

# Participating in Cognition: The Interactive Search Optimization Algorithm

Nadav Abkasis, Israel Gottlieb and Eliraz Itzhaki

*Department of Computer Science, Jerusalem College of Technology, Israel*

**Abstract.** We consider the Distributed Cognition paradigm as a framework for implementing artificial components of human cognition. We take email/internet search as a setting of distributed cognition and describe an algorithm that intervenes and enhances a component of this distributed process. Results are presented demonstrating the effectiveness of the algorithm in interaction with an automaton simulating human search activity. We define a notion of *synchronization* with a non-random, non-Turing computation and argue that the given algorithm exhibits such synchronization behavior. Our results suggest that the framework may be useful for studying a class of non-Turing computation that is central to General Intelligence.

**Keywords.** Distributed Cognition, Embodiment, Search Optimization, Non-Turing Computation.

## 1. Introduction

In Artificial General Intelligence one seeks the "General", i.e. the ability to come up with the right approach to a new situation, as opposed to merely applying an algorithm to a problem setting for which it has been designed. The phrase "applying an algorithm to a problem for which it has been designed" is a fair description of an arbitrary but specific Turing machine. A Universal Turing machine relaxes the condition that the algorithm be a fixed one, but leaves intact the requirement that the algorithm executed must be appropriate for the problem to be solved. It is therefore natural to investigate the possibility, and perhaps even the necessity of non-Turing capability for AGI, or what has been called "hypercomputing". Most work on hypercomputing has been of a theoretical sort, i.e. investigation of machine structures whose physical realizability remains a largely open question [1]. In this work we are concerned with a form of hypercomputing that is directly observable, and offers a practical context for experimentation as well.

A researcher confronted with a problem will often have practical experience that suggests an "intuition" about the correct approach to a solution. The challenge in such situations is typically expressed by the phrase, "can you formalize this intuition?" By this is meant can one translate the intuition into explicit rules, hence making them more readily subject to analysis, optimization and automated implementation.

In cognitive science a key issue is the process of intuition itself; here the challenge is modified as – "can we formalize the *algorithm* of intuition?". In both contexts we can observe an attempt to bridge the no-man's-land between the obscurity of intuition

and the clarity of an algorithm. In the cognitive science case however, the attempt has historically been problematic. If we take "algorithm" as synonymous with "Turing machine" and at the same time allow that "intuition" is non Turing-computable – as Turing himself believed [2], then "algorithm" and "intuition" are mutually exclusive and clearly we are asking the wrong question. Or perhaps our notion of algorithm needs to be extended.

In this paper we approach the task of understanding the process of intuition via a "learn by doing" method, i.e. explicit *participation* in the algorithm of intuition. We will argue that such explicit participation amounts to a broader notion of what is meant by an algorithm than the accepted sense of the term, and one that may be more suited to the goals of AGI.

How can we participate in a process that appears to us as a black box locked away in the human brain? Here, the notion of *distributed cognition* [3] is useful. Distributed cognition posits that cognitive processes should be viewed as including elements outside of the physical individual. Hence we may intervene or participate in this process by replacing one of its external components with an alternate version.

In what follows, we first briefly review the relevant tenets of distributed cognition. We then tackle the question of how to render explicit any part of a process which is by definition not well-defined, i.e. intuition. Our answer will be constructive, i.e. we consider a common cognitive process in which intuition plays an obvious role, posit a distributed structure as the setting for this process and then build an enhanced version of one part of this structure.

In this work we do not as yet draw conclusions regarding intuition nor offer a model for this or any other cognitive process. Our purpose is to suggest active participation as a technique for learning about such processes and associated possible models. Moreover, our emphasis is on the construction of working systems that produce results and can be used to learn by experiment. Accordingly, most of this paper describes a working system. Finally, we shall give an informal argument that our methodology exhibits a form of directly observable non-Turing computation.

### 1.1. *Distributed Cognition*

Distributed cognition is described in [3] as positing distribution of the process of cognition over three dimensions,

1. The social dimension, i.e. over groups rather than single individuals,
2. Over the material artifacts in the environment wherein the cognition is operative, and
3. Over the time dimension. That is, historical events influence later ones via a process that is itself integral to the cognition. The reverse may also be true, viz. the influence of history is itself influenced by the present in that it is interpreted differently according to current experience.

