2023 IEEE Symposium Series on Computational Intelligence (SSCI)
Mexico City, Mexico. December 5-8, 2023

# What Drives Evolution of Self-Driving Automata?

Michael Dubé
Mathematics and Statistics
University of Guelph
Guelph, ON, Canada
mdube04@uoguelph.ca

Kevin Olenic
Computer Science
Brock University
St. Catharines, ON, Canada
ko19af@brocku.ca

Sheridan Houghten
Computer Science
Brock University
St. Catharines, ON, Canada
shoughten@brocku.ca

***Index Terms*—evolutionary algorithm, self-driving automata, sequence matching, mutation, diversity**

***Abstract*—*Self-Driving Automata* (SDAs) are variations on finite automata that both read and output symbols. They are versatile and practical when used for the generation of data for a variety of problems. In this study, we examine several questions regarding their operation, using *sequence matching* as a test problem in the analysis. We present a new mutation operator and four dynamic mutation adjusters. We analyze these, along with crossover, for their ability to solve the problem and their relative ability to improve the population; in all of these, we also examine population diversity over time. We find that using mutation that implements a static quantity of changes outperforms one with dynamic changes. Further, while population diversity does decrease somewhat, evolution is still possible.**

## I. Introduction

A *self-driving automaton (SDA)* is a recently-introduced construct capable of generating data. SDAs are variations of finite state automata, such that when it reads symbols transitions occur from one state to another, with each generating response symbol(s). These response symbols are not just fed back into the automaton as the subsequent input but also the output. In this manner, the SDA generates an infinite series of characters usable as data for numerous problems. In conjunction with evolutionary algorithms, SDAs have been applied to generating biological sequences [4], contact networks for epidemic modelling [1] [6] [7], and dungeon-level maps for games [5]. In each case, the fitness of a given SDA relates to how well the data it generates satisfies the given problem-specific conditions. In all problems considered to date, SDAs demonstrated an ability to be versatile and practical constructs producing high fitness results. However, because they are so new, many questions exist concerning the genetic operators and their effect on evolution and population diversity.

To explore these questions we will be using sequence matching as a test problem in this paper. Alone, this is a stepping stone towards investigating the ability to use SDAs and evolutionary algorithms to discover potential patterns in a class of proteins known as *dehydrins*, stress proteins in plants that play a crucial role in surviving conditions which can lead to dehydration [4] [17]. Gaining a better understanding of SDAs is necessary to achieve this grander goal, and also helps to inform research on other problems. This involves investigating the dynamics of a population of SDAs undergoing evolution.

To develop this knowledge we use an evolutionary algorithm to evolve SDAs, investigating a new crossover operator and two types of mutation that incorporate both static and several dynamic mutation adjusters. We analyze the distribution of potential fitness values of the children compared to their parents, and examine the population diversity throughout evolution.

This research was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC). It also used the facilities of SHARCNET and the Digital Research Alliance of Canada.
Code for this research is available at tinyurl.com/SSCI23.

## II. Background

### A. Self-Driving Automata (SDAs)

SDAs are modified *finite state machines* [8] that are inspired by *Kolakoski sequences* [9] and use the Mealy architecture for transitions between states [14]. The number of symbols in the alphabet determines both the number of output symbols and the number of transitions emanating from each state, each of which points to another state or itself. Each symbol of output is also added to an input queue which drives future transitions, necessitating congruence between the size of the alphabet and the number of transitions from each state because each transition is triggered by one of the symbols in the alphabet.

Each SDA has an initial state $S_0$ and an initial symbol $c_0$ that triggers the first transition; this initiates the generation of output. Each transition has a response of length one or two, with equal probability, comprised of symbols in the alphabet. Since a single symbol triggers a transition, and responses are one or two symbols, the SDA can generate characters indefinitely, for further details see [4]. An SDA can be used as the representation in an evolutionary algorithm so long as one can convert a series of symbols (i.e. as will be generated by the SDA) into a solution to the problem under investigation.

The test problem utilized for the current study is sequence matching, meaning output from the SDAs is compared to a given target sequence. For SDAs, crossover entails exchanging, between two parents, the transitions and responses associated with a set of states, while mutation randomly modifies the transitions and/or responses of one or more states. The specific crossover and mutation operators analyzed in this study are described in detail in Section III.

