Proceedings of the World Congress on Engineering 2011 Vol II
WCE 2011, July 6 - 8, 2011, London, U.K.

Improving FSM Evolution Algorithm

Nada M. A. Al Sallami

***Abstract*— To be able to represent FSM as a binary chromosome several restrictions are required. This paper analyzes and discuses the differences between classical FSMs and semantic FSA with evolutionary algorithms. To overcome the limitation of classical Mealy and Moore machine, we use semantic FSA. It is presented that when the evolutionary process is based on the behavior of small building function, then better evolutionary algorithm is developed. In addition, the evolutionary process is highly enhanced by using fitness inheritance technique to constrain the depth of genetic programming tree to overcome its bloat problem.**

***Index Terms*— Genetic Programming, Evolutionary Algorithm, Automatic Programming, Reinforcement Learning.**

I. Introduction

Evolutionary algorithms (EAs), which are based on a powerful principle of evolution: survival of the fittest, and which model some natural phenomena: genetic inheritance and Darwinian strife for survival, constitute an interesting category of modern heuristic search. Evolutionary algorithms are superior in terms of wide space search ability because they continue to evolve various individuals and select better ones (offline learning), while reinforcement learning can learn incrementally, based on rewards obtained during task execution (online learning) [1][2][3]. In the early 1960s Fogel introduced Evolutionary Programming (EP) [4][5]. The simulated evolution was performed by modifying a population of FSM. After this other researchers are used EP for solving the problem of FSM identification. Kumar Chellapilla and David Czarnecki proposed the variation of EP to solve the problem of modular FSM synthesis [6]. Karl Benson presented a model comprising FSM with embedded genetic programs which co evolve to perform the task of Automatic Target Detection [7].Another approach to solve the problem of FSM identification is based on GA. This method has been researched by several authors. Tongchim and Chongistitvatana investigated parallel implementation of GA to solve the problem of FSM synthesis [8]. Chongistitvatana and Niparnan improved GA by evolvingonly the state transition function [9]. Chongistitvatana also presented a method of FSM synthesis from multiple partial input/output sequences [10]. Jason W. Horihan and Yung–Hsiang Lu paid more attention to improving the FSM evolution by using progressive fitness functions [11].Different types of machines can be inferred using GA: Lamine Ngome used genetic simulation for Moore machine identification [12], Pushmeet Kohli used GA to synthesize FA accepting particular language using accept/reject data [13], Philip Hingston showed in [14] how GA can be used for the inference of regular language from a set of positive examples, Xiaojun Geng applied GA for solving identification problem for asynchronous FSM [15]. The algorithm for Automated Negotiations presented by Tu, Wolff and Lamersdorf is based on GA synthesis of FSM [16]. Simon M. Lucas paid more attention to finite state transducers [17] and compares his method to "Heuristic State Merging" [18].GA has also been used for solving other similar problems: for solving State Assignment Problem [19], for identification of nondeterministic pushdown automata[20], for inferring regular and context-free grammars [21], for protecting resources [22]. The researcher in the reference [23] was use genetic programming for FSM induction, such FSM have special formalism based on input-output trajectory sets. In this paper an analysis and discussion are given about the differences between the classical FSA used by other researchers and this FSA (called Semantic Finite State Automata SFSA). It is presented that when the induction process is based on the behavior of small building function better EA is developed. Furthermore, complex systems often include chaotic behavior [24], which is to say that the dynamics of these systems are nonlinear and difficult to predict over time, even while the systems themselves are deterministic machines following a strict sequence of cause and effect. The nonlinearity of chaotic systems results in the amplification of small differences, and this is what makes them increasingly difficult to predict over time. Natural chaotic systems may be difficult to predict but they will still exhibit structure that is different than purely random systems. Chaos is important, in part, because it helps us to cope with unstable system by improving our ability to describe, to understand, perhaps even to forecast them . In this work we attempted to scale-up GP application to real live problems, by focusing on the meaning rather than the structure of a program. This paper is organized as follows. Section 2 provides the related definition of FSM and its limitation in genetic evolutionary process. In Section 3, genetic evolutionary process based on input-output specification is described. Sections 5 give discussion and conclusions.