An additional tenet of distributed cognition is that of *embodiment*. This amounts to a rejection of the view that cognition is a process consisting of interactions between purely abstract representations, from which it follows that the underlying substrate that carries these representations is of no consequence to cognitive behavior. Rather, it holds that such substrate directly influences the cognitive process and dictates the types of interactional structures that may emerge.

## 1.2. Search Application

Consider an email search task that often confronts a user at a large organization. They receive a large volume of mail, so large that they do not manage to file it all in neat separate mail folders. A common situation is looking for a piece of mail received some time back, but unfortunately, our user cannot remember how long ago – was it 2 weeks or 2 months? Neither do they remember the exact name on the "From: xxx" header, although they likely have a vague recollection of one or a few letters that it contains. They may or may not have a vague recollection of some part of the Subject line. Our user keeps mail for many months, possibly longer, knowing that these situations arise. However, finding exactly what one is looking for is more difficult.

We start with a user that has entered an initial set of keywords, e.g. Sender and Subject, based on whatever approximating string of letters they can muster from memory. The search engine produces a list of candidate emails in response and based on these, the user modifies the keywords; this process may repeat some (small) number of times. The specific modification to keywords employed by a user – in response to the material offered by the search engine – typically has no fixed recipe, and may well change substantially from one iteration to the next. Clearly, intuition is being relied upon heavily to modify keywords so as to promote the user's objective.

In this setting we can directly identify all 3 of the dimensions indicated by the distributed approach. Thus,

1. In addition to the user's brain, the search engine is a participating artifact.
2. The time scale of the process extends over successive iterations of keyword entry, examination of results and keyword modification.
3. Search is directly influenced by the qualitative content of material returned by the engine. This material in turn is the direct contribution of others in the email community of which the user is a member. The principle of embodiment allows the influence of this material on the structure of the cognitive process.

Our goal is to participate in this cognitive process, and in so doing – enhance it. The chief difficulty is that the user does not know exactly what they are looking for – else they would enter the best keywords directly. Thus standard optimization algorithms are not applicable since we don't have a definition of what to optimize. However, we do know that our user eventually finds what they are looking for in most cases, hence we can posit an approximate monotonicity of progress versus time. That is, although there may be missteps and setbacks, on average there are significantly more steps forward than backward. We can therefore undertake a supportive role comprising two elements, as follows:

1. Remind the user of how they have progressed in similar situations in the past, with the goal of avoiding backwards steps, and
2. Abbreviate multiple steps into a single one, in cases where the user has found the sequence repeatedly useful in the past.

We still have a significant challenge to overcome, viz. how can one relate "situations in the past" and situations "found repeatedly useful" in the past – to the present context, when the notion of "useful" has no representation or definition that is accessible to us? Our approach is to rely on embodiment, viz. what is going on in the cognitive process must be parallel in some sense to the artifacts in the environment to which the process is being applied, whence it follows that we can track the cognition by simulation with those artifacts. Specifically, in the current application the artifacts are keywords. If we find that in previous iterations the user has tried to pursue progress

toward the goal by modifying keyword  $a$  to keyword  $a'$ , i.e. the user has made an ordered association between two versions of a keyword, then we can conclude that this association is not the result of an intuition totally abstracted from the reality of words, word-parts or speech sounds etc. but rather that cognition is implicitly relying on a distance metric on the space of words itself. If we can find a metric for which the search iterations cluster into well defined repetitive associations then we have found a structure that parallels the cognition.

In the next section we make these ideas more precise via formal definition as an algorithm, which we shall call the Interactive Search Optimization (ISO) algorithm. We note that the above description of an email search process applies, essentially unchanged, to a search of the internet. We chose the email setting for our application because it was simpler to implement. For the internet case, a more elaborate and specialized characterization as a distributed cognitive process has been suggested in [4]. However, the three-dimensional characterization given above is adequate for our development here, and applies equally well to both problem settings.

## 2. Formal Development

We consider keywords from the 26 letter English alphabet only. The number of letters in our keywords is not restricted – other than by computational resources, and one can arbitrarily assign e.g. the first five letters to the "From:" key, the next five to the "Subject:" key, etc. Without loss of generality therefore, we consider in the sequel a single keyword of arbitrary length  $m$ .

Thus we have  $m$  dimensions of variation with 26 possible values each. Each of these variables is mapped to an angle  $\theta$  as depicted in Figure 1.

