A Vivisection of the ev Computer Organism: Identifying Sources of Active Information

George Montañez¹, Winston Ewert¹, William A. Dembski², and Robert J. Marks II²*

¹ Department of Computer Science, Baylor University, Waco, Texas, USA; ² Discovery Institute, Seattle, Washington, USA; ³ Department of Electrical and Computer Engineering, Baylor University, Waco, Texas, USA

Abstract

ev is an evolutionary search algorithm proposed to simulate biological evolution. As such, researchers have claimed that it demonstrates that a blind, unguided search is able to generate new information. However, analysis shows that any non-trivial computer search needs to exploit one or more sources of knowledge to make the search successful. Search algorithms mine active information from these resources, with some search algorithms performing better than others. We illustrate these principles in the analysis of ev. The sources of knowledge in ev include a Hamming oracle and a perceptron structure that predisposes the search towards its target. The original ev uses these resources in an evolutionary algorithm. Although the evolutionary algorithm finds the target, we demonstrate a simple stochastic hill climbing algorithm uses the resources more efficiently.

INTRODUCTION

Computer algorithms can serve as powerful tools for modeling and studying facets of nature, including biological phenomena. As with all models, extrapolation from algorithm behavior to real-world behavior requires that the algorithm be based on realistic operating assumptions and produce behavior characteristic of the system in question. Inaccuracies or hidden biases can invalidate an algorithm as a model of natural phenomena. Modeling the evolutionary process as an algorithmic search process, therefore, requires diligence on our part to ensure that all algorithm assumptions are realistic and that no undisclosed knowledge sources are used to ease the difficulty of our search.

Algorithms that conduct even moderately sized searches require external assistance to be successful. When such an algorithm produces apparently impressive results, conservation of information [1-3], including the No Free Lunch Theorem [4-11], dictates we are faced with one of two possibilities. The first is that the search problem under consideration is not as difficult as it first appears. At times, the problems solved by seemingly complex algorithms can appear extremely difficult whereas a closer inspection reveals the search is relatively simple and, from a random query or exhaustive search perspective, has a larger probability of success than implicitly supposed. The other alternative, for difficult problems, is that active information has been inserted in the search program to increase the chances of success.¹

¹ Formally, active information is defined as \(-\log(p/q)\) where \(p\) is the probability of success for an unassisted search and \(q\) is the probability of success for an assisted search. Informally, it is the amount of information added to the search that improves the probability of success over the baseline search.

A common source of active information is a software oracle² [12-14]. When the oracle is the dominant computational component in an evolutionary search and the cost of each query is the same, the efficiency of a search algorithm can be measured in query counts. Different search algorithms extract information with varying efficiencies. A simple single agent algorithm that uses a stochastic hill climbing ratchet [13], for example, is much more efficient than an evolutionary search using the oracle available in the artificial life simulation AVIDA [15]. Active information can also be extracted from a Hamming oracle.³ The effectiveness of this oracle can range from that of a standard evolutionary search to a deterministic frequency of occurrence Hamming oracle algorithm, which can produce over an order of magnitude improvement [13].

This paper presents an analysis of the ev program, a search algorithm that models the evolution of nucleotide binding sites [16]. The ev algorithm has the form of a perceptron, consisting of a single artificial neuron using a simple threshold nonlinearity. Our purpose is to analyze how ev’s search structure constrains performance in light of conservation of information theorems for search algorithms. In particular, we investigate the degree to which ev’s success is due to active information introduced into the simulation by the structure of the perceptron used to generate potential solutions and the Hamming oracle used to evaluate the fitness of the solution.

² A software oracle is a software object that answers queries posed to it. In our case, a software oracle is a function that takes in a configuration and returns a value denoting the fitness of that configuration.

³ A Hamming oracle uses the Hamming distance (number of bits that differ from a target sequence) as its fitness metric.
ANALYSIS AND RESULTS

Information measures

We use the following information measures [1,2,13,17,18] to assess the performance of a search:

**Endogenous information** is a measure of the difficulty of a search and is given by

\[ I_{\Omega} = - \log_2 p \]  

where \( p \) is the probability of success for an unassisted query.

