AUTOMATIC GENERATION OF FINITE AUTOMATON BY GENETIC ALGORITHMS

Toshihiko Ono and Tomoyuki Yokoi
Fukuoka Institute of Technology
3-30-1 Wajiro-higashi, Higashi-ku, Fukuoka, 811-0295, Japan
e-mail: ono@cs.fit.ac.jp, http://www.fit.ac.jp/~ ono/

ABSTRACT

This paper presents a new method to generate a deterministic finite automaton automatically as to satisfy the acceptance and rejection conditions given by sample data. The systems consist of genetic algorithms and automaton construction algorithms, together with integration algorithms, minimization algorithms, and automatic drawing algorithms. The automaton construction algorithms set up an automaton according to the information given by the genes with the genetic algorithms, test the produced automaton with the sample data, and return the results to the genetic algorithms as a fitness value. When the sample data are given by the logical combination of various data, each automaton obtained by the above method from each datum is integrated into an automaton by the integration algorithms, and the minimum automaton is obtained by deleting isolated and redundant nodes by minimization algorithms.

1 INTRODUCTION

A deterministic finite automaton (DFA) is a mathematical model expressing the operation of discrete systems and is frequently used as a design tool for computer systems, automatic machines, and the like. In designing an automaton, the requirements are usually given by sample data consisting of a group of input signals to be accepted and rejected. Since the problem of setting up the automaton is NP-complete [Gold 78], it requires a lot of trial and error search and thus, various methods have been studied.

Tomita [Tomita 82] proposed the hill-climbing search algorithms. He tested them using seven kinds of automata and compared the results with those obtained by an exhaustive search method. These seven kinds of automata have been adopted by many researchers as standard test data to identify performance. Giles et al. [Giles 90], Pollac [Pollac 91], and Watrous and Kuhn et al. [Watrous 92] studied the use of recurrent neural networks. Dunay et al. [Dunay 94]researched the method based on genetic programming. Brave [Brave 96] published results gained by a cellular encoding method. Dupont [Dupont 94] researched genetic search method and Angeline et al. [Angeline 96] proposed an evolutionary programming method.

Even though there have been a lot of studies as explained above, there still remain many problems which need to be investigated further. For example, when a complex automaton is to be identified, applying only a searching method would not be appropriate. Instead, more effective indentification could be attained by combining various methods. Considering these situations, we propose here a method which induces a DFA by integrating small automata obtained by genetic algorithms from each requirement constituting the originals.

2 SYSTEM CONFIGURATION

Fig.1 shows the schematic diagram of the systems which consist of two parts: the GAs and the automaton construction algorithms (ACAs). As the number of nodes of the automaton can not be determined beforehand, it is determined by a trial method in which from a minimum of two nodes, the number is increased towards the predetermined maximum value until the satisfactory automaton is obtained by the operation of the node number determining part.

In the GAs, the genetic operations consisting of selection, crossover, and mutation are performed. Those operations are specially designed to suit the application

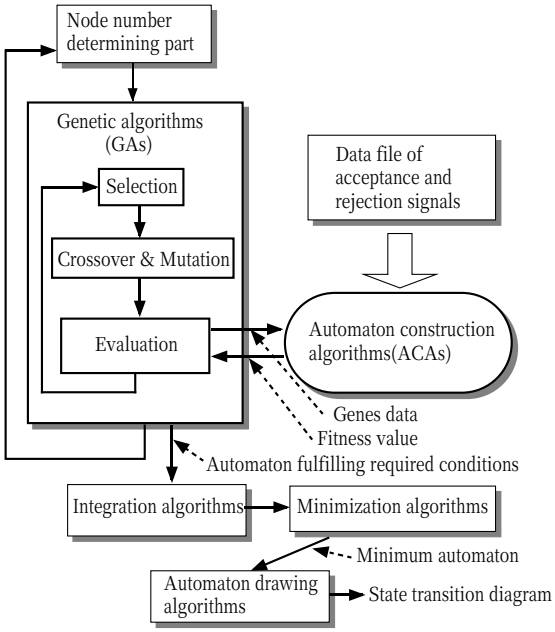

Figure 1: Schematic diagram

for constructing an automaton. In the ACAs, gene information given by the GAs is interpreted, the automaton is constructed accordingly, and the fitness value is calculated by referring to an acceptance and rejection data file defining the satisfactory conditions and then returned to the GAs.