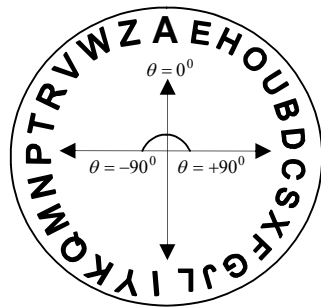


Figure 1 – Base Ordering and Initial Position

The ordering around the circle follows typical associations between the sounds of letters – as they are actually experienced by humans, viz. (a e h o u), (b d), (c s x), (g j), (i y), (k q), (m n), (p t) and (v w) are the groupings used. The remaining letters are singletons. The groups and singletons in turn are ordered approximately as they appear in the alphabet. This is called the *base ordering*. In the *initial position*, the angle  $\theta = 0^\circ$  points to the letter 'A'.

**Definition 1: User Response and Computation of Action Vector**

A *delta search vector*, is an ordered sequence of angles  $(\theta_1, \theta_2, \dots, \theta_m)$ ,  $0^0 \geq \theta_j \geq -180^0$  or  $0^0 < \theta_j \leq +180^0$ ; each  $\theta_j$  represents a (possibly zero) rotation away from the current position. The user submits a delta vector to modify the keyword represented by the current position. The intention of this arrangement is that, in addition to recording the user's revision of a keyword, we also capture a notion of how the user judges the progress of the search. Specifically, the zero direction indicates "no change required", whence user responses  $\theta_j$  that are close to zero will indicate generally that we are in the right direction, and conversely – the farther away from zero (or the closer to  $\pm 180$  degrees), the greater the user dissatisfaction with the progress of the search. The delta vector is provided by the user in the *response matrix*  $\begin{bmatrix} \theta_1, \theta_2, \dots, \theta_m \\ w_1, w_2, \dots, w_m \end{bmatrix}$ , along with a vector of corresponding weights  $w_j$ ,  $0 < w_j \leq 1$ , the latter representing the user's indication of their own confidence in the former.

Instead of applying the delta in the user response directly – to obtain the next keyword, the ISO algorithm applies transformations to the response matrix based on what it can learn from the history of user responses thus far. The transformed version is called the *action* matrix because its delta vector is the one actually input to the search engine. Thus we have a sequence of pairs  $(A_1, R_1), (A_2, R_2), \dots$  etc., where the index indicates the particular iteration of results/user response,  $A_i$  is the action matrix whose delta vector is input to the search engine and  $R_i$  is the corresponding user response matrix. The role of the weights  $w_j$  and the transformations that generate the  $A$  matrices will be defined in the sequel.

**Definition 2: Unification**

Write  $\delta_j$  to denote one of the values  $\theta_j$  or zero. Upper case  $X$  denotes any delta matrix. The unification mapping  $U(X_k, X_{k+1}, \dots, X_{k+l}) \rightarrow X'$  may be expressed as,

$$X' = \begin{bmatrix} \theta'_1, \theta'_2, \dots, \theta'_m \\ w'_1, w'_2, \dots, w'_m \end{bmatrix}, \begin{cases} P(\theta'_j = \sum_{h=k}^{k+l} [\delta_j]_h) = \prod [p_j]_h \\ w'_j = \prod [p_j]_h \quad \text{if } \theta'_j = \sum_{h=k+1}^{k+l} [\delta_j]_h \end{cases}$$

where

$$[p_j]_h = \begin{cases} [w_j]_h & \text{if } [\delta_j]_h = [\theta_j]_h \\ 1 - w_j & \text{if } [\delta_j]_h = 0 \end{cases},$$

$P(\theta = x)$  denotes the probability that the random variable  $\theta$  takes the value  $x$  and this probability is specified for all possible values of the  $m$  placeholders  $\delta_j$ . That the distribution for  $\theta'_j$  is a valid probability mass function (PMF) may be readily verified by noting that the distribution of each  $\theta'_j$  is identical with the joint PMF of

independent identically distributed binary random variables  $[\delta]_h$ ,  $0 \leq h \leq l$  with probabilities  $P([\delta]_h = True) = [w]_h$  and  $P([\delta]_h = False) = 1 - [w]_h$ .

We need the following functions, defined on any delta matrix  $X$ .

**Definition 3: User Grade**

1. The *User Grade* (URG):

$$urg(\theta) = \begin{cases} +1, & \text{if } -45^0 \leq \theta \leq +45^0 \\ 0, & \text{if } -135^0 \leq \theta \leq -45^0 \quad \text{or} \quad +45^0 \leq \theta \leq +135^0 \\ -1, & \text{if } -180^0 \leq \theta \leq -135^0 \quad \text{or} \quad +135^0 \leq \theta \leq +180^0 \end{cases}$$

2. The normalized, weighted, scalar user response  $\theta^S$ :

$$\theta^S = \frac{1}{\sum_{j=1}^m w_j} \sum_{j=1}^m E(\theta_j), \quad \sum_{j=1}^m w_j \neq 0,$$

where  $E(\theta_j)$  is the expectation of the random variable  $\theta_j$ .

