious problem, but it's there. It also seems possible that if the character sets suddenly change after each selection you have to start from scratch every time;

2□ Binary mistake irrevocable□ You can only cancel complete characters□ you can't go back a step if you notice you've made a mistake part way through the selection. This is inevitable given the binary signals, but could be fairly frustrating.

On the other hand, making common characters such as E and T easy to encode is clearly sensible, so there's something in the principle.

(□ accept that some of the force of those comments is lost in the context of the area of application, which is that of people who communicate by brain-computer interfaces, which are by and large slow and error-prone, and where one is required to make some response every few seconds or so, which itself can increase the error rate. I'm not being rude about P&B's methods□ for the people who use such interfaces, any communication at all is to be welcomed.□)

HOW CAN WE DO BETTER?

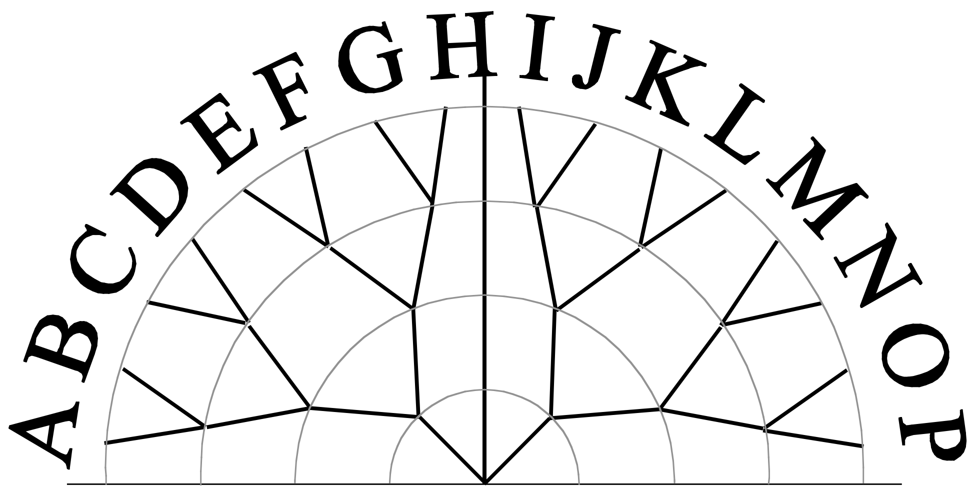
I have directed my attention to, first, finding the character you want, then keeping track of it, and then distinguishing between the sets selected by the next action.

I began with the notion of spreading out characters in alphabetical order (so you know roughly where you're aiming) and then using the binary choice principle, in effect, to drive to the letter you want through the tree. (There is also the possibility of starting with a special step which selects different trees for numbers, punctuation, etc., but either that has to be a different action or it has itself to be fitted in to the scheme in some optimal way. I haven't pursued this line very far.) I've considered possible uses of animated graphics to give the impression of approaching the letters as selection proceeds. The advantage of such a topographical approach is that it you make a mistake then there's some sense in which you're "close" to where you wanted to be, and some possibility for devising ways to get back to the track you want, so you have some room for correcting errors.

The next step – how to use the ideas – is not so much less obvious as full of possibilities. Here are some possibilities.

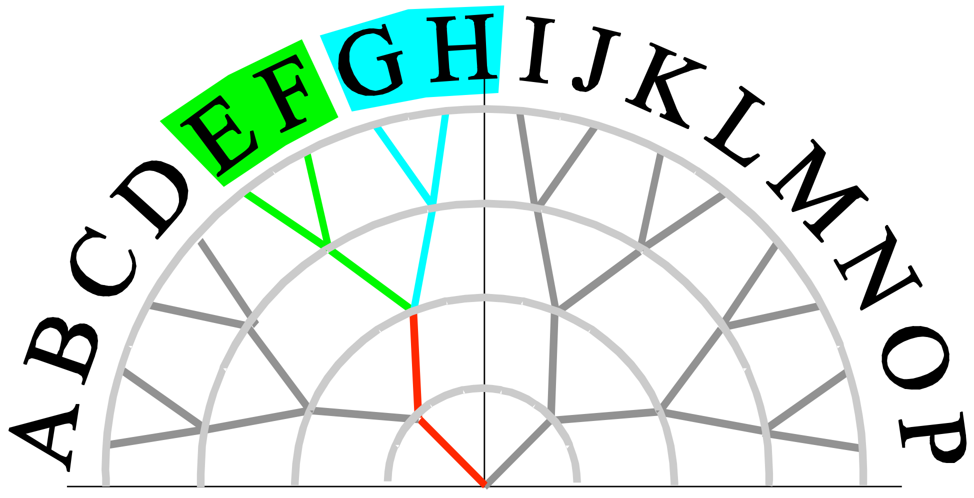
SIMPLE BINARY TREE.