While previous work on evolving SDAs for various problems is mentioned in Section I, we now briefly mention some related research into evolving similar structures. In [11] [12] evolutionary algorithms were used to evolve finite automata

using a fitness measure based on the proportion of correctly-classified strings. Evolutionary algorithms that incorporated testing, validation, and verification were applied in [19] to evolve finite state machines' ability to control elevator doors. In [3], evolutionary algorithms evolved finite state classifiers to classify polymerase chain reaction primers. Used to evolve finite state machines for the Tartarus problem, the evolutionary algorithm in [16] adapted the mutation rate during evolution; in the current study we also adapt mutation using different methods (see Section III-B).

### B. Test Problem and Analysis

For our analysis, we use *sequence matching* as a test problem. The goal is to generate a sequence $S$ that matches, as closely as possible, a given target sequence $T$, where $S$ and $T$ have the same length $n$. A description of the problem is in [4], in which it was used for matching DNA sequences. In this paper we use the first two sequences from [4], shown in Figure 1. Future analysis should include additional sequences. The fitness function, Algorithm 1, is based on maximizing the number of matches, symbol-by-symbol; a perfect match between $S$ and $T$ has fitness equal to $n$.

**Algorithm 1** Sequence Matching Fitness

```
function SequenceMatches(S, T)
    m ← 0
    for i = 1 to n do
        if S[i] = T[i] then
            m ← m + 1
    return m
```

We examine several aspects in our analysis of the evolutionary algorithm used to evolve SDAs for sequence matching. With the states, transitions, and responses of an SDA all intrinsically linked, it is vital to examine how exchanging information between two SDAs may or may not create improvements. For example, [2] shows a situation in which crossover was routinely *destructive*, with a mutation-only strategy being far more effective. We also analyze multiple varieties of mutation and ascertain which operators tend to lead to the most improvement. In all of this, we also study the *diversity* of the population, which has long been known as crucial in helping to prevent premature convergence [13] [18].

We note that the fitness function for sequence matching is directly related to the output of the SDA, allowing for close evaluation of the SDA and genetic operators. In some applications, the fitness functions may have a much less direct relationship, thus requiring further analysis.

## III. Methodology

The steady-state evolutionary algorithm starts by randomly generating an initial population of SDAs having $N_s = 20$ states. The population undergoes a maximum of 10 million mating events with a statistical output generated every 10 000 mating events (the *reporting interval*). Each mating event involves *selection*, *crossover*, and *mutation*. Evolution terminates when the best population fitness value does not improve for 50 reporting intervals. The experiment repeats for a total of 50 runs. Size-seven tournament selection is used; randomly selecting seven SDAs then sorting them according to their fitness. The two with the best fitness are copied as children. Next, the copies have a 50% chance of undergoing crossover (see Section III-A), followed by a 100% chance of mutation (see Section III-B), with one exception: if crossover improves fitness over the current population's best fitness and then applying mutation would worsen fitness then mutation is not applied. Following this step, the children replace the two members of the tournament with the lowest fitness.

To increase the diversity of the population, we use *culling* as follows: every $CF$ reporting intervals, $CR\%$ of the population is discarded and randomly regenerated. Culling was shown to improve results in [4], with the best results achieved through random culling of $CR = 25\%$ of the population every $CF = 4$ reporting intervals. For our experiments, we use a culling rate of $CR = 25\%$, while testing both $CF = 1$ and $CF = 5$.

### A. Crossover

*Two-point crossover* on two SDAs starts by randomly selecting two distinct crossover points $cp_1$ and $cp_2$ in the interval $[0, N_s - 1]$. All transitions and responses within the interval $[cp_1, cp_2]$ are swapped between the SDAs; if $cp_1 = 0$ then the initial characters of the SDAs are swapped. In *one-state crossover*, a state is randomly selected, and the two SDAs swap transitions and responses for this state. Preliminary investigation compared these two forms of crossover, and multiple crossover rates. As no significant difference was observed in these investigations, we selected two-point crossover with a crossover rate of 50% for all further experiments.

### B. Mutation

Previously in [4] a program parameter $M$ specified the maximum number of mutations, and a single mutation operator was used which was applied $m \in [1, M]$ times on the two children produced from a crossover event, using a uniform distribution to determine $m$. Each application of mutation would either re-select the SDA's initial symbol $c_0$, randomly re-assign one transition, or randomly regenerate one response.

