Abstract—The development of general edit metric decoders is a challenging problem, especially with the inclusion of additional biological restrictions that can occur when using error correcting codes in biological applications. Side effect machines (SEMs), an extension of finite state machines, can provide efficient decoding algorithms for such edit metric codes.

Several codes of varying lengths are used to study the effectiveness of evolutionary programming (EP) as a general approach for finding SEMs for edit metric decoding. Direct and fuzzy classification methods are compared while also changing some of the EP settings to observe how decoding accuracy is affected. Regardless of code length, the best results are found using the fuzzy classification methods. For codes of length 10, a maximum accuracy of up to 99.4% is achieved for distance 1 whereas distance 2 and 3 achieve up to 97.1% and 85.9%, respectively. The accuracy suffers for longer codes, as the maximum accuracies achieved by codes of length 14 were 92.4%, 85.7% and 69.2% for distance 1, 2, and 3 respectively. Additionally, the SEMs are examined for potential bloat by comparing the number of reachable states against the total number of states. Bloat is seen more in larger machines than it is in smaller machines. Furthermore, the results are analyzed to find potential trends and relationships among the parameters, with the most consistent trend being that, when allowed, the longer codes generally show a propensity for larger machines.

I. INTRODUCTION

An array of tools and methods have been developed for identifying genetic markers — DNA sequences that uniquely identify an organism or a trait. However, corruption of these markers is a common phenomenon that occurs due to a variety of reasons. When errors occur they need to be corrected to identify the original sequence. Unfortunately, the current set of tools used in biological applications is extremely limited.

One approach is to use error-correcting codes: the markers are stored as codewords and decoding can correct errors that may occur. Decoding is a well-studied problem across several scientific disciplines. However, the inclusion of biological restrictions makes it particularly challenging in biological applications. Several new methods have been proposed by researchers in the last few years. One of them, the side effect machine (SEM), used in conjunction with evolutionary programming (EP), has shown promising results in terms of decoding speed and accuracy.

This research uses EP to evolve SEMs for decoding a wider range of codes than in previous studies [11], [12], [23], [26].

It also performs in-depth analyses of the structures of the successful SEMs, especially with respect to the number of states and their connectivity to accurately identify and measure possible bloat in the SEMs. This study aims to establish a better realization of some of the key parameters involved in working towards improving the accuracy of the decoder.

II. BACKGROUND

Data transmitted over communication channels is subject to noise and thus prone to corruption. Such modifications during the course of transmission are called errors. Errors degrade the quality of communication and may modify the sender’s data to such an extent that no meaningful information can be retrieved by the receiver. Therefore, detection and correction of errors is of paramount importance to ensure reliable communication.

An error can be represented as the difference between the original message and the received message. The Hamming distance is the number of unmatched symbols between two words of the same length. An error correcting code transforms data such that errors introduced into it can be detected and corrected to a certain extent. Each of the original strings is called a codeword and the set of all codewords is called a code. Mathematically, an error correcting code is denoted as \( (n, M, d) \) where \( n \) is the length of the codewords, \( d \) is the minimum distance by which the codewords are separated, \( M \) is the total number of codewords the code contains, and \( q \) is the number of symbols that occur in the codewords.

Maximum likelihood decoding assumes that when a word is received, the codeword closest to it is the original word. Mathematically, an error correcting code can correct up to \( t = \lfloor (d - 1)/2 \rfloor \) errors [25]. For example, with minimum distance 7, an error correcting code can correct up to 3 errors.

Sequence tags [1] are relatively short DNA sequences that provide identifying information about an organism. These tags are unique and easily detectable in the genome by polymerase chain reaction (PCR). As such they serve as important elements in a genetic construct. The process of sequencing is prone to error: misreading a base, skipping a base, or reading a base that is not in the tag are common errors.

Upon observation, it can be seen that these are fundamentally similar to errors encountered during transmission over a noisy communication channel. This makes the correction schemes discussed in coding theory applicable in the field of genomics. If the tags are stored well separated from one
another, they can be used as codewords in a code that can correct such errors. Such a code should have \( q = 4 \), to correspond to the 4 symbols A, C, G and T in DNA.

While Hamming distance is appropriate for dealing with substitution errors, it is not useful for detecting insertions and deletions. Instead, edit distance is an appropriate choice. The edit distance between two words is the minimum number of insertions, deletions or substitutions required to change one word into another. Codes that use edit distance to measure the distance between codewords are called edit metric codes.

