Constructing Slowly Synchronizing Automata Using Genetic Algorithms

Student(s)

Şeyma Selcan Magara
Yiğit Aras Tunalı

Faculty Member(s)

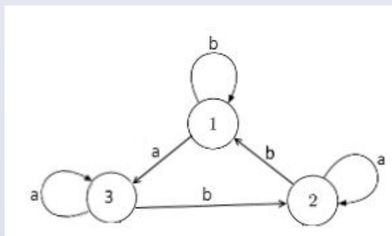
Hüsnü Yenigün
Kamer Kaya

Sabancı Üniversitesi

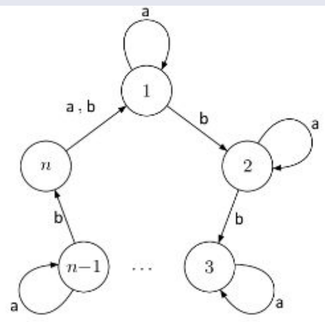
PURE
PROGRAM FOR UNDERGRADUATE RESEARCH

ABSTRACT

- Computation of a shortest synchronizing sequence of an automaton is an NP-Hard problem. The longest possible shortest path and it's structure is well known and studied.
- There are various automatons having shorter synchronizing length than the maximum but still are considered hard finite state machines.
- We used a Genetic Algorithm approach to generate hard finite state machines of different sizes



A synchronizing automata
synchronizing word: bb



General structure of Cerny Machines

OBJECTIVES

Using Genetic Algorithms to generate slowly synchronizing automata and further improving this approach with different crossover and mutation functions.

PROJECT DETAILS

- A Deterministic Finite Automaton is a tuple A = (Q,δ,Σ,q,F) where
  - Q is a finite set of states
  - Σ is finite input alphabet
  - q is the initial state q∈Q
  - F is the final state set F ⊆ Q
  - δ is the transition function where δ : Q × Σ → Q;(q,σ) → δ(q,σ) ∈ Q
- For an automaton A = (Q,δ,Σ,q,F), a Synchronizing Sequence (SS) of A is an input sequence **w** ∈ Σ* such that |**δ**(Q, **w**)| = 1.

- The approach we took in this research was to use the insights from (Podolak, Roman, Szykula, & Zielinski, 2017) and apply them to our mutation and crossover functions to construct a Genetic Algorithm scheme that would produce the desired slowly synchronizing automata using a randomly generated population.
- A shortest-path algorithm is used to give each member of the population a "fitness" level, which we use to rank the automata and eliminate the ones having low scores throughout our genetic algorithm cycles.
- After some testing , we have decided not to mutate the highest scoring 2-5 automata and keeping them as elites of the population. This decreased the number of cycles required for improvement and eliminated the off chance of losing automatas with high "fitness" level due to random mutations.

• **Parameters:** $n, c, r \in Z$.
procedure GENETIC GENERATOR(n, c, r)
  Population ← n number of randomly generated automata and their scores as tuples
  while $c \neq 0$ **do**
    Sort *Population* according to the scores of automata
    $a = floor(n * (r - 1)/r)$
    remove a^{th} to n^{th} elements of Population
    Use *Cross-Over* function $n - a$ times to reproduce the eliminated automata
    apply *Mutation* functions probabilistically to the *Population*
    recalculate the scores of new and changed automata
    $c = c - 1$
  end while
end procedure

- There was some number of crossover and mutation functions we have used but after various tests, we have concluded that a couple of them were contributing to a higher "fitness" score.
- The mutation functions we used were inspired by the results of (Podolak et al. 2017). The best performing mutations were the ones that introduced self-loops to states and the ones that balanced the in-degree values of states. Thus we used the aforementioned mutations coupled with a random switch mutation, each with a separate probability so that we would introduce randomness to our population.