In the current work each mutation step begins with a 10% chance of changing the initial symbol, then $m_t$ transition mutations and $m_r$ response mutations. We performed a preliminary investigation of mutation rates of 50% and 100%; as 100% yielded better results this was used for all further experiments.

The values of $m_t$ and $m_r$ are given as input parameters. These either remain *static* during the program's lifetime, or are adjusted dynamically during evolution. Four different *dynamic* versions are defined based on methods $d_1$, $d_2$, $d_3$, and $d_4$ that all adjust $m_t$ and $m_r$ at the end of each reporting interval. For all adjusters, $m_t$ and $m_r$ are restricted to the range $[0, 50]$. Method $d_1$ simultaneously raises $m_t$ by one and lowers $m_r$ by one. Method $d_2$ mirrors $d_1$, instead lowering $m_t$ by one and raising $m_r$ by one. Method $d_3$ compares the best fitness between the previous and current report; if there is

| ID | Length | Sequence |
|---|---|---|
| 0 | 57 | ATGGGACGCAAGGACGAGCAGAAGCAAACGAGCGCCACAAGCACGCCGGGGCAGGGG |
| 1 | 87 | GGAAGTAAGGATAAGGACAAGGAGGAACACAAGGAAACCACCACCACCACCGCCACCGCCACGGCGGAAGAATCCCACCACGAACAC |

Fig. 1: Target sequences used for sequence matching (see [4]).

no improvement then $m_t$ and $m_r$ both increase by two, else both decrease by two. Finally, method $d_4$ increases both $m_t$ and $m_r$ by the number of reporting intervals since the last improvement in the population's best fitness; if improvement occurs, then $m_t$ and $m_r$ are reset to their initial values.

### C. Investigation into Effect of Genetic Operators

In [4] two population sizes were examined, with a population of 500 SDAs producing the best results. In the current work we again use a population size of 500. To allow investigation into the effect of the crossover and mutation operators throughout evolution, we also use a smaller population size of 50 as follows. For crossover, every SDA was paired with every other SDA in the population and then underwent 100 two-point crossover events on separate copies of these parents. This resulted in 9800 children for every member of the population, with the distribution of fitness assigned to these children providing a picture of the potential outcomes from the crossover operator. A similar investigation for mutation was conducted, in which 100 copies of each SDA underwent one application of mutation. This test was performed at the start and end of evolution and every 500 000 mating events.

### D. Experiments Performed

Table I shows all parameter settings, with all combinations tested as separate experiments to determine which configuration produced the best results. The justification for the selection of these parameters is explained above, with the exception of the initial number of transition and response mutations ($m_t$ and $m_r$), the tournament size, and the number of states $N_s$. As explained in Section III-B, in [4] the maximum number of mutations was set to $M$, with $3 \leq M \leq 10$ typically producing the best results. Now using the new mutation operator, values of two and four were used for both $m_t$ and $m_r$. In [4] several tournament sizes were tested, with negligible differences in results and so only tournament size-seven is used here. The number of states is 20, as this achieved the best results in [4].

TABLE I: Parameter settings

| Parameter | Values Tested |
|---|---|
| Population Size | 50, 500 |
| Crossover Rate | 50% |
| Crossover Operator | 2-point crossover |
| Mutation Rate | 100% |
| Mutation | Static, $d_1$, $d_2$, $d_3$, $d_4$ |
| Culling Rate $CR$ | Random 25% |
| Culling Frequency $CF$ (Reporting Intervals) | 1, 5 |
| Initial # Transitions Mutations $m_t$ | 2, 4 |
| Initial # Response Mutations $m_r$ | 2, 4 |
| Tournament Size | 7 |
| # SDA States $N_s$ | 20 |

## IV. Results and Discussion

Box plots for the performance of the evolutionary algorithm are shown in Figure 2, for both sequences and both population sizes. Most glaring is the difference in performance between the static and dynamic mutation methods. It appears that a static quantity of each type of mutation produces better results than any of the dynamic adjusters tested in this paper; we note that it is likely that the range of allowed mutations (50) is too large in the dynamic methods, so this needs further investigation. For the static results there is a slight preference for more frequent culling of the population whereas when dynamic mutation is used there is a preference for less frequent culling; this is stark for $d_1$, $d_2$, and $d_3$. A change in culling frequency has more impact on fitness when the population is larger. For the static form of mutation, there is a slight negative trend as the number of mutations (of either type) increases; furthermore, no preference for transition or response mutation is demonstrated. For sequence 0 the smaller population size achieves the best results while for sequence 1 (which is longer), the larger population produces better results. Overall the spread of fitness values is larger when the population size is larger.