The Hamming distance between \( x = 01010101 \) and \( y = 10100101 \) is 4 as all of the first 4 symbols differ in \( x \) and \( y \). The edit distance is 2 as \( x \) can be transformed into \( y \) by deleting the first 0 from \( x \) and inserting a 0 at the 4th bit.

### III. Related Work

In [3] edit metric codes were created for use as embeddable markers for cDNA libraries. Several later studies [2], [4], [5], [24] used a variety of approaches, including evolutionary algorithms, other metaheuristics, and combinatorial searches to create codes useful for biological applications.

Given a corrupted word, decoding is the process of finding the correct codeword. The process of decoding may vary based on code properties which are usually dictated by applications.

Marker codes [29] and Watermark codes [14] each use concatenation of codes to allow for correction of insertions and deletions. These codes are generally not suitable in DNA sequencing problems as they maintain a specific structure [9]. Depending on the problem, there may be several constraints, e.g. GC content restrictions or disallowed substrings [8] [30] [31]. Meanwhile, general edit metric decoding uses an \( O(n^2) \) algorithm, which calls for more efficient alternatives.

In an effort to improve performance, several previous studies [11], [12], [26] used various forms of evolutionary computation to evolve side effect machines (SEMs) for decoding. The SEMs use Euclidean distance as a proxy for edit distance, reducing the complexity from \( O(n^2) \) to \( O(n) \) but making the decoding process an approximation. All of these studies considered codes of the same length (12) and attempted to evolve good SEMs of a given fixed size. These studies found that decoding accuracy was strongly tied to the size of the SEM. In general, larger machines had improved accuracy, although this varied depending on the representation for the genetic algorithm, and accuracy plateaued after a given point.

In [10] SEMs were examined to estimate their placement within the Chomsky hierarchy; this work also provided a mathematical relationship between the length of an error correcting code and the number of states needed for decoding.

The latest study [23] used evolutionary programming (EP) for the creation of SEM decoders. The ability to modify the size of the SEMs using mutation operations that add or remove states during evolution added a new dimension: the machines were not restricted to a fixed size, but allowed to freely evolve (within a given range) to a size that would produce the best results. Although larger machines performed better than smaller ones, the fitness did not improve significantly after

14 to 16 states. Moreover, although a preference for larger machines was observed, their association to better fitness could not be confirmed due to potential bloat.

These studies clearly demonstrate the importance of studying the different aspects of SEMs for decoding. Furthermore, analyzing the best machines to determine the exact count of the used states to simplify the resulting machines has yet to be attempted. Importantly, all of the previous studies used codes of the same length. Therefore, it is a definite requirement to look at codes of other lengths to find out the effect of SEMs in different problem spaces. This study seeks to fill these missing areas while attempting to improve the decoding accuracy of the SEMs in the process.

### IV. Side Effect Machines

A side effect machine [6] is an extension of deterministic finite automata. The machine consists of a set of states, one of which is preselected as the start state. Each state has a counter that is initialized to zero. The machine takes a string as input and reads its symbols in sequence. Each symbol triggers a state transition. As a state is visited, its counter is incremented.

The counter values are stored in a classification vector [23]. Figure 1 shows a simple SEM with four states 0, 1, 2, and 3. State 0 is the start state as denoted by the double circle. The sequence of ACTGCGCA produces the transition path 1, 1, 2, 3, 2, 3, 0 and yields the classification vector \( c = (1, 2, 3, 2) \).

A. Decoding using Side Effect Machines

A SEM can be used to quantify how different two words are from each other. The words can be compared by running their symbols through the state machine and finding the Euclidean distance between the classification vectors produced.

This principle can be used for decoding using a SEM. The idea is to compare an error pattern with each codeword to find which one it resembles most closely. The error pattern and the codewords are run through the SEM to compute their classification vectors. Next, the Euclidean distances between the classification vector of the error pattern and those of the codewords are calculated. The codeword associated with the smallest distance is considered to be the original word.

Table 1 shows an example of decoding. Suppose a code has 3 codewords \( w_1 \), \( w_2 \) and \( w_3 \). If the word \( x \) is received, error
correction can be done by finding the smallest edit distance to all codewords. It is found that \( x \) has the smallest edit distance to \( w_1 \). A SEM (Figure 1) can also be used. First, the SEM finds the classification vectors of all codewords. Upon receiving \( x \), the SEM finds its classification vector. We then compute the Euclidean distances between the classification vector of \( x \) and those of the codewords. In this example it is found that the classification vectors of \( x \) and \( w_1 \) have the smallest Euclidean distance, leading to a successful decoding.