PROJECT DETAILS II

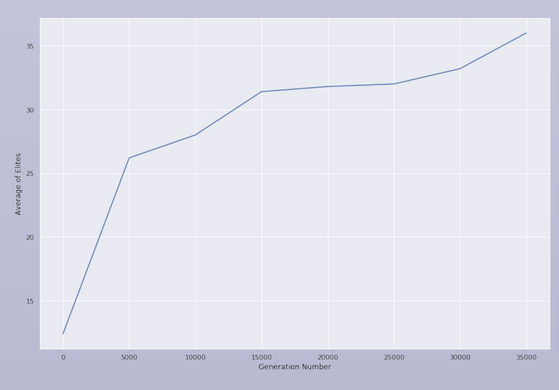
- The location of the self-loop and the in-degree balancing mutations were all executed with different probability values so that our population wouldn't converge on an unwanted result and would keep on increasing. The random switch had the lowest probability of happening, whereas the self-loop mutation had the highest.

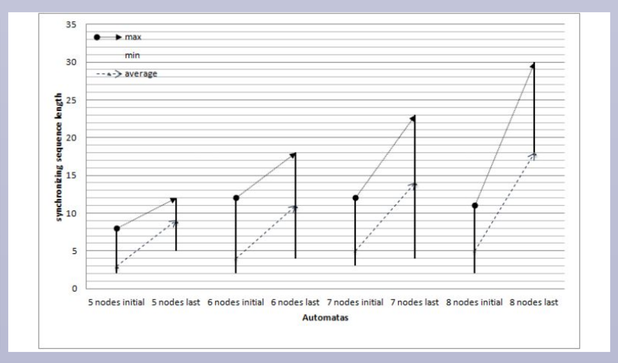
procedure NODE-WISE MIX(*Population*, *NewAutomata*)
  for $k \leftarrow 0$ to $N - 1$ **do** ▷ where N is the number of states
    $i \leftarrow$ random integer [0 (N*P-1)]
    $j \leftarrow$ random integer [0 (N-1)]
    NewAutomata[k] ← transitions of Population[i]'s j^{th} state
  end for
  return NewAutomata
end procedure

- For cross-over function we had; Node-Wise Mix, Alphabet-Wise Mix, Three-Way Mix, and a random initialization. One of these crossover methods would be picked at random for creating each new member of the population during the end of each cycle. The probability of random initialization was the lowest and the probability of Node-Wise Mix was the highest.
- With the Node-Wise Mix, we aimed at creating a new member by mixing the nodes of randomly chosen automata. With the Alphabet-Wise Mix, on the other hand, a new automaton was created by mixing the languages of P different automata where P is the size of the alphabet. For example, if our alphabet was {a,b} we would take the transitions from one automaton and b transitions from another automaton to create the new one.
- The Three-Way Mix was just a simple mixing of 3 automata, where we take apart from each automaton, we have to mention that for ease of computation we are storing automata as N*P length arrays where N is the number of states and P is the size of the alphabet, and create a new one.

procedure THREEWAY CROSSOVER(*Automata*1, *Automata*2, *Automata*3)
  $i \leftarrow$ random integer [0 (N*P-1)]
  $ii \leftarrow$ random integer [0 (N*P-1)] ▷ i<ii and i+ii < N*P-1
  $NewAutomata = Automata1[0 : i] + Automata2[i : i + ii] + Automata3[i + ii : end]$
  $returnNewAutomata$
end procedure

CONCLUSION





- Genetic algorithms can be used to generate slowly synchronizing automatas from randomly generated machines although improvement of the algorithm is necessary.
- Shortest synchronizing word lenght increases as the number of genetic cycles increases.
- As the size of our automatas increase the necessary numbers of cycles required for increase in scores also rapidly increases
- Diversity of the population and selection of the elite group are the important issues to be studied in order to improve the algorithm

References

J. Černý, "Poznámka k homogénnym eksperimentom s konecnými automatami," 1964.

I. Podolak, A. Roman, M. Szykula, and B. Zielinski, "A machine learning approach to snychronization of automata," Dec 2017.

G. Ananichev and Volkov, "Slowly synchronizing automata and digraphs," 2010.

M. V. Berlinkov, "On the probability to be synchronizable.," 2013.

D. Eppstein, "Reset sequences for monotonic automata," Feb 1990