When the automaton should satisfy complex requirements consisting of various requirements, it is not appropriate to try to build it by the above-mentioned method only, considering the vast search space required. The requirements are divided into small parts and the automata, each of which is induced from each requirement, are integrated into an automaton.

Since the automaton obtained by the above method frequently has redundant and isolated nodes, these are eliminated by the minimization algorithms to make an automaton of a minimum number of nodes. Finally, with the automaton drawing algorithms, the schematic diagram of the automaton is drawn automatically according to the information of the genes.

3 GENETIC ALGORITHMS

When we apply the GAs to the automatic configuration of automaton, it is important to determine the expression of genes suitable for the problem and to adopt appropriate genetic operations such as selection, crossover, and mutation.

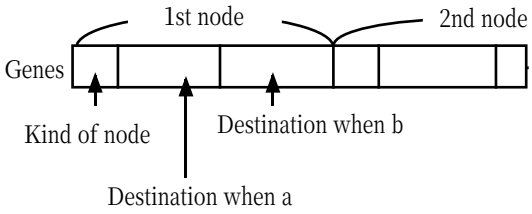

Figure 2: Representation of genes

3.1 Gene expression

Since genes should have necessary information to construct an automaton, each gene corresponding to each node of the automaton should explain the following data:

1. Identification number of each node
2. Two identification numbers of the destinations to proceed , according to the kind of input charactor
3. Indication of whether it is an accepted or rejected node

Considering these requirements, in case of the input of binary bit string consisting of the charactors 'a' and 'b', we have decided the expression of genes as shown in Fig.2. The genes consist of the sections equal to the number of nodes, each of which is comprised of parts expressing the kind of node, the destination node when accepted, and the one rejected.

3.2 Selection operator

As a selection operator, the rank selection method is used together with an elite preserving method. The selection probability is determined proportional to the rank of object value and the best individual is always inherited to the next generation.

3.3 Crossover and mutation operator

A two point crossover is adopted with the crossover point selected from the borders between adjacent genes to avoid destruction of genes. A mutation is performed by changing each bit of the genes randomly to the opposite one.

4 AUTOMATON CONSTRUCTING ALGORITHMS

The overall operation of the ACAs is shown in Fig.3, which sets up an automaton according to the gene's

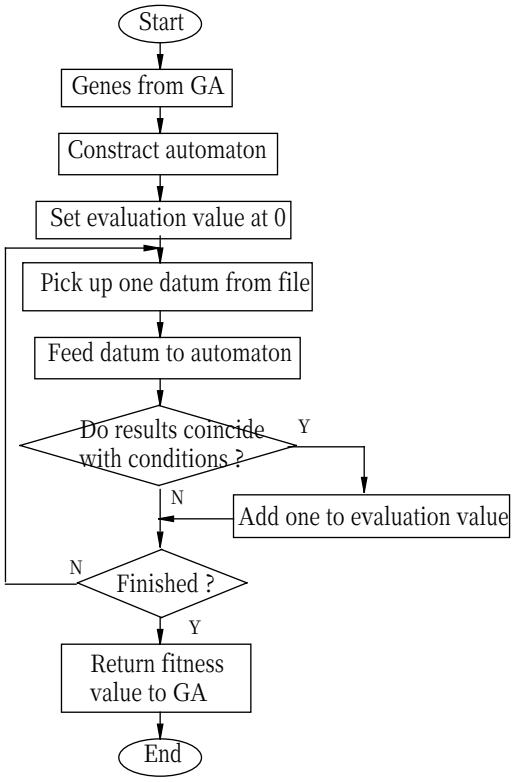

Figure 3: Flowchart of ACA

information and calculates the fitness value. Since the necessary data to determine the automaton in the GAs are given by the genes, the systems have to specify the following information at each node:

1. Whether it is an accepted or rejected node.

2. The destination of each node, according to each character in the input string.

From these data given by the genes, an automaton is constructed.

The fitness value is calculated to indicate how the obtained automaton satisfies the required conditions. To do this, an input data file is used in which every possible sequence of input string and output result is defined. More specifically, the data file contains every possible combination of binary bit string, whose length is from the minimum length two, to the predetermined maximum one, along with the result "accepted" or "rejected".

The fitness value is calculated as follows, using the evaluation value initially set at zero.