This technique reduces the complexity of decoding, because Euclidean distance is an \( O(n) \) operation in comparison to \( O(n^2) \) for edit distance. However, it does not always accurately decode. Some SEMs perform better than others for a given code. Furthermore, two different sequences can obtain the same classification vectors, or their classification vectors could have the same Euclidean distance from another. This makes the decoding process ambiguous.

V. EVOLUTIONARY PROGRAMMING

Evolutionary programming, first conceived in [21], has been effectively used for numerical and combined optimization challenges [17]–[19] and is also suitable for problems for which there exist many locally optimal solutions [32]. It was first used for solutions represented as finite state machines and later enhanced to use other representations. Biological applications of EP include [13], [16], [20], [22]. It has also been used in other disciplines, such as in problems relating to networks and power planning [15], [27].

In the current study, the solutions are represented as SEMs and EP provides the ability to mutate them by adding or removing states and modifying transitions.

A. Representation

The size of a SEM is equal to the number of states. Each state has 4 output transitions, for the 4 DNA symbols. When a SEM reads an input sequence, it starts from an initial state. In this study, state 0 is the initial state for every machine, but can be replaced with another state through mutation. Some states may not be reachable from the initial state. These states, acting as bloat, stay in the machine without affecting the fitness.

B. Initialization

The candidate solutions are initialized randomly, with each having a random size from within the allowed range.

<table>
<thead>
<tr>
<th>Word</th>
<th>Classification Vector</th>
<th>Edit distance to ( x )</th>
<th>Euclidean distance to ( x )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( w_1 )</td>
<td>ACTGCGA</td>
<td>(1,2,3,2)</td>
<td>1</td>
</tr>
<tr>
<td>( w_2 )</td>
<td>CCTAGA</td>
<td>(4,1,2,1)</td>
<td>5</td>
</tr>
<tr>
<td>( w_3 )</td>
<td>ATCGACGT</td>
<td>(2,2,3,1)</td>
<td>3</td>
</tr>
<tr>
<td>( x )</td>
<td>ACTGCCGT</td>
<td>(0,3,3,2)</td>
<td>-</td>
</tr>
</tbody>
</table>

C. Bout System

The bout system is a process similar to tournament selection in genetic algorithms [28]. Each SEM in the population gets an individual score from a series of bouts, that can later be sorted to select the best for the next generation.

D. Mutation

Mutation is a unary operation to modify a SEM. Four different types of mutation are used in this study:

1) Change a Transition: randomly selects a transition and changes its ending state to another state.

2) Add a State: adds a new state into the machine at a random position, as long as the size of the resulting machine stays within the upper bound. Outgoing transitions are created to randomly selected states. Note that the newly added state may be unreachable from the initial state, thus relying on future mutations to alter transitions in ways that allow the new state to be reached.

3) Delete a State: deletes a state from the machine at a random position as long as the size of the resulting machine stays within the lower bound. The input edges from other states to the removed state are connected to its previous state. If the state to be deleted is the initial state, then the state that follows it numerically is the new initial state.

4) Change Initial State: randomly selects a state to be the new initial state of the machine.

VI. METHODOLOGY

Nine codes are used in this study. While previous studies [11], [12], [23], [26] dealt only with codes of length 12, we vary the length of the codes to see the effectiveness of the approach for different lengths. Three codes of each of lengths 10, 12 and 14 are used. The codes of length 12 are taken from [23] while all other codes are new. The minimum distance of the codes is held constant at 7.

A. Creation of Error Patterns

As each code has minimum distance 7 they can correct up to 3 errors. The training and verification sets each consist of \( n \) error patterns for each codeword and for each distance 1, 2 and 3, where \( n \) is the length of the code. Table II summarizes information on all codes.

The idea behind creating an error pattern \( e \) from a codeword \( c \) is to check if a SEM correctly decodes it, i.e. correctly classifies \( e \) as being closest to \( c \) in comparison to all other codewords. If a SEM correctly decodes all error patterns in a given set then it obtains a perfect score, equal to the total number of error patterns.