### A. Investigation into Effect of Crossover

The results from the investigation into the crossover operator described in Section III-C are provided in Figure 3. Before evolution, the majority of fitness values occupy the same low range and it is common for children to have better fitness than their parents. This indicates that growth is possible through the exploitation of these parents. Conversely, parents with better fitness tend to produce children with worse fitness, indicating they could benefit from greater exploration. After 2 500 000 mating events the distributions once again mostly overlap and obey the same pattern depending on the parent's relative fitness to the rest of the population. However, more-fit parents rarely produce children with better fitness. This indicates that the parents with the population best fitness would likely benefit from an increased amount of exploration. Whereas, those with worse fitness are still benefiting from exploitation.

### B. Investigation into Effect of Mutation

Recall that every SDA (in this analysis, called the "parent") is copied 100 times and each copy undergoes a single mutation to create 100 SDAs called the "children" in this analysis. Details on the effect of mutation are shown in Figure 3. In the initial population the violins encompass the fitness of the parent with some children performing better and others worse. Thus, mutation is able to improve fitness, although the more fit parents are more likely to have children with worse fitness, and vice-versa. For most parents many of their children have

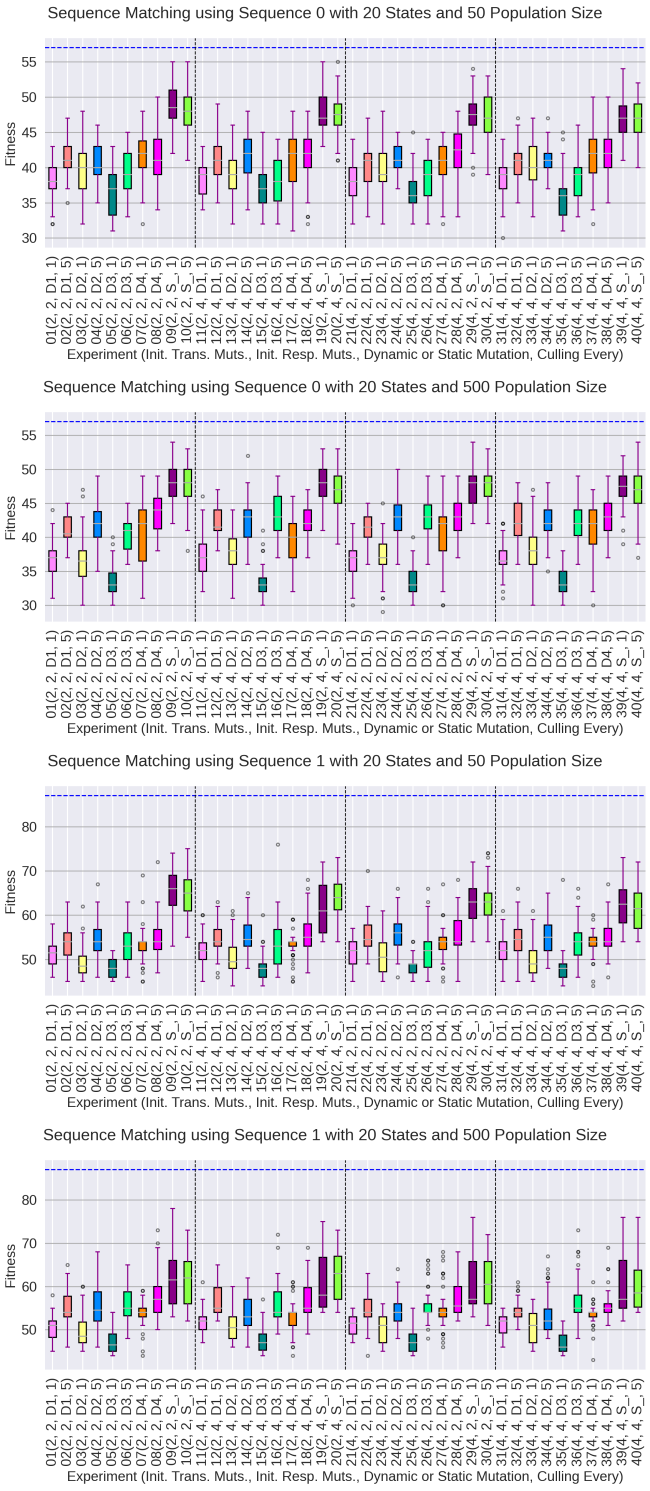



Fig. 2: Box plots of the best fitness from the 50 runs for each experiment using the specified population size and sequence. Each experiment is listed along the x-axis with its associated parameter settings for the initial number of transition mutations, initial number of response mutations, whether the number of mutations is static or dynamic, and the number of reporting intervals between cullings.