- Step 1. Pick up one datum, that is one string, from the data file.

- Step 2. Feed the datum to the induced automaton and obtain the result of acceptance or rejection.

- Step 3. If this result coincides with the one defined in the data file, add one to the evaluation value.

- Step 4. Repeat steps 1 to 3 until all the data in the data file are processed.

- Step 5. Calculate the difference between the evaluation value and the number of data in the data file, and return it to the GAs as a fitness value. If the difference is zero, the desired automaton has been obtained.

5 INTEGRATION ALGORITHMS

When the FDA to be identified is large and complex, the above-mentioned method only is not sufficient, because the search space for the optimal solution becomes wide and the time required for processing becomes large. For example, this will happen when the input signal is given by the logical combination of various strings. As a solution to this problem, we have adopted a method to set up a complete automaton by combining small automata induced from every element of input string with integration algorithms. The integration algorithms are based on the compounding theorem of automata explained in the appendix. Fig.4 shows one example of the integration of automata.

6 MINIMIZATION ALGORITHMS

Since the automaton produced by the GAs and the integration algorithms can sometimes contain isolated and redundant nodes, minimization algorithms are included in the systems to obtain the automaton with the minimum number of nodes. The main functions of the algorithms are as follows:

1. Remove isolated nodes that are not reachable from any other node even when every possible input signal is applied.

2. Replace all equivalent nodes with one node.

To find equivalent nodes, a table is provided and the results of the operation of the automaton are written out for every possible pair of nodes. Take one pair of nodes and make trees starting from each node. If all the descendants in the trees have the same values at every input, these two nodes are concluded as equivalent and replaced by one node. Fig.5 shows one example of the minimizing operation.

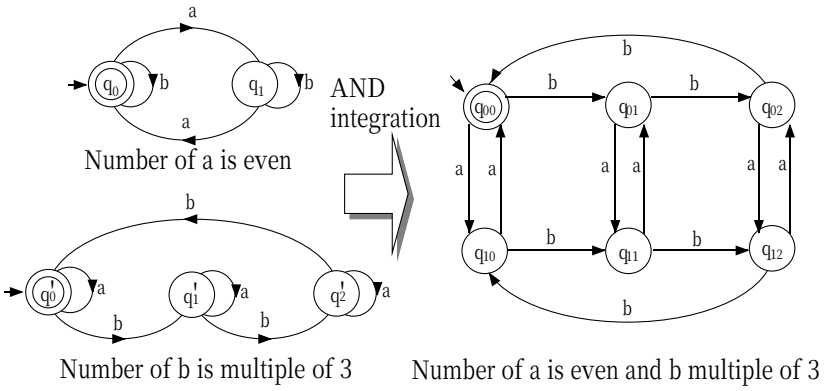

Figure 4: Example of integration

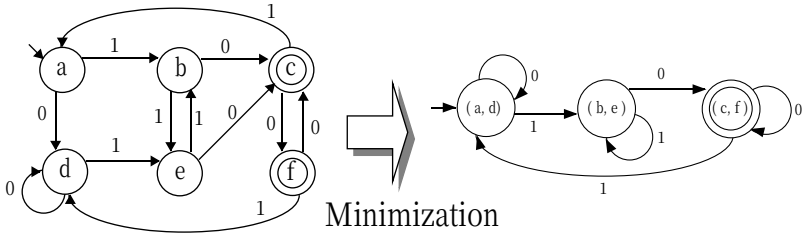

Figure 5: Example of minimization

7 SIMULATION STUDIES

To investigate the performance of the proposed method, various simulation tests have been executed using a Sun Sparc Classic type workstation. The first simulation was done to confirm that a satisfactory automaton is obtained by this method for seven kinds of automaton which accept each of the following input strings.

- Case-1: b*.
- Case-2: (b a)*.
- Case-3: no odd-length a-string anywhere after an odd-length b-string. For example aaabaabb.
- Case-4: any string not containing bbb. For example, bbabbaa.
- Case-5: bit pairs, the sum of number of (ab)'s and number of (ba)'s is even. For example aabbaaaba.
- Case-6: the difference of the numbers between b's and a's is a multiple of three. For example, abbaabbbb.
- Case-7: a*b*a*b*. For example aababb.