**Exogenous information** in a search program is given by

\[ I_q := - \log_2 q \]  

where \( q \) is the probability of success of an assisted query under the same set of constraints.

**Active information** is the difference between the endogenous and exogenous information, denoted as

\[ I_\oplus = I_{\Omega} - I_q = - \log_2 \frac{p}{q} \]  

Each of the three information measures has units of bits. If the knowledge about the space is not accurate or is otherwise misleading, the active information can be negative.

In addition, for any search algorithm, such as **ev**, a resource constraint must be imposed. Otherwise, unlimited time and computer resources would allow an exhaustive search. Let \( Q \) denote the query count, \( Q_{\max} \) the maximum allowable number of queries before abandoning the search, and \( E \) the expectation \([19]\). Under a query count constraint, the active information per query, \( I_{\oplus} \), is

\[ I_{\oplus} = E \left[ \frac{I_{\oplus}}{Q} \right] \]  

The active information per mean query, \( I_{\oplus} \), can be estimated by averaging the active information per query over \( K \) trials of \( Q_{\max} \) queries or less. For the \( k^{th} \) trial, there are two possibilities. Either success is achieved with \( Q_k \leq Q_{\max} \) queries, in which case the point estimate of \( I_{\oplus} \) is \( I_{\oplus} = I_q/Q_k \), or if a success is not achieved with \( Q_{\max} \) queries, then the point estimates of \( I_\oplus \) and \( I_\ominus \) both have a value of zero.

Thus, defining \( \xi_k \) such that \( \xi_k = 1 \) for a success in \( Q_{\max} \) or fewer queries, and \( \xi_k = 0 \) and \( Q_k = Q_{\max} \) for a failure, we can estimate \( I_\oplus \) as the average over \( K \) trials.

\[ \langle I_{\oplus} \rangle = \frac{1}{K} \sum_{k=1}^{K} \xi_k \left( \frac{I_{\ominus}}{Q_k} \right) \]  

\[ = \frac{I_{\ominus}}{K} \sum_{\text{successes}} \frac{1}{Q_k} \]  

This estimate, \( \langle I_{\oplus} \rangle \), needs to be interpreted with the same caution as the average speed of an auto on a road trip. Instantaneous values can be significantly higher or lower than the average.

We refer to \( I_q/I_{\ominus} \) as the normalized active information per query, and the estimated normalized active information per query is given by

\[ \frac{\langle I_{\oplus} \rangle}{I_{\ominus}} = \frac{1}{K} \sum_{\text{successes}} \frac{1}{Q_k} \]  

(6)

A similar measure is the active information per mean query,

\[ I_{\ominus} = I_{\ominus}/E[Q] \]  

(7)

which we can estimate by

\[ \langle I_{\ominus} \rangle = \frac{I_{\ominus}}{\langle Q \rangle} \]  

(8)

where

\[ \langle Q \rangle = \frac{1}{K} \sum_{\text{successes}} Q_k \]  

(9)

Analysis of endogenous information in ev

The **ev** search algorithm can be viewed as an inversion of a perceptron \([21-23]\). This is illustrated in Figure 1 on the next page. The **ev** simulation creates a population of 64 synthetic organisms, each with a genome consisting of a 256 base string (excluding five extra bases at the end that are not part of the genome proper), with the bases stored as two-bit integers: A=00, C=01, G=10 and T=11. There are, therefore, \( n = 256 \) nucleotides at the perceptron output that serve as potential binding sites each with a \( \lambda = 4 \) possible bases. In this sequence, 16 of the 256 nucleotide site locations are designated as binding sites. The **ev** simulation begins by randomly assigning these 16 binding sites within the second half of the genome and fixes them for the duration of a run. The sequence of ones (binding sites) and zeros (not a binding site) are shown at the bottom of Figure 1. A published example \([16]\) uses the specific locations

\[ t = [1, 10, 17, 26, 33, 43, 50, 60, 70, 76, 83, 92, 101, 109, 117, 125]. \]  

(10)