B. Direct Classification

Direct classification employs Euclidean distance as specified in Section IV-A. Euclidean distance is used as a cheap proxy for edit distance and its affinity to edit distance depends heavily on the SEM. This is why EP is used to evolve SEMs that provide good accuracy in terms of error correction.
TABLE II
DATASET, SPECIFYING SIZE (NUMBER OF CODEWORDS), LENGTH, AND THE NUMBER OF ERROR PATTERNS IN EACH SET FOR EACH CODE.

<table>
<thead>
<tr>
<th>Code</th>
<th>Size</th>
<th>Length</th>
<th>Error patterns at each of distance 1, 2, 3</th>
<th>Total error patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code17-1</td>
<td>17</td>
<td>10</td>
<td>170</td>
<td>510</td>
</tr>
<tr>
<td>Code17-2</td>
<td>17</td>
<td>10</td>
<td>170</td>
<td>510</td>
</tr>
<tr>
<td>Code18</td>
<td>18</td>
<td>10</td>
<td>180</td>
<td>540</td>
</tr>
<tr>
<td>Code55</td>
<td>55</td>
<td>12</td>
<td>660</td>
<td>1980</td>
</tr>
<tr>
<td>Code60-1</td>
<td>60</td>
<td>12</td>
<td>720</td>
<td>2160</td>
</tr>
<tr>
<td>Code60-2</td>
<td>60</td>
<td>12</td>
<td>720</td>
<td>2160</td>
</tr>
<tr>
<td>Code201</td>
<td>201</td>
<td>14</td>
<td>2870</td>
<td>8442</td>
</tr>
<tr>
<td>Code205-1</td>
<td>205</td>
<td>14</td>
<td>2870</td>
<td>8610</td>
</tr>
<tr>
<td>Code205-2</td>
<td>205</td>
<td>14</td>
<td>2870</td>
<td>8610</td>
</tr>
</tbody>
</table>

TABLE III
PARAMETER VALUES FOR FOUR SETS OF EXPERIMENTS

<table>
<thead>
<tr>
<th>Experiment</th>
<th>E1</th>
<th>E2</th>
<th>E3</th>
<th>E4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population Size</td>
<td>300</td>
<td>300</td>
<td>300</td>
<td>300</td>
</tr>
<tr>
<td>Number of Generations</td>
<td>1250</td>
<td>1250</td>
<td>1250</td>
<td>1250</td>
</tr>
<tr>
<td>Bout Size</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Probability of Changing a Transition</td>
<td>0.6</td>
<td>0.75</td>
<td>0.8</td>
<td>0.85</td>
</tr>
<tr>
<td>Probability of Changing Initial State</td>
<td>0.1</td>
<td>0.05</td>
<td>0.1</td>
<td>0.05</td>
</tr>
<tr>
<td>Probability of Adding a State</td>
<td>0.15</td>
<td>0.1</td>
<td>0.05</td>
<td>0.05</td>
</tr>
<tr>
<td>Probability of Removing a State</td>
<td>0.15</td>
<td>0.1</td>
<td>0.05</td>
<td>0.05</td>
</tr>
</tbody>
</table>

C. Fitness
Fitness, which should be maximized, is a simple count of the number of error patterns accurately decoded by a SEM. An error pattern is considered decoded when it is found to be closer to its original codeword than to the other codewords, in terms of Euclidean distances of classification vectors (see Section IV-A). The fitness score is incremented if the distance for the original codeword is the smallest amongst all.

D. Parameter Settings
Four parameter settings are used on all datasets, as shown in Table III. The population size, number of generations, and bout size are the same as in [23]. Although initial tests were done with different values for these parameters, no significant improvements were seen. Parameter settings E1 and E2 were first used in [23], while settings E3 and E4 are new.

1) Range of States: In earlier studies [11], [12], [26], the effectiveness of a SEM was found to be related to its size. With EP, SEMs are allowed to shrink or grow within a range. In [23] this range was 4–18; it generally showed higher accuracy for larger machines but the best machines were prevalent in sub-ranges of sizes rather than being inclined to just one size, and the sub-ranges varied from code to code. In the current study, we use 8 smaller ranges to investigate whether some codes react better to certain ranges and understand how the machines evolve within them. The ranges are: 4–6, 4–8, 6–12, 6–18, 8–14, 8–18, 10–16 and 14–18.