To insert images in *Word,* position the cursor at the insertion point and either use Insert | Picture | From File or copy the image to the Windows clipboard and then Edit | Paste Special | Picture (with "float over text" unchecked).

II. FSA Definition

A. Moore Machine

is a six-tuple : (Q, Σ, Δ, δ, λ, qo)
, where: Q is a finite set of states, Σ is the input alphabet, Δ is the output alphabet, $\delta: Q \times \Sigma \rightarrow Q$ is the transition function, $\lambda : Q \times \Sigma \rightarrow \Delta$ is the output function that shows

Nada M. A. Al Sallami is an Associated Prof. in MIS department, faculty of Economic and administration sciences. She interested in Evolutionary algorithm and the theory of computer sciences.

ISBN: 978-988-19251-4-5
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

what character from Δ will be printed by each state that is entered., and qo denotes the start state

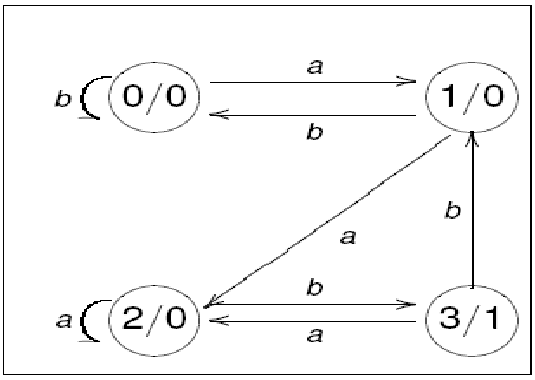


Figure 1: Moore machine example

B. Mealy Machine

It is a six-tuple : (Q, Σ, Δ, δ, λ, qo)
, where: Q is a finite set of states, Σ is the input alphabet, Δ is the output alphabet, δ: Q × Σ → Q is the transition function, λ : Q × Σ → Δ is the output function that . Character from Δ will be printed by each transition that is processed, and qo denotes the start state.

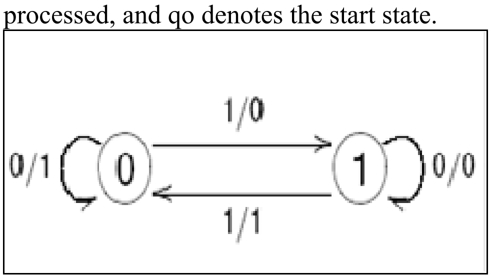


Figure 2: Mealy machine example

C. Evolutionary Algorithm

The goal is finding a minimum size deterministic FSM consistent with the training set, clearly its NP-complete problem. Evolutionary algorithms can be used to solve such problem. Figure 3 show how training and test sets can be used to get complete solution with EA.

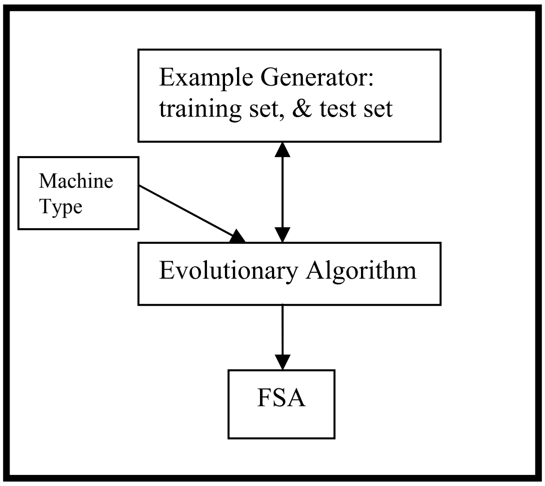


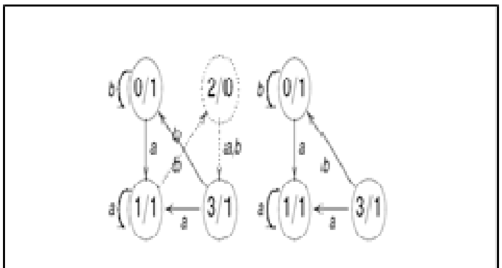
Figure 3: Creating sets of training and set with EA

D. Limitation of FSA Induction

To be able to represent FSM as a binary chromosome several restrictions are required.