There is also a \( \Gamma \times \lambda \) weight matrix (see Figure 1) with \( \Gamma \lambda = 24 \) elements that are represented as integers in the range of \([-R, R-1]\) where \( R = 512 \). Since \( 2R = 1024 = 4^5 \), five bases are needed to give each weight its value. There is a single bias, \( \theta \), specified by five bases and having the same range as the weight elements. The perceptron works by sequentially processing the genome in blocks of \( \lambda = 6 \) bases. As shown in Figure 1, each of the six columns are activated at one of four locations (A, C, G or T) in accordance to the nucleotide above. The weights of these activated locations are summed. The bias, \( \theta \), is subtracted from this sum and the result compared to a threshold. If positive, the location is announced as

---

4 The plus sign subscript on \( I_q \) represents information being added about the search.

5 A similar active information rate was used in the dissection of AVIDA \([18]\) using an instruction count rather than queries.

6 From Jensen’s inequality \([20]\), \( \|I_q\|_2 \leq \|I_q \|_1 \) \( \frac{1}{10} \). Thus \( I_{\ominus} \geq I_q \).

7 This is a process by which we find a set of weights and bias that give us some desired output behavior for our perceptron.

8 Although all 256 positions along the genome are evaluated for errors and contribute to an organism’s fitness, the randomly placed binding sites are restricted to the second half of the genome. In Figure 1 of reference 16, these correspond to bases 126 to 256. There are other nucleotides whose identities are interpreted as weights, window values, or the bias in the construction of the perceptron. Five additional bases are used at the end to accommodate a sliding window used in **ev**.

9 The target binding sites start at location 131 (zero-indexed) in the first Figure of reference 16. Thus, location 16 here corresponds to nucleotide 141.
Analysis based on ev perceptron structure. The more interesting case for analysis involves a search using the perceptron structure and the organism’s genome, rather than modifying its output directly. In this case, we search for a sequence of nucleotides that produces a weight matrix and bias that calculates a perceptron output identical to the target.

In the search for the binding sites, the target sequences are fixed at the beginning of the search. The weights, bias, and remaining genome sequence are all allowed to vary. Finding them is a form of the perceptron inversion problem [22]. A search space \( \Omega \) consists of all possible values of weights, genome sequence, and bias. There are 256 nucleotide bases (A, C, G, T) used for each simulation of ev. There are 5 additional bases used for a boundary condition for a sliding window across some of the bases. Including these bases, the search space is the 261 fold Cartesian product of the (A, C, G, T) bases. Bases are interpreted in different ways in the search:

1. As a nucleotide on the input strip at the top of Figure 1.
2. As a weight in the \( \mathcal{T} \) matrix. Each weight has 10 bits (5 bases) of precision.
3. As the bias threshold \( \theta \). The bias also has 10 bits of accuracy.

Since there are four bases, the cardinality of the search space\(^{10}\) is
\[
|\Omega| = 4^{261} = 1.37 \times 10^{157} \quad (11)
\]

Every point in \( \Omega \) generates an XOR output of 256 bits. The search target \( \Omega_e \) is the set of all points in \( \Omega \) that generate the 256 bit target sequence at the output.

Performance analysis of the ev perceptron

We now analyze the source the active information provided by ev’s perceptron structure. Denote the sum entered into the threshold operator in Figure 1 by \( S \). There are 256 values of \( S \) in the output string. Let’s call the \( k \)th sum \( S_k \) and concatenate them into the vector
\[
\vec{S} = [S_1 \ S_2 \ S_3 \ \ldots \ S_k \ \ldots \ S_{256}]^T \quad (12)
\]

A given \( S_k \) is the sum of seven independent and identically distributed (i.i.d.) uniform discrete random variables. The random variables in \( \vec{S} \), though, are not independent. Each, for example, contains the same bias, \( \theta \), as one of the seven numbers in its sum. If the output bits were independent, every possible binary string would have the same probability as any other and the same distribution of error. According to the Laplace-DeMoivre theorem [19, 24], they would therefore be Gaussian as in Figure 2B. However, the normalized histogram for ev in Figure 2A is far from Gaussian (see below for a demonstration). If the bias is large, it can push

---

\(^{10}\) This number includes the five bases that are not searched.
An upper bound on exogenous information. Despite this predisposition, the probability of an unassisted search generating zeros with sparsely occurring ones, as specified for example in Equation 10, is still very small. We have two empirical estimates of the upper bound on the endogenous information of the ev problem. In both cases we conservatively assume there is only one perceptron that generates an output matching a desired 16 bit target, like that of Equation 10.