2) Count of Reachable States: Not every state is guaranteed to be visited when a pattern is passed through the machine. Additionally, at the time of reproduction, mutation may cause a state which was visited at least once in the parent machine to never be visited in the child machine. A depth-first search is performed to determine which states are reachable or unreachable. The unreachable states can be purged to simplify a machine without altering its behaviour and thus help to determine the true machine size required.

E. Algorithm to Generate SEMs using EP
At each generation the population of candidates acts as parents for the next. Each parent is mutated as described in Section V-D to produce a child. The type of mutation applied is determined by the probabilities in Table III. Reproduction doubles the population size. All of these candidates are then given fitness scores and ranked using the bout system and the top half is selected to form the next generation.

F. Fuzzy Classification
Fuzzy classification tries to improve upon direct classification by checking the most probable error codes first. It creates a list of all codewords in ascending order of Euclidean distance of their classification vectors to the error pattern's classification vector. In this order, the codewords are then compared with the received error pattern using edit distance until a match is found. A tolerance can be used to narrow the search by filtering out codewords with Euclidean distance above the tolerance. The algorithm stops when either the correct codeword is found or no codewords remain within the tolerance. Fuzzy classification can also be used to identify when decoding fails, i.e., no codewords within the given tolerance have edit distance $\leq (d-1)/2$ from the error pattern.

VII. RESULTS AND ANALYSIS
The summary statistics of maximum fitness accuracy for each code, along with their corresponding range and experiment number, are presented in Table IV. The full tables for these results are presented in [7]. In general, codes of smaller length have better maximum accuracy than longer codes, in both direct classification and fuzzy classification. The only exception is code18 in direct-verification, which has lower accuracy than code55. In general, no particular range or experiment setting outperforms the others. Deeper analyses are presented in the following subsections.

A. Number of States
Previous studies [11], [23] found an inclination towards larger machines with respect to decoding accuracy, a trend that is analyzed further in this study. The extent of this impact appears to vary based on a number of other factors such as the length of the code, the classification method and the dataset used i.e. direct vs fuzzy and training vs verification.

To understand this relationship, the median accuracy rate was plotted against the machine size (number of visited states) of all the best machines found across mutation types and ranges. Figure 2 shows this graph for direct training, for all codes in the dataset. It demonstrates that larger machines show a higher and steady rate of improvement for longer codes. This is evident in the steady incline that the codes of length 14 (code201, code205-1, and code205-2) experience as the machine size grows from 8 to 18. On the other hand, the decoding
accuracy for the smaller length codes starts plateauing or even declining beyond a certain machine size, namely 13 and 15 respectively for codewords of length 10 (code17-1, code17-2, code18) and 12 (code55, code60-1, code60-2). In contrast to the gradual improvement noticed across codes, a slight dip was noticed for code18 and code201 just as they were approaching the maximum size, which could be an indication of a possible plateau once the machine grows past a certain size. However, the evidence is somewhat inconsistent.

It should be noted that the fitness did not plunge from machine size 6 to 7, even though it appears that way. The graph includes machines across all ranges. For some of the larger codes, no best machines with 5 or 7 states were created, which formed the peaks at the left of the graphs. A possible explanation is that the larger codes tend to produce larger machines. Therefore, for some of the larger codes, range 4 to 6 only produced machines with 6 states whereas ranges 4 to 8 and 6 to 12 produced machines with 8 states or more.

The improvement in accuracy is negligible with fuzzy classification for lengths 10 and 12 after the machine reaches more than 8 states. The accuracy for codes of length 14, however, slowly but steadily improves as the machine grows all the way to the maximum size allowed. These results support the observation in [23] that machine sizes have little impact on fitness using fuzzy classification once the machines go past a certain number of states.

B. Different Lengths of Codes

Figure 2 also offers insight into the relative performance of SEMs for codes of different lengths. For any machine size, the SEMs for codes of the same length achieve very similar decoding accuracy. This is noticed for all lengths studied, with the exception of code18 which does not have the same level of accuracy as code17-1 and code17-2 despite having the same length. This is likely due to the fact that the higher number of codewords in code18 means that they are packed more tightly together, i.e. it is harder to distinguish them from one another due to their closer proximity.