the same fitness as they do. Later in evolution, the majority of the population has fitness equal to the population best fitness. For these experiments most of the children have a significantly worse fitness than their parents while a few retain the same or slightly worse fitness. No mutations result in children achieving a better fitness than the population best. For the SDAs with fitness below the population best, the violins are nearly identical in shape to those before evolution. Examining the average child fitness minus the parent's fitness it is observable that generally the child's average fitness was significantly worse than the parent's fitness, with some outliers having the child's fitness being slightly better.

### C. Genetic Operators and Convergence

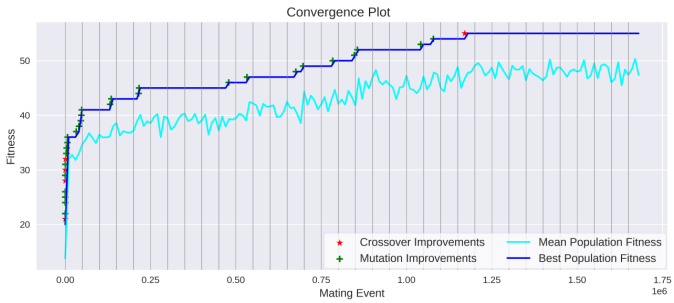
To investigate how fitness changes over time we examine two convergence plots. First, consider Figure 4a, corresponding to experiment 10 with population size 50 for sequence 0, one of the best-performing experiments. Each improvement in population best fitness is depicted, and reveals that the vast majority of improvement is attained during the initial 10 000 mating events, and largely a result of mutation. The emphasis on mutation remains for the rest of evolution while crossover still improves fitness a few times. Overall there are only 33 improvements over the 1 680 000 mating events that occur, meaning the likelihood of a given crossover or mutation operation improving fitness is extremely rare. Now consider Figure 4b, corresponding to experiment 8 with a population of size 50 for sequence 0, an experiment with mediocre results. This experiment uses dynamic mutation method $d_4$ and is emblematic of the effect all dynamic methods have on the overall population: the mutation operator becomes destructive, lowering the mean population fitness once the number of mutations approaches 50; which should be considered in future work. Most convergence plots show a decisive preference for mutation and are similar to those provided here. To investigate this preference the best experiment from each box plot in Figure 2 was repeated without crossover and again without mutation. Using only mutation yielded fitness of $48.6 \pm 1.1$, $47.5 \pm 1.0$, $65.7 \pm 1.6$, and $64.1 \pm 2.0$, respectively. Only crossover yielded fitness of $25.9 \pm 0.5$, $28.5 \pm 0.5$, $35.3 \pm 0.6$, and $41.3 \pm 0.7$. Mutation alone was similar to the best experiments in each box plot while crossover drastically underperformed, hence the preference seen in the convergence plots.

### D. Population Diversity

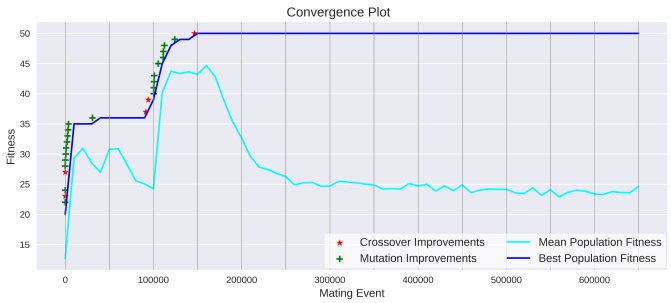
The above investigations indicate that shortly after the start of evolution much of the population coalesces to the population best fitness. To investigate how similar the SDAs are to each other, we generate heatmaps of the population that show the concentration of transitions and corresponding triggering symbols shared among the members of the population at given points of evolution. Figure 5 shows the heatmaps for experiment 9 with population size 50 for sequence 0.