First, from the data used to plot Figure 2A, the frequency of occurrence of strings with zeros with precisely 16 ones is 0.0024. The probability of hitting the target in Equation 10 is then \( p = \text{Pr}[\text{hitting target}] = \text{Pr}[\text{hitting target | 16 ones}] \times \text{Pr}[16 \text{ ones}] \). There are

\[
\frac{1}{\pi} = \left( \frac{256}{16} \right) = 10^{25}
\]

(14)

possible ways to arrange 16 ones and 240 zeros. Because there have been successful simulations, we know at least one of these can match the target sequence. Thus \( \text{Pr}[\text{hitting target | 16 ones}] \geq \pi \) and \( p \geq 0.0024 \times 2 = 2 \times 10^{-28} \). We can therefore estimate

\( I_S < 92 \text{ bits.} \)

(15)

Second, ev requires that all binding sites be restricted to the second half of the genome (positions 126 through 256), as they are in the target sequences. After randomly initializing 1.5 billion independent genomes, 187 outputs were found to have exactly 16 ones occurring along the second half of the genome. Repeating the above calculations results in the slightly tighter bound of

\( I_S < 90 \text{ bits.} \)

(16)

Search algorithms using the ev perceptron.

A search algorithm’s role is to extract active information from the knowledge sources. Some search procedures do this better than others. In the case of ev, sources of knowledge guiding the search include the Hamming oracle and the predisposition of the perceptron to generate outputs that favor the type of targets shown. We show here that the evolutionary algorithm originally used for ev performs worse than simple stochastic hill climbing.

Algorithm A1 (stochastic hill climbing). Stochastic hill climbing can be performed at the perceptron level involving all of the

all of the elements in \( S \) to be positive. Once the threshold operator is applied, all of the positive values become one. Likewise, the opposite polarity of the bias will push the outputs of the threshold to zero. Only with biases near zero will the resulting binary string be a nearly even mix of ones and zeros. Outcomes with a large number of ones and a few zeros, or few ones and many zeros, are therefore more probable outcomes of the perceptron.

No site a binding site. Analyzing the target sequence consisting of all zeros \( (i.e., \) no site is a binding site) illustrates the tendency for the perceptron to produce certain outcomes more frequently than others. To see this, we ran the perceptron in Figure 1 once using uniform random sampling with no iteration and assessed the Hamming distance between the perceptron output bits and the target sequence of all zeros. The process was repeated using freshly generated random numbers. Figure 2A shows a histogram of the Hamming distance between randomly generated perceptron outputs and the target sequence for ten million such trials when the target sequence is all zeros. Rather than dipping to nearly zero at the origin, the empirical probability of success is

\[
q = 0.00155.
\]

(13)

The exogenous information for an output of all zeros is therefore only \( I_S = \log_q(q) \approx 9 \text{ bits.} \)

12 The substantial decrease in difficulty (9 bits instead of the expected 256 bits) is due to active information introduced by the perceptron structure for an all zeros target. The reason, as we have shown, is that the ev perceptron is heavily predisposed for generating successful solutions for targets of the type shown in Equation 10 where a few ones are sprinkled in a sea of zeros.

A total of 261 nucleotides defines the perceptron structure in Figure 1, and the size of the search space is given by Equation 11. It follows that an astounding \( q|\Omega| \approx 2 \times 10^{154} \) genome sequences produce an output of all zeros.

The substantial decrease in difficulty (9 bits instead of the expected 256 bits) is due to active information introduced by the perceptron structure for an all zeros target. The reason, as we have shown, is that the ev perceptron is heavily predisposed for generating successful solutions for targets of the type shown in Equation 10 where a few ones are sprinkled in a sea of zeros.

A total of 261 nucleotides defines the perceptron structure in Figure 1, and the size of the search space is given by Equation 11. It follows that an astounding \( q|\Omega| \approx 2 \times 10^{154} \) genome sequences produce an output of all zeros.