Here's a road map:

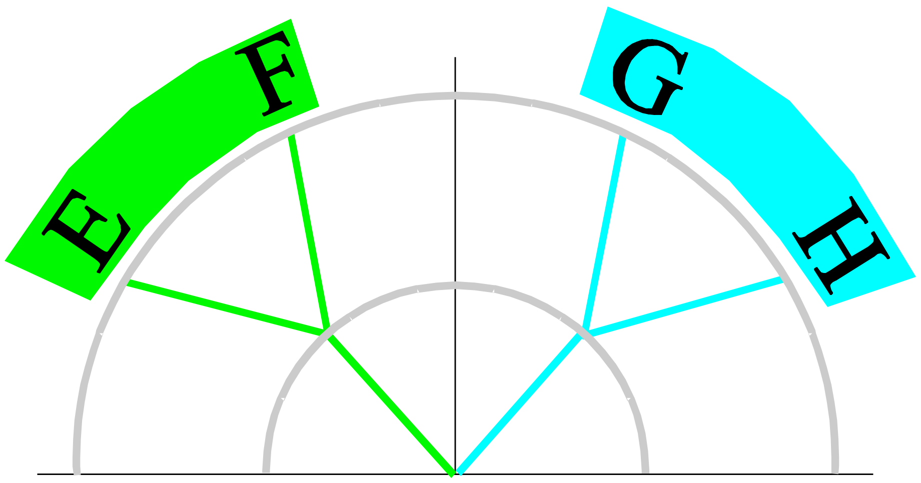


You start at the bottom, and drive along the roads to the letter you want. At each junction you choose "yes" or "no" to turn left or right. If you can manage a sort of "very yes" and "very no", you could go a bit further left or right, which might be a sort of error correction.

Additional decorations can be used to remind you where you are; I think this diagram is pretty well self-explanatory:



On a reasonably big screen, that would probably do. An alternative might be to animate the action, smoothly spreading out and centring the remaining significant letters as the selection proceeds, while letting the others slide off the edges of the diagram□



The smooth change is important; the idea is that you can keep your attention on the letter you've chosen without any sudden jumps□ so, once found, you don't have to find it again. If selections were made quickly enough, the change could be made continuous, which would strengthen the image of driving. If a selection was late, a smooth reduction in "speed" could be used to suspend the "journey".

Once you've reached a letter, the next road map should become visible□ indeed, it could even be slightly anticipated, which would improve continuity. I think (□r, more precisely, guess□ that would be better than "bursting through" a letter to find the next map behind it, but there are perhaps psychological factors here which I don't know about.

I am not entirely convinced by that suggested implementation□ it doesn't seem to add a lot□ but there it is. It could help by keeping the choice boundary more or less vertical, which might help people to see what's happening, but I'm not sure about that.

That's all about navigating through the tree, but so far I haven't addressed the optimised unbalanced trees.

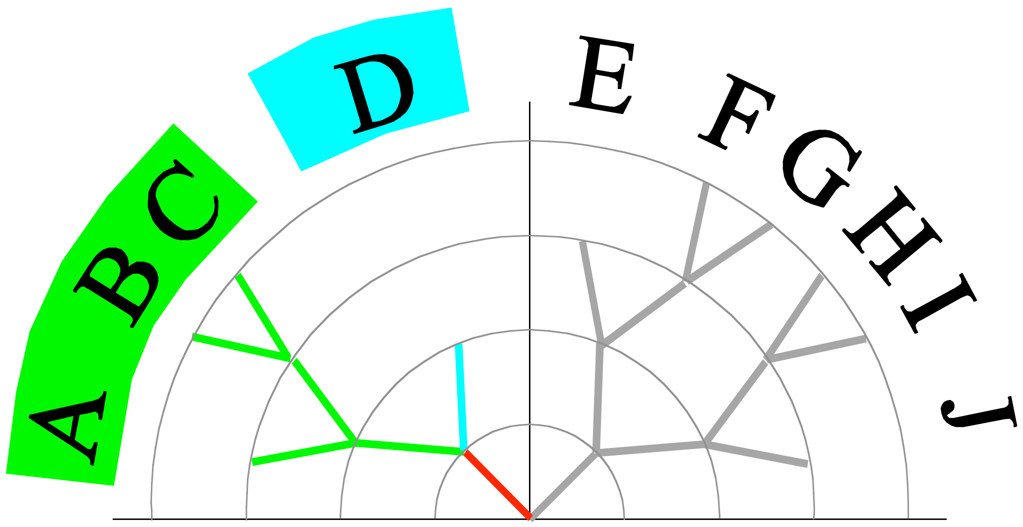
UNBALANCED TREES.