1. No final state. FSM finishes its work then precedes the input string to the end.
2. Initial state. FSM must have only one initial state and this state is always labeled as '0'. Deterministic.
3. FSM has only one initial state and only one possible transition for each input value. Complete.
4. For each state and each input symbol, there must be one edge.

Figure 4, show a Moore machine with dynamic number of states, In that case there appear some problems like: Non-complete transitions; Un accessible states. Solving the problem of non–complete transitions may base on partial solution. FSM will stop working when it reaches the state it cannot leave and will produce partial output (during 'original' work-flow FSM can stop only if an input string is processed). In addition post–processing stage must be used to solve problem of un accessible states.



		a	b			a	b			a	b			a	b
1	1	1	0	1	1	1	2	0	0	3	3	1	1	1	0
1	1	01	00	1	1	01	10	0	0	11	11	1	1	01	00

Figure 4: Moore machine with dynamic number of states

III. SEMANTIC FSA

As given in [23], semantic Finite State Automata SFSA is defined as 9- tuples:
P=(x, X, T, F, Z, I, O, γ, $X_{initial}$), where: x is the set of system variables, X is the set of system states, X= { X initial, ….. , X final}, T is the time scale, T =[0, ∞), F is the set of primitive functions, Z is the state transition function, Z = {(f, X, t): (f, X, t) Є F × X × T , z(f, X, t) = (•X, •t)}, I is the set of inputs, O is the set of outputs, γ is the readout function, X initial is the initial state of the system, X initial Є X.
The evolution SFSM depend on input-output behavior of the system, and it is expressed as 7-tuples: (IOS, S, F, α1, T_{max}, β, υ). Where: IOS is the input-output boundaries of the system, S is the syntax term, F is the primitive function, . α1 is the learning parameter, T_{max} and β are the complexity parameters , and υ system proof plan.
IOS describes the inputs that the system is designed to handle and the outputs that the system is designed to produce. An IOS is not a system, but it determines the set of all systems that satisfy the IOS. It is a 6-tuples: IOS = (T, I, O, Ti , To, η). Where T, is the time scale of IOS, I is the set of inputs, O is a set of outputs, Ti is a set of input trajectories defined over T, with values in I, TO, is a set of

ISBN: 978-988-19251-4-5
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

output trajectories defined over T, with values in O, and η is a function defined over Ti whose values are subset of To; that is, η matches with each given input trajectories Ti the set of all output trajectories that might, or could be, or eligible to be produced by some systems as output, experiencing the given input trajectory Ti.. . A system P satisfies IOS if there is a state X of P, and some subset U not empty of the time scale T of P, such that for every input trajectory g in Ti, there is an output trajectory h in To matched with g by η such that the output trajectory generated by S, started in the state X is:

γ (Z (f (g), X, t) = η(h(t))For every t Є U …Eq. 1

Learning parameter a_1 is a positive real number specifying the minimum accepted degree of matching between an IOS, and the real observed behavior of the system over the time scale, T_x, of IOS only. T_{max} and β parameters are merits of system complexity: size and time, respectively. It is important to note that there is a fundamental difference between a time scale T and an execution time of a system. T represents system size, it defines points within the overall system, whereas, β, is the time required by the machine to complete system execution, hence it is high sensitive to the machine type.

The search space in Genetic Program Generation (GPG) algorithm is the space of all possible computer programs described as an 9-tuples SFSA. Multi-objective fitness measure is adopted to incorporate a combination of correctness (satisfy IOS), parsimony (smallness T), and efficiency (smallness β).The fitness value of individual is computed by the following equation:

$$fitness(i) = \delta\left(\alpha_1 - \sum_{j=0}^{Tx} \left|\eta(T_i(j)) - \eta(R_i(j))\right|\right) + (T_{max} - T_i) + (\beta - \beta_i)$$

…Eq.2

Where: δ is the weight parameter, δ >=2, βi the run time of individual i, Ti is the time scale of the individual i, Ri is the actual calculated input trajectory of individual i. For complex problem, it becomes difficult to apply the proposed GPG because the cost of determining fitness values for an entire population is prohibitive. A child's inherited fitness can be seen as an approximation of average fitness of the common schemata of the parents [23][24][25]