Before evolution begins (Figure 5a) the majority of the transitions and driving symbols are diverse, as few SDAs in the population share the same connections. In fact no more


Fig. 3: Violin graphs depicting fitness of children generated by the crossover and mutation investigations explained in Section III for the best run of experiment 9 with a population size of 50 with sequence 0 before evolution and after 250 000 mating events. This was the experiment leading to the best overall fitness for this sequence.




(a) Experiment 10 (Static with $CE = 5$)



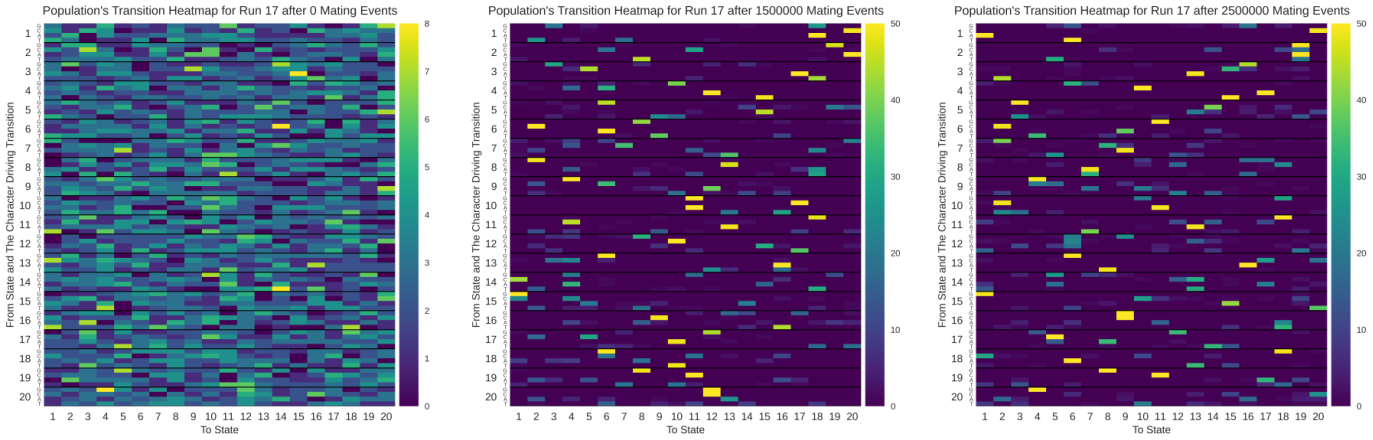

(b) Experiment 8 (Dynamic4 with $CE = 5$)

Fig. 4: Convergence graphs depicting best and mean population fitness of the best run for the indicated experiment with population size 50 using sequence 0. Every improvement in fitness is indicated by the operator that improved it. The vertical lines represent when cullings take place.

than 8 SDAs observed share a particular connection and there is an adequate distribution of associations between the states and transition-driving symbols in the population. After 1 500 000 mating events (Figure 5b), some significant changes to the heatmap are observed. Most noticeably, the shared connections have increased, as now all population members have several transitions and triggering-symbols shared between them. Additionally, many potential transitions within the population do not exist, thus the diversity has decreased. Finally, the heat map after 2 500 000 mating events (Figure 5c) shows a similar pattern in the distribution and concentration of transitions and triggering-symbols. However, note that in this heatmap, several of the concentrations of connections have changed position in comparison to Figure 5b while others remain in the same location. This indicates that the population is still evolving.

## V. Conclusions and Future Work

In this paper we examined several questions regarding evolution of SDAs, using sequence matching as a test problem. It was revealed that using the static mutation method provided better results than dynamically adjusting values, whose upper bounds proved too high for the scenarios examined. Also, less frequent culling events tended to produce better results, with culling having a greater effect on larger populations. Further investigation of the crossover and mutation operators revealed both can be destructive, especially for parents having the best fitness. However, SDAs with worse fitness see improved fitness in some children. It was observed that most improvements occurred in the first 10 000 mating events, predominantly from mutations, with crossover contributing less frequently to overall gains. In fact, forgoing crossover entirely yields similar results to when both are used. Finally, the investigation




(a) Initial Population (b) After 1 500 000 Mating Events (c) After 2 500 000 Mating Events

Fig. 5: Heatmaps depicting the frequency of transitions between states and the symbol driving these transitions for the population of 50 SDAs at the indicated point of evolution for experiment 9 with population size 50 with sequence 0 in Figure 2.

into the population's diversity revealed that before any mating events occur, great diversity is present in the population. As evolution progresses diversity decreases, but importantly, transitions within the population still change over time.