This is what you want to approach P&B's method. The letter targets occur at different levels in the tree, and it takes longer to reach some than to reach others. The difference depends on the frequency of use; E is close to the root, X is further away.

If you want to get maximum information-theory efficiency, you order the characters as in P&B's analysis□ then you lose the alphabetical order cue, but it wouldn't be dreadfully hard to learn where things are. (□Compare typewriter keyboard, Morse, etc.□

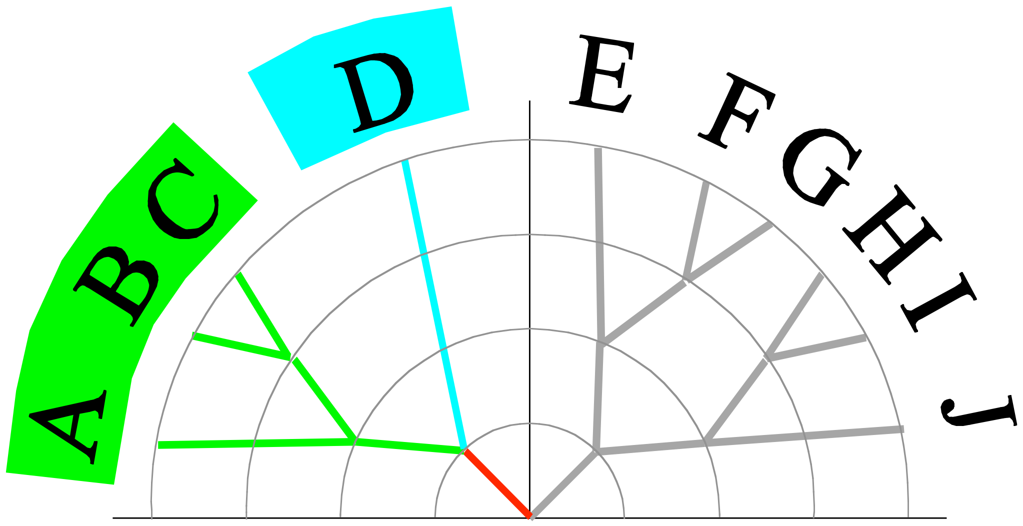
An alternative is to keep the alphabetical order, which is simpler for beginners, and modify the treatment. I originally guessed that it wouldn't make much difference, as the most common letters wouldn't be significantly affected, but that's certainly oversimplified and equally certainly wrong in extreme cases; for example, however popular letter E might be, you could never reach it with just one selection because you have to preserve some route to letters on both sides. It would be interesting to probe the various issues involved.

Here I don't consider the order of the letters, nor do I use realistic frequencies; I'm concerned with the selection process, so the letters which appear on the diagrams are purely for illustration. Here's an example of a simple unbalanced tree as one might construct it from the balanced tree examples with minimal alteration.



Clearly, the same methods can be used.

I have left the letters around the circumference rather than moving them inwards where possible because I think that the orderly presentation is more important than the distance from the origin; it might be a little clearer if the short lines are extended to the circumference



– but in that diagram it is less clear that only two selections are necessary to reach D, a fact which might be helpful in anticipating the next character. Also, the extension of the lines does obscure the relationship between routes which is significant for error correction, if there is any, as the error correction to the left for the turquoise D route can get you only to the end of the first step of the green branch.

WHAT ABOUT ERROR CORRECTION

Incorporating P&B's error signal is straightforward it appears as one more character, and is dealt with in the same way as any other character.

My suggestion of a sideways shift within the pattern isn't so easy. I suggested a "very yes", for example, but encoding the "very yes" is a problem. It must require some bandwidth, and if there's only one channel we're probably already using the bandwidth to the maximum by pushing the selection speed up as far as possible. Notice that in fact it would be necessary to double the information rate, for to make it work one must be able to select one from four possibilities at each point instead of one from two.

A rather less expensive, and in some ways rather more effective, error-correction method would be to have a single additional signal, meaning "reverse the previous selection". This really does reverse the previous choice, and therefore gives access to the whole of the other tree, while the "very yes" method only reaches the nearer part of the other tree; the cost is that it's slower, as you don't move along the tree at all.

If a second communication channel were available, this could in principle be used for error control. This might not be the optimal way to use the channel, though. The error control signal must be presented about as fast as the ordinary navigation signal, so a channel of comparable bandwidth would be required. If such a channel were available, it might be much better used in augmenting the primary navigation control.

I don't know the answer to this question.