IV. Analysis

It clear that the evolutionary process of our system is highly depends on input-output specifications, more precisely input and output trajectory sets, and η function. Unfortunately, when we deal with complex systems and real live problem, strong feedback (positive as well as negative) and many interactions exist: i.e. chaotic behavior, as we explain in part I. Thus, we need to find a way to control chaos, to understand, and predict what may happen long term. In these cases input and output specifications are self organized, which mean that trajectory data are collected and enhanced over time, when genetic generation process runs again and again. Genetic generation process begins with initial version of input and output trajectory sets, and η function. Then change them over time to reflect input-output characteristic of the required system. The output of each SFSA individual, at any generation, is computed through using equation number 1, and evaluated to determine it's fitness by using equation number 2. Figure5, specify clearly that SFSA populations, with high trajectory information converge to the solution in less time than these populations with little trajectory information. From the figure little data trajectory information always may lead to un convergence state, that is maximum generation number allowed her is 5000. Therefore, main problem will appear if the system has little trajectory information. In this case, problem properties can be described mathematically by using formal software engineering methods, if it has poor input-output properties. These mathematics are implemented in the context of a formal specification language, such as Z. Formal methods are focus primarily on function and data, therefore, they must be redesigned in a way that overcome their difficulty to represent timing, control, and behavioral aspects of a problem.

V. Conclusion

Generally SFSM solve most problem and limitations found in traditional FSA. The states are connected by trajectory information sets, so it is possible that only the essential problem's behavior obtained in the current situation are used in the network flow, and it can determine an action by not only the current, but also the past information. .The inheritance analyses indicate that the expected effects of inheritance are consistent with the schemata-based processing of the GA. Reduction of the expense of evaluating a population, through inheritance techniques could substantially enhance the GA's applicability, especially in the application where the typical GA's population based approach may be prohibitively expensive. SMA can yield systems with small overall size, and hence less time is required to execute such optimized system on parallel machines.

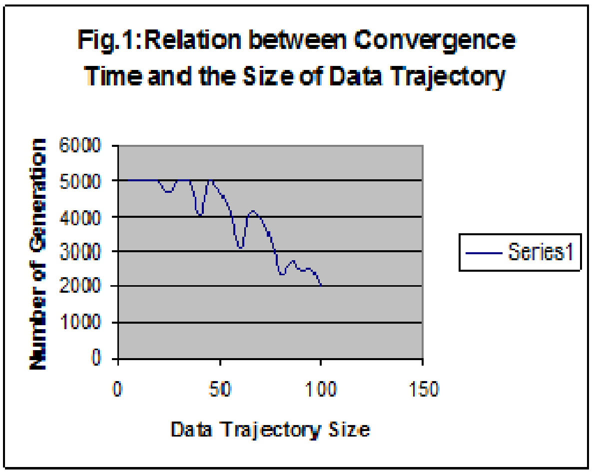


Figure 5: Data Trajectory effect

ISBN: 978-988-19251-4-5
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

References

[1] Koza, J. R., “Genetic *Programming, on the Programming of Computers by Means of Natural Selection”, 1992* MIT Press, Cambridge, MA.

[2] Koza, J. R. , “ *Genetic Programming II, Automatic Discovery of Reusable Programs*”, 1994 MIT Press, Cambridge, MA H. Poor, *An Introduction to Signal Detection and Estimation*. New York: Springer-Verlag, 1985, ch. 4.

[3] ShingoMabu, Kotaro Hirasawa, Jinglu Hu, “A Graph-Based Evolutionary Algorithm: Genetic Network Programming (GNP) and Its Extension Using Reinforcement Learning”, 2007 by the Massachusetts Institute of Technology Evolutionary Computation 15(3): 369-398.

[4] L. J. Fogel, A. J. Owens, and M. J. Walsh., " Artificial Intelligence throughSimulated Evolution", John Wiley, 1966.J. Wang, “Fundamentals of erbium-doped fiber amplifiers arrays (Periodical style—Submitted for publication),” *IAENG International Journal of Applied Mathematics*, submitted for publication.

[5] Fogel, D. B. , “An introduction to simulated evolutionary optimization”, 1994 *IEEE Transactions on Neural Networks*, 5(1):3–14.

[6] Kumar Chellapilla and David Czarnecki. A preliminary investigationinto evolving modular finite state machines, 1999.