Evolutionary algorithms using SDAs as a representation have been applied to many problems, although the use of an alphabet larger than two was introduced only recently. This study provided some valuable information as to the underlying dynamics influencing the evolution of SDAs, which can now be employed in attacking problems mentioned in Section I, and others. This serves as only the start of the investigation. The dynamic methods used in mutation would likely perform better with a smaller lower bound, and this should be tested. Further it may be useful to reduce how quickly the numbers of mutations increase. Investigating how the modifications used in this work affect fitness of different problems would also provide more insight into the underlying evolution of SDAs.

Investigating the distribution of all possible children generated by two parents was planned to be part of this study but excluded due to page constraints. This can give a better understanding of potential diversity within the population, and could be a restraint introduced in future work. The introduction of multiple populations which evolve concurrently and reproduce with one another sparingly could be another means to increase overall population diversity. A further possible avenue would be to include such concepts as Novelty Search [10] and MAP-Elites [15], which attack this head-on by differently focusing the search to ensure novelty and to "illuminate" regions of the feature space, respectively.

## References


[1] D. Ashlock and M. Dubé. A comparison of novel representations for evolving epidemic networks. In *2021 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology*, pages 1–8, 2021.

[2] D. Ashlock and S.K. Houghten. Dna error correcting codes: No crossover. In *2009 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, pages 38–45. IEEE, 2009.

[3] S.M. Corns. On the effects of graph based evolutionary algorithms for training finite state classifiers. In *2011 IEEE Congress of Evolutionary Computation (CEC)*, pages 1020–1026. IEEE, 2011.

[4] M. Dubé, J. Sargant, S. Houghten, and S. Graether. From bits to bases: Evolving a versatile construct for biological sequence and network data. In *2023 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology*, 2023.

[5] M. Dubé and D. Ashlock. Procedural content generation of levels with increased connectedness using complex string generators. In *2021 IEEE Symposium Series on Computational Intelligence*, pages 1–8, 2021.

[6] M. Dubé and S. Houghten. Evaluation of frameworks for epidemic variants and infectivity using an evolutionary algorithm. In *2022 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology*, pages 1–9, 2022.

[7] M. Dubé and S. Houghten. Now I know my alpha, beta, gammas: Variants in an epidemic scheme. In *2022 IEEE Congress on Evolutionary Computation*, pages 1–8, 2022.

[8] J.E. Hopcroft. *Introduction to Automata Theory, Languages, and Computation*. Always Learning. Pearson Education, 2008.

[9] W. Kolakoski. Self generating runs. *American Mathematical Monthly*, 72:674, 1965. Problem 5304.

[10] J. Lehman and K.O. Stanley. Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary computation*, 19(2):189–223, 2011.

[11] S.M. Lucas and T.J. Reynolds. Learning dfa: evolution versus evidence driven state merging. In *2003 Congress on Evolutionary Computation (CEC)*, pages 351–358, 2003.

[12] S.M. Lucas and T.J. Reynolds. Learning deterministic finite automata with a smart state labeling evolutionary algorithm. *IEEE transactions on pattern analysis and machine intelligence*, 27(7):1063–1074, 2005.

[13] M.L. Mauldin. Maintaining diversity in genetic search. In *1984 AAAI National Conference on Artificial Intelligence*, pages 247–250, 1984.

[14] G. H. Mealy. A method for synthesizing sequential circuits. *Bell System Technical Journal*, 34:1045–1079, 1955.

[15] J-B. Mouret and J. Clune. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909*, 2015.

[16] K. Oğuz. Adaptive evolution of finite state machines for the tartarus problem. In *2019 Innovations in Intelligent Systems and Applications Conference (ASYU)*, pages 1–5. IEEE, 2019.

[17] A.C. Riley, D.A. Ashlock, and S.P. Graether. Evolution of the modular, disordered stress proteins known as dehydrins. *PLOS ONE*, 14(2):1–20, 02 2019.

[18] P. Shell et al. Improving search through diversity. In *1994 AAAI National Conference on Artificial Intelligence*, pages 1323–1328, 1994.

[19] F. Tsarev and K. Egorov. Finite state machine induction using genetic algorithm based on testing and model checking. In *Proc. GECCO*, pages 759–762, 2011.