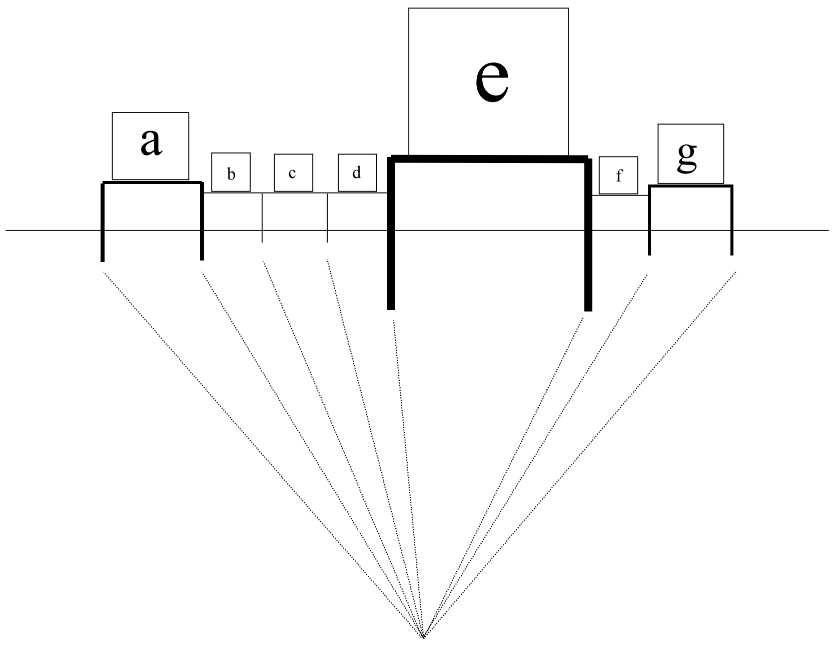
There is one intriguing possibility, connected with the original context of P&B's work, which was brain-computer interfaces. They use the slow cortical potentials of the EEG to convey signals[2]. People can learn, by a biofeedback technique, to control these potentials, and there is no obvious way to provide alternative channels. However, in another interface based on event-related potentials[3], subjects produced signals by engaging in, or even just thinking about, various actions of different parts of the body; in effect, specific thoughts are detected. Is it possible to detect the thought roughly expressed by "Oh, NO□" which everyone has on noticing that one has just made a mistake□ That's an independent channel which costs essentially nothing, and if it could be detected and reliably characterised could be a valuable error signal.

A DIFFERENT WAY□

I've used the analogy of driving from the origin to a destination letter along a network of roads representing the binary tree. Following this analogy, the "very yes" represents a cross-country journey not marked on the map□ which is sensible enough, as it's an exceptional operation, and if you're driving with a binary switch it makes a lot of sense.

But if you extend the driving analogy to a steering wheel, other possibilities become apparent. The roads become redundant, as the steering is all the control you need, and the question of error correction doesn't quite disappear but is absorbed into the related question of how fast you should drive.

The rest of the principle remains much the same. In this case, it is probably appropriate to bring the big letters "closer" to exploit the higher probabilities without artificially, and inexplicably, changing the apparent speed. I think a natural metaphor would be that of driving along a motorway and choosing directions (□probably not amounting to marked lanes, but it's a thought□ according to motorway-like notices suspended above the "road". This could give a much smoother transition from letter to letter. It might look something like this□



(□ but not very like that. The sizes and distances of the gates□ and particularly the widths of the lanes□ should reflect the probabilities of the letters, but a circular, rather than flat, effect would be better.□

One would hope to adjust the "distances" so that the time available for driving there was properly related to the probability of the letter's appearance. The meaning of "properly" remains to be determined.

As one approaches the gates, the next set of gates should appear in the distance. This raises an interesting question. Clearly, the next gates should be straight ahead when you pass the earlier gates wherever that might be□ so if you change direction after the next gates have appeared they should apparently move. This is not entirely satisfactory. Would it be better to regard the controls as moving the gates over the road rather than steering the vehicle beneath a required gate□

BOTH WAYS□

I think it possible that both the road map and the free steering methods might have their uses. The map is better adapted to someone who does have very slow communication or not very precise control, while the steering suits someone with reasonably good fine motion control. I'm not qualified to say much more about this question.

REFERENCES.

1□ J. Perelmouter, N. Birbaumer□ "A binary spelling interface with random errors", *IEEE Trans RE* **8**, 227-232 (□000□

2□ N. Birbaumer, A. Kübler, N. Ghanayim, T. Hinterberger, J. Perelmouter, J. Kaiser, I. Iversen, B. Kotchoubey, N. Neumann, H. Flor□ "The Thought Translation Device (□TTD□ for completely paralyzed patients", *IEEE Trans. Rehab. Eng.* **8**, 190-193 (□000□

3□ S.P. Levine, J.E. Huggins, S.L. BeMent, R.K. Kushwaha, L.A. Schuh, M.M. Rohde, E.A. Passaro, D.A. Ross, K.V. Elisevich, B.J. Smith□ "A direct brain interface based on event-related potentials", *IEEE Trans. Rehab. Eng.* **8**, 180-184 (□000□