[7] Karl A Benson. Evolving finite state machines with embedded geneticprogramming for automatic target detection within SAR imagery. InProceedings of the 2000 Congress on Evolutionary Computation CEC00,pages 1543–1549. IEEE Press, 6-9 July 2000.

[8] Shisanu Tongchim and Prabhas Chongstitvatana. Parallel genetic algorithm for finite state machine synthesis from input/output sequences.In Erick Cantu-Paz and Bill Punch, editors, Evolutionary Computationand Parallel Processing, pages 20–25, Las Vegas, Nevada, USA, 8 2000.

[9] Nattee Niparnan and Prabhas Chongstitvatana. An improved geneticalgorithm for the inference of finite state machine. In GECCO ’02:Proceedings of the Genetic and Evolutionary Computation Conference,page 189, San Francisco, CA, USA, 2002. Morgan Kaufmann PublishersInc.

[10] Chatchawit Aporntewan Prabhas Chongstitvatana. Improving correctness of finite-state machine synthesis from multiple partial input/output sequences..

[11] Jason W. Horihan and Yung-Hsiang Lu. Improving fsm evolution withprogressive fitness functions. In GLSVLSI ’04: Proceedings of the 14thACM Great Lakes symposium on VLSI, pages 123–126. ACM Press,2004.

[12] Lamine Ngom, Claude Baron, and Jean-Claude Geffroy, "Genetic simulationfor finite state machine identification" In SS ’99: Proceedings of the Thirty-Second Annual Simulation Symposium, page 118, Washington,DC, USA, 1999. IEEE Computer Society.

[13] Pushmeet Kohli, "A new genetic algorithm based scheme for inferringfinite state machines from accept/reject data samples", In IICAI, pages632–645, 2003.

[14] Philip Hingston, " A genetic algorithm for regular inference" In Lee Spector,Erik D. Goodman, Annie Wu, W. B. Langdon, Hans-Michael Voigt,Mitsuo Gen, Sandip Sen, Marco Dorigo, Shahram Pezeshk, Max H. Garzon,and Edmund Burke, editors, Proceedings of the Genetic and EvolutionaryComputation Conference (GECCO-2001), pages 1299–1306, SanFrancisco, California, USA, 7-11 2001. Morgan Kaufmann

[15] Xiaojun Geng, "Solving identification problem for asynchronous finite state machines using genetic algorithms", In GECCO ’06: Proceedingsof the 8th annual conference on Genetic and evolutionary computation, pages 1413–1414, New York, NY, USA, 2006. ACM Press..

[16] M. T. Tu, E. Wolff, and W. Lamersdorf, " Genetic algorithms for automated negotiations: A FSM-based application approach", In DEXA ’00:Proceedings of the 11th International Workshop on Database and Expert Systems Applications, page 1029, Washington, DC, USA, 2000. IEEE Computer Society.

[17] Simon M. Lucas, " Evolving finite state transducers: Some initial explorations.In Euro GP", pages 130–141, 2003.

[18] Simon M. Lucas and T. Jeff Reynolds, " Learning finite state transducers: Evolution versus heuristic state merging", 2007.

[19] J. Amaral, K. Turner, and J. Ghosh., "Designing genetic algorithms for the state assignment problem", 1995.

[20] M. Lankhorst., "A genetic algorithm for induction of non deterministic pushdown automata.",

[21] Simon Lucas., "Structuring chromosomes for context-free grammar evolution", In Proceedings of The IEEE Conference on Evolutionary Computation ,IEEE World Congress on Computational Intelligence, 1994.

[22] William M. Spears and Diana F. Gordon., "Evolving finite-state machine strategies for protecting resources". In International Syposium on Methodologies for Intelligent Systems, pages 166–175, 2000.

[23] A. Al Salami N. M. A., "Evolutionary Algorithm Definition", American J. of Engineering and Applied Sciences 2(4): 789-795, 2009.

[24] AL-Salami, N.M.A., "System Evolving using Ant Colony Optimization algorithm.", J. Computer. Science, 2009 5 (5): 380-387. http://www.scipub.org/fulltext/jcs/jcs55380-387.pdf.

[25] Al Salami N. M. A., "Genetic System Generation", WEC 2009, pp 23-27.