In the simulation, the number of nodes is set at two initially and is increased one by one up to the preset maximum value, if the automaton that satisfies the input conditions is not obtained until 150 generations. The population was set at 100, with the probability of crossover at 0.7 and the probability of mutation at 0.02. The input data file that defines the input conditions consists of every possible binary bit string of length up to 12 along with the required output of automaton, showing whether "accepted" or "rejected".

In each case, the same simulation is repeated 300 times by changing the initial conditions of each individual in the GAs with a different series of random numbers. Table 1 shows the results of simulation. The first column of the table shows the case number, the second the number of simulations in which the satisfying automaton is gained, the third column the probability of success which is calculated by the successful number of simulations divided by the number of simulations, the fourth column the number of nodes obtained by simulation, the fifth column the minimum generation when the satisfactory automaton is obtained, the sixth the maximum generation, and seventh the mean value of generations. With these simulations, the automaton satisfying the required conditions is acquired

Table 1: Result of simulation

No.	suc. counts	suc. prob.	nodes	min.	max.	mean
1	300	100.0	2	0	2	0.05
2	300	100.0	3	8	192	24.3
3	300	100.0	5	255	2530	829.0
4	299	99.7	4	69	2782	300.9
5	300	100.0	5	322	3914	1454.5
6	300	100.0	3	8	2252	166.4
7	300	100.0	5	340	3243	1327.1

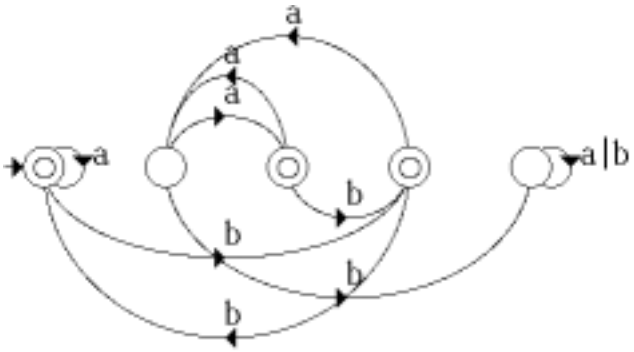

Figure 6: Created automaton in Simulation-3

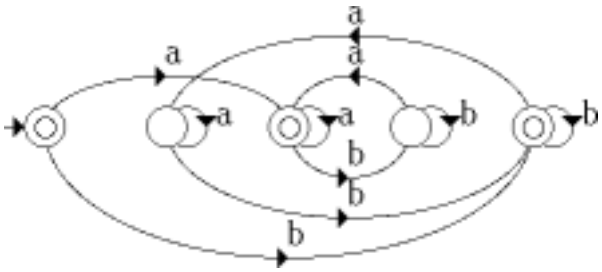

Figure 7: Created automaton in Simulation-5

in all cases except for Case-4, in which in all except one among 300 simulations the satisfactory automaton is obtained. Fig.6 and Fig.7 show two examples of the state transition diagram of the thus obtained automaton for Case-3 and Case-5, drawn automatically by the drawing algorithms.

The second simulation was done to investigate the performance of the integration algorithms. The object was to make an automaton that can accept both of the following two input strings:

1. Input signals given by Case 3.
2. Ditto given by Case 6.

Two automata that were obtained separately by the GAs to satisfy each of the above requirements were integrated into the minimum automaton by integration algorithms and minimization algorithms, as shown in Fig.8.

8 CONCLUSION

We have proposed a method to obtain the minimum automaton to satisfy input requirements which specify acceptance and rejection conditions, using genetic algorithms together with automaton construction algorithms, minimization algorithms, and drawing algorithms to draw a state transition diagram.

Through the simulation studies, the performance of the systems has been confirmed. The proposed method will be applicable to other type of automaton, such as a push down automaton.

REFERENCES

[Giles 90] Giles, C. L. et al.(1990). High Order Recurrent Networks & Grammatical Inference. *Advances in Neural Information Processing*, Vol.2, Morgan Kaufmann, pp.380-387.

[Angeline 96] Angeline, P. J. et al.(1996). A Comparison of Self-Adaptation Methods for Finite State Machines in Dynamic Environments. *Evolutionary Programming V; Proceedings of the Fifth Annual Conference on Evolutionary Programming*, pp.441-449.

[Brave 96] Brave, S.(1996). Evolving Deterministic Finite Automata Using Cellular Encoding. *Proceedings of the First Annual Conference of Genetic Programming*, pp.39-44.

