# Solving Identification Problem for Asynchronous Finite State Machines Using Genetic Algorithms

Xiaojun Geng
Department of Electrical and Computer Engineering
California State University
Northridge, CA 91330
01-818-677-4755
xjgeng@csun.edu

## ABSTRACT

A Genetic Algorithm, embedded in a simulation-based method, is applied to the identification of Asynchronous Finite State Machines. Two different coding schemes and their associated crossover operations are examined. It is shown that one operator / coding pair outperforms the other in that the scheme reduces noticeably the production of invalid chromosomes thus increasing the efficiency and the convergence rate of the evolution process.

## Categories and Subject Descriptors

I.6.5 [Model Development]: Modeling methodologies.

## General Terms

Algorithms, Performance, Experimentation.

## Keywords

Model Identification, Asynchronous Finite State Machines, Genetic Algorithms, Coding Schemes.

## 1. INTRODUCTION

The modeling of Asynchronous Finite State Machines (AFSMs) plays an important role in design, testing, and control. To discover an internal model of the AFSM system from the given external input/output behavior, a simulation-based method using Genetic Algorithm (GA) is proposed in this paper.

The main points addressed are the chromosome coding of AFSMs based on graphical matrices, the relation between coding schemes and crossover operators, and the improvements that could be obtained in the performance of the GA, particularly the reduction of invalid chromosomes generated during the process of GA search and the acceleration of GA convergence.

## 2. THE IDENTIFICATION PROBLEM

An AFSM is a finite state machine which moves from one internal state to another in response to an input signal change. The output signal of the AFSM depends on the present state of the machine.


*GECCO'06*, July 8–12, 2006, Seattle, Washington, USA.
ACM 1-59593-186-4/06/0007.

An AFSM differs from a finite state machine in that it may have two types of states: stable and unstable states. After an input is asserted, an AFSM will reach the next stable state in no time by going through a number of intermediate unstable states [2,3]. The lack of a synchronizing clock causes the lack of control of the transient state changes. In this paper, we restrict the AFSMs to operate in fundamental mode. In this mode, when an input string is applied to an asynchronous machine, the next input is asserted only when the machine is in a stable state with the present input.

The identification problem is depicted in Figure 1. The work of solving model identification problem for Finite State Machines with GA can be found in [1,4].

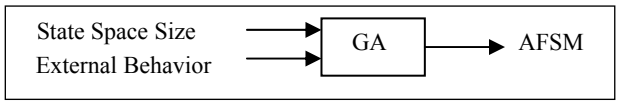


**Figure 1. Identification Problem of AFSMs.**

## 3. CHROMOSOME REPRESENTATIONS

The transition matrix is a matrix which has a row for each state and column for each input character, with the entry (i,j) being the destination stable state from the state $x^i$ with the input $u^j$.

One type of chromosome representation is built by sequentially arranging the rows of the corresponding transition matrix in a single row, which is referred to as row-coding #1. Associating a row with each input character and column with each state, we get the transpose of the transition matrix. In similar way, this transpose matrix can produces a row-structure chromosome representation, which is referred to as row-coding #2. Based on these representations, next, we work on how to select evolutionary operators and how to conduct genetic algorithms to increase the computation efficiency.

## 4. IMPLEMENTATION ISSUES

### 4.1 Crossover and Coding Schemes

With the two chromosome representations presented above, crossover operators with 1-dimensional cut points can be easily employed, such as the 1-point crossover operator. An example of the crossover operation on two chromosomes is shown in Figure 2(a). The shadow highlights the selected crossover region. When these two machines are encoded with the row-coding #2 scheme,

the equivalent effect of the operation in Figure 2(a) is shown in Figure 2(b).

Notice that the 1-point crossover operator for the row-coding #1 turns into a 'special' multi-point operator for the row-coding #2 representation. Conversely, for the machine encoded with the row-coding #2, the 1-point crossover operator behaves as a 'special' multi-point operator on the row-coding #1 chromosome as well.