$\theta^S$  summarizes the information in a delta matrix with a single number.

3. The normalized weighted User Grade  $urg(\theta^S)$ . In the sequel, URG refers to the normalized weighted version, unless otherwise indicated.
4. The *Dimensional Progress Index* (DPI). This is a relative measure of cumulative progress toward the goal – as perceived by the user. It is defined recursively as follows:

$$dpi_i = \begin{cases} dpi_{i-1} + urg_i, & i > 0 \\ 0, & i = 0 \end{cases}$$

We think of the search space as comprising dimensions that are successively constrained or fixed so that the balance of the space to be searched gets progressively smaller. In this conceptualization, progress toward the goal may be measured by the number of dimensions that have been successfully fixed – hence the term Dimensional Progress Index.

The pair  $[urg_i, dpi_i]$  is our overall user grade for iteration  $i$ . We must also specify if the grade function is taken on the response matrix  $R_i$  or on the action matrix  $A_i$ .

**Definition 4: Generations**

The delta matrices  $A_i$ ,  $R_i$  and their associated user grades  $[urg, dpi]$  comprise the  $i$ th *actual generation*. The accumulated sequence of actual generations is called the *history*. An actual generation is always a single  $(A_i, R_i)$  pair. Actual generations are distinguished from *virtual* generations (defined next), which are delta matrices the ISO algorithm will compute as candidates for input to the search engine, and which may or may not become actual. A virtual generation may have more than one element.

At each iteration  $i$ , the algorithm will compute a virtual next generation of candidate matrices  $A_{i+1}$  by applying unification to subsequences of actual generations

in the history, in accordance with the unification rules below. The virtual generation is computed from scratch at each iteration; there is only one such generation at any time.

**Definition 5: Unification Rules**

The symbol ' $\cong$ ' denotes fuzzy equality, useful for e.g. comparison between DPI values which are in any case user subjective. We use standard fuzzy set membership based on a parabolic function [5].

In what follows,  $i$  is the current iteration number, while  $k$  is an arbitrary iteration number in the history. A superscript distinguishes functions associated with  $R$  matrix from those associated with the  $A$  matrix.

1. Unify all successive pairs of action matrices found in the history ( $\dots A_k, R_k, A_{k+1} \dots$ ) such that

$$urg_k^a = urg_i^r \text{ and } dpi_k^a \cong dpi_i^r .$$

2. Unify all subsequences of action matrices found in the history ( $\dots A_k, R_k, A_{k+1}, \dots A_{k+l}, \dots$ ) such that

$$urg_{i-1}^r = +1, \quad urg_i^r = -1,$$

$$urg_{k-1}^a = +1, \quad urg_{k+h}^a \neq +1, \quad \forall h \in [0, l], \quad urg_{k+l+1}^a = +1 \text{ and } dpi_k^a \cong dpi_i^r .$$

Note that the functions  $urg$  and  $dpi$  for the current iteration associate with the response matrix  $R_i$  while the those from the history associate with the action matrix  $A_k$ . In general, we look for precedent in the action matrices of the history rather than the response matrices, since the former correspond what actually took place. In contrast, for the current iteration, we are in the process of computing the action matrix so the response matrix  $R_i$  is the most up-to-date indication of progress. The second rule is intended to capture a sequence of miss-steps, i.e. skip over all the "wrong way" attempts a user might make while chasing a dead end. Note that the cases are not mutually exclusive, i.e. where  $urg_k^a = urg_i^r = -1$  both unifications may apply.

A delta matrix computed from unification will be called a *synthetic* action, i.e. we are taking the liberty of transforming the user's response based on the history. We apply a ranking of the synthetic actions in the virtual generation. The winner, if there is one, will be used instead of the current user response to form the next actual generation.