This behaviour can be further analyzed by examining Figure 3, which is a heatmap representation of the overall accuracy achieved by each range for each code grouped by mutation techniques during direct training. Each block represents the median fitness across 30 experiments, expressed as a percentage of the maximum fitness. As evident by the colours of their respective blocks, machines for codes of length 10 (code17-1, code17-2, code18) and length 14 (code201, code205-1, and code205-2) produce the highest and lowest accuracy scores respectively, while length 12 codes (code55-1, code55-2, code60) are in the middle. This behaviour is consistent with the grouping seen in Figure 2 and reinforces the notion that codes of the same length react in a similar manner to the SEMs. Interestingly, for fuzzy training SEMs for codes of length 12 perform better than those for length 10.

C. Mutation

As seen in Figure 2, the overall fitness improves with increased machine size before potentially plateauing or dipping. To understand how the different mutation strategies influence final machine size, the total number of machines generated for each machine size was studied for each experiment. Experiments that promote machine growth, marked by higher weights for addition, produce larger machines more often than their counterparts. When compared in wider ranges (that can accommodate reasonable growth), E1 consistently produces more machines with larger sizes than E2, E3 and E4, and the same trend is seen when comparing E2 with E3 and E4.
Fig. 3. Representation of the overall accuracy achieved by each range for each code grouped by mutation techniques during direct training. Each block represents the median fitness across 30 experiments.

However, the impact of such growth on the overall fitness was not conclusive. Figure 4 can be examined to understand the role the mutation settings play on the overall evolution process and in turn the accuracy. This figure shows the accuracy for one code of each length using direct classification. Each subplot, further divided by ranges, shows the distribution of final fitness for each experiment over 30 runs. Focusing on any given range in any given code, no significant difference was noticed among the four experiments in terms of accuracy. Similar results are seen for all other codes, and also for direct verification, fuzzy training and fuzzy verification.

D. Total States vs Reachable States

In [23] it was hypothesized that bloat may manifest itself in the SEMs due to the presence of unused states, and it recommended that future work should investigate this further to find the reachable state count. This information can help simplify the machines by excluding unreachable states. The bubble chart shown in Figure 5 plots the final SEM machine size (total states) against the number of states that were visited. Samples across runs for all codes and mutation types were combined and grouped by range. Machines with one or more unused states appear below the $x = y$ line, while those in which all states are visited appear on it. The size of the bubble reflects the size of the data points, i.e. the number of machines that appear at a given coordinate. This graph provides insight into the effectiveness of the mutation strategies used — whether they are influencing the machines to evolve towards a truly good solution or simply bloating the machines without improving their fitness.

Fig. 4. Decoding accuracy of SEMs on code17-1, code55 and code201 with direct classification on training dataset across the four experiments.

It can be seen that a fair number of SEMs feature one or more unused states. This is most noticeable in larger machines and rare in smaller machines. This is evident by the higher occurrences of such machines in ranges with the upper bound of 18 (6–18, 8–18, and 14–18) than others. All states were visited in machines generated with ranges 4–6 and 4–8. This can be explained by the lack of opportunity a machine has to grow while being constricted by a smaller bound.

This also exposes a potential flaw in the mutation algorithm, where growth in number of states does not always lead to improved fitness, but instead adds unnecessary bulk. This is due to the fact that when adding a new state, that state may not be reachable from the start state due to a lack of appropriate transitions into it.

E. Ranges of Machine Size

In [23] the machine size had a range of 4–18 states. The distribution of final machine sizes was examined, showing that the sizes of the SEMs generated were somewhat uniformly dispersed from 9–18 states with the median varying between 11 and 13.5 states. Further investigation was proposed to determine if a smaller range would minimize the deviation and encourage the machines to move towards a particular size. The current study further examines this range by splitting it into smaller subranges (sometimes overlapping) to see if any allows faster convergence towards the best machines.

Figure 3 can be analyzed to see the effectiveness of different ranges on different codes. When compared to their larger counterparts, the two smallest ranges (4–6 and 4–8) prove...
to be much less effective. The general trend noticed is that ranges with higher maximums tend to produce SEMs with better decoding ability. However, the improvements are not too significant and there are some inconsistencies between adjacent or overlapping ranges, e.g., range 10–16 produces slightly better accuracy than 6–18 for code60-2 and code205-2 in subplot E3. For direct training, range 14–18 can be identified as the overall best as it produces good accuracy across codes. Also, it can be seen in Figure 4 that for the longer codes, range 14–18 had consistently high accuracy. Range 6–18 performed well across classification methods (direct, fuzzy) and datasets (training, verification). This possibly suggests that a wider range with a high upper bound may be suitable if it must be used for different code lengths and classification methods. However, this is not conclusive for any specific code.