Figure 2. Occurrence frequency of perceptron outputs. A: The normalized histogram of the errors in ev for ten million trials using an all ones target (red line), and ten million trials using an all zeros target (black line). The two histograms are nearly graphically indistinguishable. There are zero errors in 0.15% of the trials for both. B: The Gaussian distribution we expect from the Laplace-DeMoivre theorem. C: This curve, equal to the results in A divided by the results in B, reveals the degree to which the frequency of occurrence for output sequences has been amplified by ev’s perceptron. The range of all three plots is from zero to 256. The frequency of all zeros expected by a randomly chosen string of bits is multiplied by \( 1.8 \times 10^{74} \) by the perceptron. The frequency of occurrences of exactly 16 ones is amplified by a factor of \( 2.8 \times 10^{49} \).

13 We recalculated using \( \frac{1}{\pi} = \left( \frac{256}{16} \right) = 10^{25} \) to reflect the smaller number of potential binding site positions.

This is also the case for A VIDA software [15]. The evolutionary search algorithm used in the original paper was shown to extract information from the knowledge sources in the program much less efficiently than other search algorithms [18].
genome bases in ev. We initialized by choosing 261 genome bases at random, and then computed the error output (see Figure 1). One of the 261 bases was replaced at random and the error output was recomputed. If the error was the same or smaller, the change was kept. If not, the change was discarded and the process repeated. Simulation of \( K = 10,000 \) separate searches\(^{15} \) with \( Q_{\text{max}} = 100,000 \) and random initializations produced a 100% success rate. The average number of queries for success was \( Q = 10,601. \) According to Equation 8, the estimated normalized active information per mean query for algorithm \( A1 \) is therefore

\[
\frac{I_{\text{III}}}{Q_{\text{max}}} = 1.09 \times 10^{-4} \quad (17)
\]

Algorithm \( A2 \) (evolutionary perceptron inversion of ev). The original ev program \([16]\) used an evolutionary search algorithm for finding binding sites. The search was seeded with \( M = 64 \) randomly selected organisms and, in each iteration, two adjacent bits from a randomly chosen nucleotide in each organism were discarded, replacing them with two randomly chosen bits. (As before, the nucleotides are represented by two-bits.) Half of the mutated genomes with the highest fitness were then selected to be the parents of the next generation.\(^{17} \)

Using \( Q_{\text{max}} = 100,000 \) as the maximum query count per search, simulation of \( K = 10,000 \) separate searches using randomly generated initializations produced 9,115 successes. The average number of queries for success was, \( \langle Q \rangle = 63,568 \) or 993 generations. This compares to the single simulation result of 704 generations reported in the original ev paper \([16]\). Applying Equation 6, the estimated normalized active information per mean query for algorithm \( A2 \) thus gives

\[
\frac{I_{\text{III}}}{Q_{\text{max}}} = 1.55 \times 10^{-7} \quad (18)
\]

Comparing with Equation 17, we see that algorithm \( A1 \) is roughly 700 times more efficient than \( A2 \), the evolutionary algorithm used by ev.

The effect of differing mutation rates. The above results were obtained using a fixed mutation rate of one base change (two bits) per organism per generation for both \( A1 \) and \( A2 \). We further measured the search performance of the \( A2 \) strategy using several different mutation rates\(^{19} \). A comparison of success rate and mutation rate for the \( A2 \) algorithm is shown in Figure 3. The optimal mutation rate for \( A2 \) was found to be roughly 1.75 mutations per child,\(^{20} \) per generation, with the 1 mutation per child rate chosen by Schneider being slightly less efficient, but still within the range of workable mutation rates.