We compute a Consistency Index on the elements in the virtual generation. Denote by  $A'$  a synthetic action for which the index is to be computed, and let  $\mathbf{A}$  be the set of all action matrices accumulated thus far, synthetic or not, i.e. the union of the virtual generation and all actual generations. Consistency means the degree to which the action of  $A'$  is confirmed by other action deltas, and it is computed for  $A'$  separately relative to each  $A \in \mathbf{A}$ . Consistency has two criteria. The first checks that the action  $A \in \mathbf{A}$  was applied with DPI similar (fuzzy equality) to the current value<sup>1</sup>. The second looks for similarity of the user grade  $\theta^S$ . Denote by  $eq : \mathfrak{R} \times \mathfrak{R} \mapsto [0, 1]$  a fuzzy equality function on pairs of real numbers, i.e. with image in the continuous interval  $[0,1]$ . We first compute a Local Consistency Index (LCI) for  $A'$  relative to a particular  $A \in \mathbf{A}$ ,

---

<sup>1</sup> Unifications may be nested to arbitrary depth, hence this requires that we maintain, for any synthetic action matrix, a record of the DPI that held at the first iteration of the first level of unification.

$$\text{LCI}(A') = \text{Min} (eq(\theta^S(A'), \theta^S(A)), eq(dpi(A), dpi_i)), \quad A \in \mathbf{A},$$

where  $dpi_i$  is the DPI associated with the current user response. We then apply *concentration* and *fuzzification*, standard techniques from fuzzy logic [11], based on the actual *urg* which followed the action  $A$ . If that  $urg \neq -1$ , we take the square root of the LCI, otherwise we square it<sup>2</sup>. The former magnifies the LCI; the latter reduces it.

Finally, we compute the Global Consistency Index (GCI) for each synthetic  $A'$  in the virtual generation,

$$\text{GCI}(A') = \sum_{A \in \mathbf{A}} \text{LCI}(A', A).$$

There are two cases in which the above procedure fails to produce a candidate for the action matrix  $A_{i+1}$ : where no sequence in the history qualifies for unification, or where the GCI is zero for all unifications computed. These will occur particularly in the early iterations when the history is short (or non-existent). In either of these situations we say that *synthesis fails*. We can now define the transformation that computes the action matrix  $A_{i+1}$  at each iteration. Denoting by  $\mathbf{VG}$  the virtual generation, we have,

$$A_{i+1} = \begin{cases} U(R_i) & \text{if synthesis fails} \\ \arg \max_{A \in \mathbf{VG}} \text{GCI}(A) & \text{otherwise} \end{cases}.$$

Unification of the user response matrix with itself – indicated for the case of synthesis failure, is just a non-deterministic rendering of the user's response, viz. each  $\theta_j$  is applied with probability  $w_j$ ; with probability  $1 - w_j$  we apply a zero delta. The algorithm iterates until the user indicates the target has been found (or gives up).

### 3. Experimental Results

In this section we present our experience with the use of the ISO algorithm described in the previous section<sup>3</sup>. Our first task was to verify that the algorithm performs its intended function. That is, allowing our hypothesis from Section 1.2 that intuition makes use of a distance metric on keywords, does the algorithm in fact improve the efficiency of search? We constructed automata that simulate a user's search activity under different behavioral assumptions. We then compared the number of steps required by an automaton to reach the search objective to the number of steps required when that automaton is supported by the ISO algorithm. The results show that, for a significant range of behavioral parameters, and in particular such as would be expected in realistic human contexts, the algorithm significantly reduces the number of search steps. Of course the situation of a human user is fundamentally different in that they do not know the target in advance. Hence these experiments cannot directly verify the key "Embodiment" thesis, viz. that an ISO-like process actually tracks cognition. Experiment with humans remains for future work.

---

<sup>2</sup> Applicable only to LCI relative to actual matrices  $A$  in the history, not virtual ones.

<sup>3</sup> In practice, some minor additional tuning and/or approximation to the algorithm was used. They are not detailed due to lack of space.

### 3.1. Types of Automata

In each of the automata described, the software knows the target keyword, and advances toward it, away from it or just "gets lost" – in accordance with a specific behavioral policy that is intended to approximate a human user's behavior. A parameter  $\alpha$  introduces "errors" into the search behavior; with probability  $\alpha$ , the automaton presents a user response (i.e. a delta angle  $\theta$ ) in accordance with its particular behavioral policy, and with probability  $1 - \alpha$  it sets the delta angle  $\theta$  at random. We constructed 4 different automata, as follows:

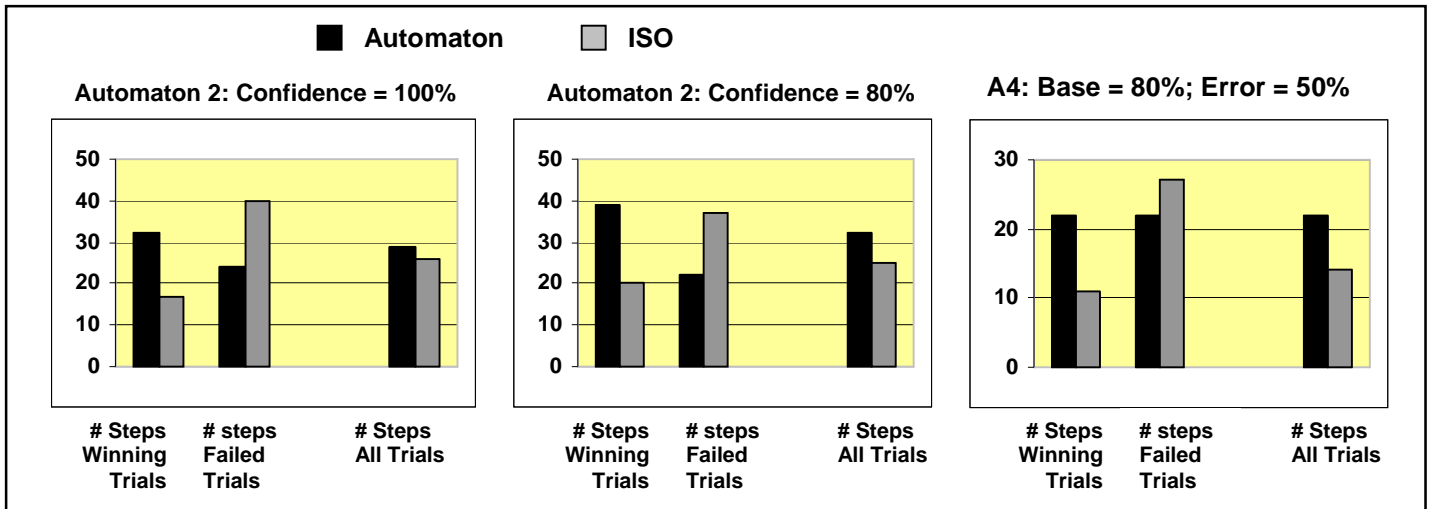
- A 1: Each response delta advances 1/4 of the distance to the target, relative to the base ordering. The random errors introduced with probability  $\alpha$  are not given any special treatment, and enter the history as any other user response.
- A 2: Same as Automaton 1, however, we simulate also the likely intervention of a human user after a move in the wrong direction. After a random delta is executed the automaton retracts its move, i.e. presents an inverse delta, that returns the search to the same position it was before the random move.
- A 3: Same as Automaton 1, except as follows. We maintain a parallel record of the progress that would have been achieved by the unaided automaton. If at any point the unaided automaton gets ahead of the ISO algorithm, it "notifies" that the system is not responding effectively and repeats its last response, i.e. the one which brought the unaided search into the lead.
- A 4: The response deltas produced by the automaton alternately advance toward, and retreat away from the target keyword, in accordance with the following sequence.

$$\frac{1}{n} - \frac{1}{n+2} + \frac{1}{n+1} - \frac{1}{n+4} + \frac{1}{n+3} - \frac{1}{n+6}, \dots, \text{etc.}$$