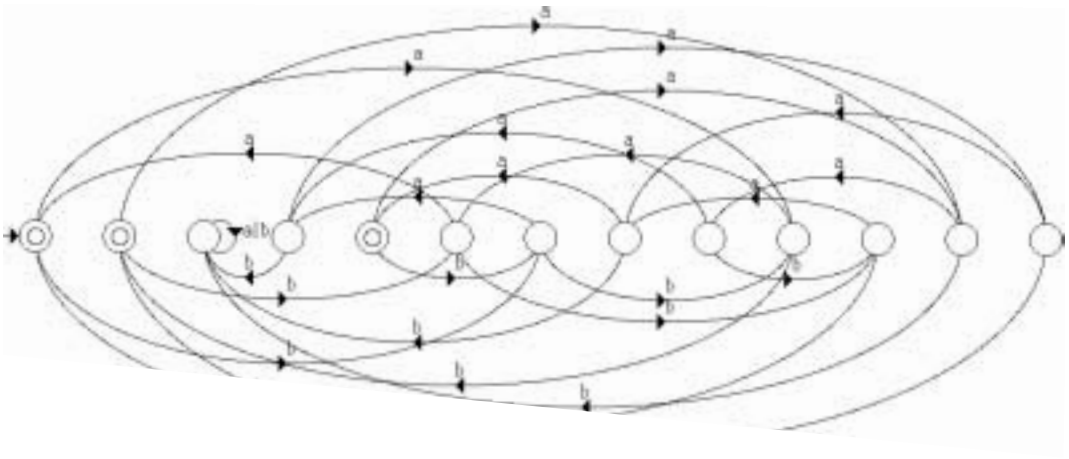

Figure 8: Created automaton by integration and minimization

[Dunay 94] Dunay, B. D. et al.(1994). Regular Language Induction with Genetic Programming. *Proceedings of the First IEEE Conference on Evolutionary Computation*, Vol.1, IEEE Prress, pp.396-400.

[Dupont 94] Dupont, P.(1994). Regular Grammatical Inference from Positive and Negative Samples by Genetic Search : the GIG method. *Proceedings of the Second International Colloquim*, pp.236-245.

[Giles 90] Giles, C. L. et al.(1990). High Order Recurrent Networks & Grammatical Inference. *Advances in Neural Information Processing*, Vol.2, Morgan Kaufmann, pp.380-387.

[Gold 78] Gold, E. M. (1978). Complexity of Automaton Identification from Given Data. *Information and Control*, Vol.37, pp.302-320.

[Lipschutz 95] Lipschutz, S. (1995). *Theory and Problems of Discrete Mathematics*, Japanese edition, Ohmsha,Ltd.

[Pollac 91] Pollac, J. B.(1991). Language Induction by Phase Transition in Dynamical Recognizer. *Advances in Neural Information Processing*, Vol.3, Morgan Kaufmann, pp.619-626.

[Tomita 82] Tomita, M.(1982). Dynamic Construction of Finite Automata From Examples Using Hill-Climbing. *Proceedings of the Fourth Annual Coginitive Science Conference*, pp.105-108.

[Watrous 92] Watrous, R. L. and Kuhn, G. M.(1992). Induction of Finite-State Languages Using Second-Order Recurrent Networks. *Nural Computation*, Vol.4, pp.404-414.

APPENDIX

[Compounding theorem] [Lipschutz 95]

Let $M =< A, S, T, q_0, f >,\ M' =< A, S', T', q_0', f' >$ be two automata having the same sets of input strings A, S and S' the sets of states, T and T' the sets of accepted states, q_0 and q_0' the initial states, and f and f' the state transion functions. When $L(M)$ and $L(M')$ indicate the input strings accepted by the automata M and M', the automaton N that accepts the set of input strings expressed by $L(M) \cap L(M')$ is given by the followings:

The set of states of N is given $S \times S'$. When the accepted states of M and M' are expressed by q and q', the state of N is given by $(q,\ q')$ and the initial state of N is $(q_0,\ q_0')$. The state transition function of N

$$g : ((S \times\ S') \times A \rightarrow (S \times S')$$

is defined as

$$g : ((q,\ q'),\ a) = (f(q,\ a),\ f'(q',\ a)).$$