The effect of random uniform generated binding sites in stochastic hill climbing. We have shown the perceptron favors generation of strings of output zeros peppered with occasional ones, or ones peppered with zeros. The target sequences generated in the original ev program are the former. Further experiments demonstrated a severe performance decrease when the binding sites are chosen by a 50-50 coin flip. We simulated the ev search using randomly generated binding sites with each site along the second half of the genome (positions 126 through 256) having a 50% chance of being marked as a binding site. We measured the performance using mutation rates of 1, 1.5, 2, 4 and 8 mutations per child, per generation, with a query cutoff of \( Q_{\text{max}} = 100,000 \) queries for 10,000 runs each. The ev search was only successful for mutation rates of 1 and 1.5, failing to find the target in under 100,000 queries for all other mutation rates. A mutation rate of 1 mutation per child using the random 50-50 bindings resulted in a success rate of only 0.05 (compared to the prior success rate of 0.91), while a mutation rate of 1.5 mutations per child resulted in a success rate of 0.0003 (compared to the prior success rate of 0.99). Changing the binding sites pattern to an even mix of zeros and ones therefore severely hampers search performance.

CONCLUSIONS

The success of ev is largely due to active information introduced by the Hamming oracle and from the perceptron structure. It is not due to the evolutionary algorithm used to perform the search. Indeed, other algorithms are shown to mine active information more efficiently from the knowledge sources provided by ev \([13]\).

Schneider \([16]\) claims that ev demonstrates that naturally occurring genetic systems gain information by evolutionary processes and that “information gain can occur by punctuated equilibrium”. Our results show that, contrary to these claims, ev does not demonstrate “that biological information...can rapidly appear in genetic control systems subjected to replication, mutation, and selection” \([16]\). We show this by demonstrating that there are at least five sources of active information in ev.\(^{21} \)

1. The perceptron structure. The perceptron structure is predisposed to generating strings of ones sprinkled by zeros or strings of zeros sprinkled by ones. Since the binding site target is mostly zeros with a few ones, there is a greater

\( \text{Figure 3. Search success rate vs. mutation rate for algorithm } A2. \text{ Results are from a population size of 64 and a query cutoff of } Q_{\text{max}} = 100,000 \text{ queries. doi:10.5048/BIO-C.2010.3.F3} \)
predisposition to generate the target than if it were, for example, a set of ones and zeros produced by the flipping of a fair coin.

2. The Hamming Oracle [13]. When some offspring are correctly announced as more fit than others [27], external knowledge is being applied to the search and active information is introduced. As with the child’s game, we are being told with respect to the solution whether we are getting “colder” or “warmer”.

3. Repeated Queries. Two queries contain more information than one. Repeated queries can contribute active information [1,2,5].

4. Optimization by Mutation. This process discards mutations with low fitness and propagates those with high fitness. When the mutation rate is small, this process resembles a simple Markov birth process [27] that converges to the target [1,2,5].

5. Degree of Mutation. As seen in Figure 3, the degree of mutation for ev must be tuned to a band of workable values.

Our analysis highlights the importance of disclosing sources of knowledge in computer searches when measuring the ability of search mechanisms to generate novel information. As far as ev can be viewed as a model for biological processes in nature, it provides little evidence for the ability of a Darwinian search to generate new information. Rather, it demonstrates that preexisting sources of information can be re-used and exploited, with varying degrees of efficiency, by a suitably designed search process, biased computation structure, and tuned parameter set. This confirms that the conservation of information principle, as manifest in the No Free Lunch Theorems, is “very useful, especially in light of some of the sometimes-outrageous claims that had been made of specific optimization algorithms” [4].

Acknowledgements

The authors are appreciative of the detailed, insightful and useful comments made by one of the reviewers. We also are indebted to Ann Gauger for her detailed reading of the manuscript and numerous invaluable suggestions to improve the presentation.


Erratum

The section entitled *An upper bound on exogenous information* on page 3 is incorrect. It confuses the output of the perceptron with the input fed into the perceptron. The logic was intended to derive from assuming that a single genome produced the correct output, but actually proceeds as though all sequence with 16 ones have equal probability. If we make that assumption, the result of 92 bits is correct although it is an approximation not a bound. Furthermore, the result of 90 bits depends on the same assumption being made for a smaller set of output sequences. The difference in the bounds indicates that the assumption does not hold precisely, and in fact having all sixteen ones in the second half is 4 times more likely than the assumption predicts. The precise exogenous information could be either higher or lower than these approximations depending on the outworking of the perceptron. We offer 90 bits as a rough approximation of the exogenous information. The rest of the paper does not depend on this as a precise bound, and so is unaffected by this error.

We thank Robb for his attention to detail and bringing the error to our attention.