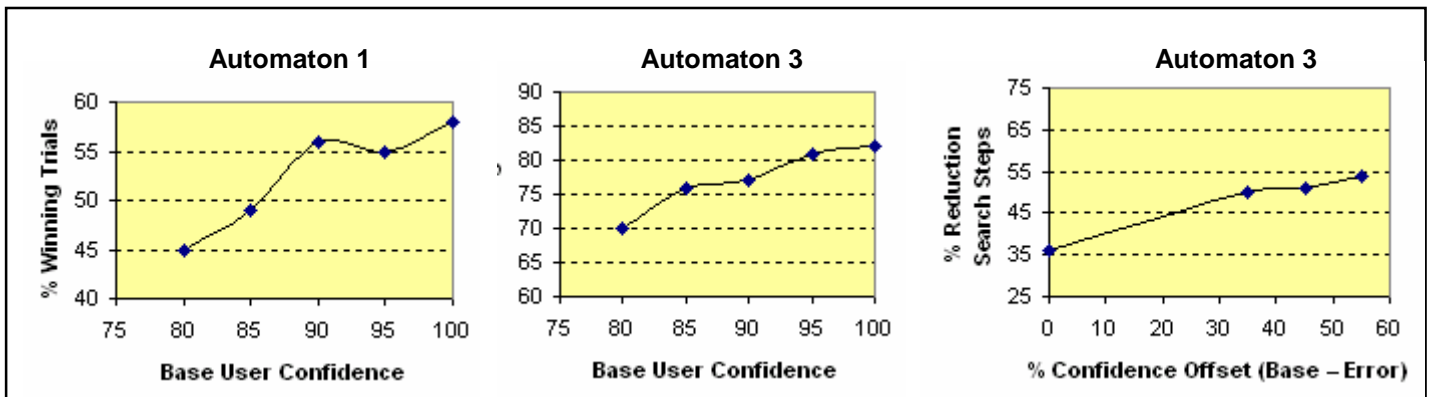
where  $n$  is a small integer such as 2 or 4, and each term represents a fraction of the current distance to the target. This series is dominated by  $\sum 2/n^2$  and hence converges. As long as the limit of the sum is  $> 1$ , one necessarily arrives at the target after a sufficient number of steps. More interesting in our context is that this behavior simulates "two steps forward and one step back".

Parameters for all the automata were varied over a range of values. In addition to the value of  $\alpha$ , we varied the confidence weights associated with each delta in two different ways. In the simplest approach, the same confidence level was fixed for all user (i.e. automaton) responses throughout the search. For a more realistic scenario, we applied one level of confidence to those responses which followed the behavioral policy of the automaton and a different confidence value for the "error" responses, i.e. those generated randomly.

**Figure 2** gives representative results for two of the automata; results for the other two are similar. In **Figure 3** we show the results as a function of varying confidence, and of varying differential between confidence for useful Responses (labeled "Base Confidence") as opposed to confidence for "error" responses. All data represent an average of 100 searches repeated with the same target keyword.



**Figure 1** – Reduction in number of search steps required, relative to the unaided user as simulated by Automaton 2 and 4. In the first two columns the automatons used a fixed confidence level; in the last column the random inputs indicated lower confidence. A "Winning Trial" means the ISO algorithm succeeded in reducing the number of steps. All data points are an average of 100 trials.



**Figure 2** – The first two graphs show the proportion of winning trials for ISO, as a function of the user confidence parameter. Error confidence is constant at 50%. For the 3rd graph error confidence varies while base confidence is held constant at 85%. The % reduction for ISO is shown as a function of their difference.

#### 4. Related Work

The ISO algorithm has parallels with Estimation of Distribution algorithms (EDA) [6,7]. After unification is applied, ISO filters the results via the local and global consistency indices; this may be viewed as a selection function. Unification itself amounts to construction of a distribution and its use to perturb candidate actions.

However, there are key technical differences between ISO and EDAs. Whereas EDAs retain the basic GA notion of perturbing the entire surviving population, our

algorithm selects particular individuals for Unification, as dictated by the current user Response. Of course the current user Response is an influence that is absent in the conventional GA problem setting. Further, in contrast to ISO, EDAs and evolutionary algorithms in general are concerned at any generation only with the currently surviving population. Additional important differences may be added to this list.

Similarity with the "online" form of Neural Networks maybe noted [8]; more specifically, Recurrent Neural Networks [9,10] learn from a history of errors, as does ISO. However ISO differs in many ways from any form of neural net. An obvious difference is that it does not adjust weights.

In our view, the key innovation of ISO is the way the problem is cast rather than the technical particulars of the algorithm used to solve it. Simply, we consider a search where we begin without any well-defined criteria for testing a correct solution. This of course, is exactly what happens when a human user searches email or the internet for a target that is only vaguely remembered. In the next section we explore the implications of treating this situation as a legitimate computational problem.

A more significant similarity may be observed between ISO and so-called "Putnam-Gold Machines" [11]. Computation in the latter do not have a fixed ending. Similarly, any particular output produced by ISO, is not, in general, the "real" output.

## 5. Discussion and Conclusions

The ISO algorithm is not a self-contained "artificial intelligence", since it relies explicitly on human direction. Nevertheless, we have an AI "result", i.e. the useful outcome of reduced search times. This is achieved by artificial enhancement of one part of a human cognitive process, rather than by a process that is entirely artificial.

As noted in the introduction, Turing and his colleagues believed that Turing computation, i.e. any mechanical process is necessarily devoid of intelligence. This suggests that to achieve general intelligence we must go beyond Turing computation. What follows is an informal argument, suggesting that the ISO technique offers a way to actively participate in just such a non-Turing computation.