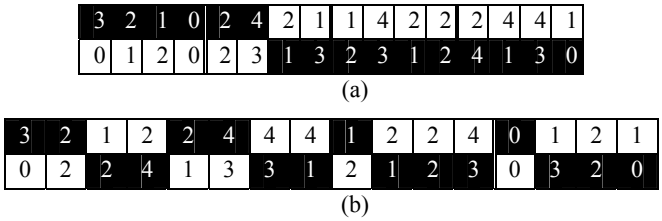


**Figure 2. (a) 1-point crossover operations with row-coding #1; (b) the equivalent effects of (a) with row-coding #2.**

Going back to the coding of transition matrix structure, the corresponding effect of the 1-point crossover operation in Figure 2(a) is depicted in Figure 3(a).

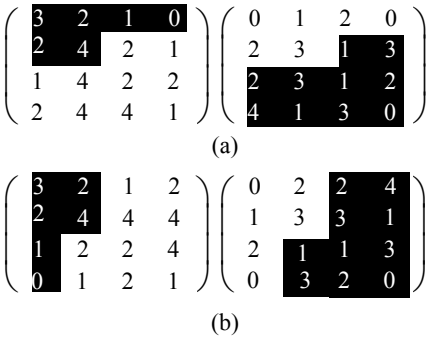


**Figure 3. (a) The 'Γ'-shaped #1 crossover operation; (b) The 'Γ'-shaped #2 crossover operation.**

We refer to the 2-dimensional crossover operator illustrated in Figure 3(a) as 'Γ'-shaped #1 crossover, which is defined as follows. Given the size of the matrix $m \times n$, randomly select two integers $m'$ and $n'$ for which $0 < m' < m$ and $0 < n' < n$, then the crossover region contains the entries $(i,j)$ where $i \le m'$, or $i = m'+1$ and $j \le n'$. In the example of Figure 3(a), we have $m' = 1$ and $n' = 2$.

When the transpose of the transition matrix is used to encode machines, the equivalent operation of this 'Γ'-shaped #1 crossover is shown in Figure 3(b). Let us call this crossover operator for matrix structure as 'Γ'-shaped #2 crossover, which can be considered as a transpose version of 'Γ'-shaped #1 crossover operator.

## 4.2 Reduction of Invalid Machines

During the population initialization stage, many invalid machines are created due to the random generation of the chromosomes. From various experiments, the percentage of invalid machines created may range from 95% to 99%. Experiments show that increasing the quality of initial population speeds up the convergence of the evolution process.

Crossover and mutation operation produce invalid machines as well. In the following, we compare the effect of the 1-point crossover operation applying to the row-coding #1 and row-coding #2 chromosome representation.

*Proposition*: Given two asynchronous machines free of infinite cycles, applying the 1-point crossover operator to their row-coding #1 representation, results in a higher percentage of invalid child chromosomes than to their row-coding #2 representation.

The above proposition states that using the row-coding #2 to represent chromosomes could lead to better performance than using row-coding #1, with the 1-point crossover operator applied.

# 5. EXPERIMENTS AND CONCLUSIONS

Experiments are conducted to show that the types of crossover operators are closely related to the coding scheme of chromosomes, for example, the 1-point crossover operator works equivalently as some type of multi-point crossover operator when different chromosome coding is used. We have proved that the proposed row-coding #2 scheme is very effective in two aspects: faster convergence and lower percentage of invalid machines generated when coupled with the 1-point crossover operator.

# 6. REFERENCES


[1] Hingston, P. *A Genetic Algorithm for Regular Inference*. Genetic and Evolutionary Computation Conference, GECCO-2001, San Francisco, CA, July 2001, 1299-1306.

[2] Kohavi, Z. *Switching and Finite Automata Theory*, McGraw–Hill Book Company, New York, 1970.

[3] Murphy T., Geng X., and Hammer J. *On the control of asynchronous machines with races*, IEEE Trans. on Automatic Control, vol. 48, 2003, 1073 – 1081.

[4] Ngom, L., Baron, C., and Geffroy, J.-C. *Genetic Simulation for Finite State Machine Identification*, 32nd Annual Simulation Symposium, April 1999, 118-125.