F. Error Correction Capability on Different Distances

The error correcting ability depends on the edit distance between the codeword and the error pattern. Table V shows the maximum accuracy on three different distances for both direct and fuzzy classification. As expected, the accuracy of the machines decreases as the distance increases. This is also shown in previous studies [11], [12], [23], [26] in several ways. This included an investigation of the effectiveness of SEMs for decoding codes of different lengths and for more ranges of machine size.

Previous studies [11], [23] observed a preference for SEMs with a higher number of states, a trend also witnessed in this study. The propensity for larger machines is stronger for longer codes. However, the fitness saturates once a certain number of states is reached and this point appears to depend on the length of the code. Future work can include finding a good upper bound for the number of states for longer codes.

Fuzzy classification greatly improves accuracy compared to direct classification, especially for error patterns with larger edit distance. This is consistent with observations in previous studies. Note that 100% accuracy is possible with a very high tolerance value, as then the fuzzy method essentially allows all codewords to be checked using edit distance. However, this takes away from the purpose of using SEMs to reduce the time required for decoding. Possible future work includes a study of appropriate tolerance.

VIII. CONCLUSION AND FUTURE WORK

This study extended the scope of earlier studies that examined SEMs as edit metric decoders [11], [12], [23], [26] in several ways. This included an investigation of the effectiveness of SEMs for decoding codes of different lengths and for more ranges of machine size.

The error correcting ability depends on the edit distance between the codeword and the error pattern. Table V shows the maximum accuracy on three different distances for both direct and fuzzy classification. As expected, the accuracy of the machines decreases as the distance increases. This is also shown in previous studies [11], [12], [23], [26].

As an example to show the differences for distances 1, 2 and 3, see Figure 6. The violin plot is colour coded based on the classification method (direct vs fuzzy) and the dataset (training vs verification) used.
TABLE V

<table>
<thead>
<tr>
<th>Code</th>
<th>Max Acc (Direct-training)</th>
<th>Max Acc (Direct-verification)</th>
<th>Max Acc (Fuzzy-training)</th>
<th>Max Acc (Fuzzy-verification)</th>
</tr>
</thead>
<tbody>
<tr>
<td>code17-1</td>
<td>98.2</td>
<td>94.7</td>
<td>98.8</td>
<td>94.1</td>
</tr>
<tr>
<td>code17-2</td>
<td>97.6</td>
<td>95.3</td>
<td>99.4</td>
<td>97.1</td>
</tr>
<tr>
<td>code18</td>
<td>96.1</td>
<td>95.0</td>
<td>97.2</td>
<td>91.1</td>
</tr>
<tr>
<td>code55</td>
<td>92.4</td>
<td>91.7</td>
<td>95.5</td>
<td>90.9</td>
</tr>
<tr>
<td>code60-1</td>
<td>91.5</td>
<td>90.0</td>
<td>94.6</td>
<td>90.3</td>
</tr>
<tr>
<td>code60-2</td>
<td>91.0</td>
<td>90.3</td>
<td>95.6</td>
<td>90.4</td>
</tr>
<tr>
<td>code201</td>
<td>86.5</td>
<td>85.5</td>
<td>91.9</td>
<td>83.4</td>
</tr>
<tr>
<td>code205-1</td>
<td>86.9</td>
<td>85.6</td>
<td>91.9</td>
<td>83.6</td>
</tr>
<tr>
<td>code205-2</td>
<td>87.7</td>
<td>86.3</td>
<td>92.4</td>
<td>85.7</td>
</tr>
</tbody>
</table>

In progressing through generations the population loses diversity, hindering evolution towards a consistent well-balanced machine. This trend is stronger for codes of smaller length, with duplicate machines appearing sooner than for longer ones. This is expected due to the smaller search space. Future work could incorporate ways to increase diversity in the population.

The larger search space for longer codes hinders the ability of mutation to find a compact SEM. Future work can include experimenting with various EP settings, and examination of codes with different minimum distances. The current study considers a variety of rates for the different types of mutation. Further study of mutation types and rates may prove valuable.

Another important contribution is the examination of the connectivity of the best machines by finding the number of reachable states. This knowledge can help simplify the machines by excluding unvisited states. It also conclusively demonstrates the bloat that was hypothesized in [23].

REFERENCES