### *Proposition 1:*

A Turing computation is completely specified, hence must have a well-defined goal. At a minimum, the definition of this goal is the specification of what it does. The exception is random input, i.e. a non-deterministic machine; the random influence makes the machine incompletely specified.

### *Proposition 2:*

The search problem for ISO does not have a well-defined goal, hence not a Turing computation. Likewise, the sequence of user inputs to ISO is non-Turing.

Next we show that this non-Turing behavior is not random. The ISO method interprets user choices based on the earlier history of those same choices, i.e. we assume the user may not have the memory or analytical resources to choose in a manner that is consistent with previous choice behavior. The implicit assumption is that effective native user progress is itself based in some sense on offsets from previous choices. Put another way, we assume the user would want to be consistent with the history, if they did have the resources.

If the ISO algorithm consistently reduces search times, then a correspondence must exist between the sequence of action deltas generated by the algorithm and the goal of

the process at work in the mind of the user. For example, it may be that the sequence approaches the target of the search in some limit sense. Alternatively, the sequence of outputs from ISO may correspond to a succession of revisions of the target, without specifying any structure on the succession of targets or the basis for revision. There may be other ways of setting up this correspondence.

Denote by  $I = [i_1, i_2, \dots, i_n]$  the sequence of inputs provided by the user to ISO and by  $O = [o_1, o_2, \dots]$  the outputs it generates in response. That a correspondence exists between  $o_i$  and  $i_i$  is straightforward. The more significant observation however, is that each  $o_i$  relates in some way to  $i_n$ , the final target of the search. This must be so since the user succeeds in reaching the target in fewer steps when supported by ISO than with unaided intuition alone. We say that the ISO algorithm is *synchronizing* with the user cognitive process. We have:

*Proposition 3:*

It is not possible to synchronize with a random sequence. The ISO algorithm synchronizes with the user input sequence, so that sequence is non-random. *Hence ISO synchronizes with a non-random, non-Turing computation.*

The ISO algorithm, by exhibiting the above-described synchronization behavior, offers a means to explore the process by which such a non-Turing results may be arrived at. It also suggests that explicit formulation of one of the distributed components of a cognitive process may be a promising alternative paradigm for "algorithm". Learning from experiment with such working systems, we may be able to construct explicit enhancements – or indeed replacements – for progressively more of the components of such processes. Eventually, we may construct all of them, hence achieving a self-contained artificial intelligence.

## References

- [1] T. Ord, *The Many Forms of Hypercomputation*, Applied Mathematics and Computation, Vol .178, 2006, pp. 143-155.
- [2] M. H. A. Newman, *Allan Mathison Turing*, Biographical Memoirs of the Royal Society, 1955, pp. 253-263.
- [3] J. Hollan, E. Hutchins and D. Kirsh, *Distributed Cognition: Toward a New Foundation for Human-Computer Interaction Research*, ACM Transactions on Computer-Human Interaction, Vol. 7, No. 2, June 2000, Pages 174–196.
- [4] C. Heintz, *Web search engines and distributed assessment systems*, In: Harnad, Stevan and Dror, Itiel E. (eds.), *Distributed Cognition: Special issue of Pragmatics & Cognition* 14:2 (2006). 268 pp. (pp. 387–409)
- [5] G. Klir and B. Yuan, *Fuzzy Sets and Fuzzy Logic: Theory and Applications*, Prentice Hall, 1995.
- [6] P. Larrañaga and J. A. Lozano (Eds.), *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*, Series: Genetic Algorithms and Evolutionary Computation , Vol. 2, Kluwer Academic, 2001.
- [7] M. Pelikan, D. E. Goldberg and F. G. Lobo, *A Survey of Optimization by Building and Using Probabilistic Models*, Computational Optimization and Applications, Springer, 2002, pp.5-20.
- [8] D. Saad (ed.), *Online Learning in Neural Networks*, Cambridge University Press, January 1999.
- [9] M. Danilo, and J. Chambers, *Recurrent Neural Networks*, Wiley, 2001.
- [10] S. Hochreiter and J. Schmidhuber, *Long Short-Term Memory*, Neural Computation, Vol 9 No 8, 1997, pp. 1735-1780.
- [11] P. Kugel, *Computing Machines Can't Be Intelligent (...and Turing Said So)*, Minds and Machines, Vol.12, No. 4, September 2002.