International Journal of Knowledge-based and Intelligent Engineering Systems 15 (2011) 43–53
DOI 10.3233/KES-2010-0207
IOS Press

The identification of discrete-event dynamic systems based on the evolutionary programming

Jose Aguilar*
Universidad de Los Andes, Facultad de Ingeniería, Departamento de Computación, CEMISID Núcleo la Hechicera, Mérida 5101, Venezuela

Abstract. The present work evaluates the use of the evolutionary programming in problems of Industrial Automation. It is well known that technical of the evolutionary calculation as the Genetic Algorithms have provided satisfactory results when facing problems of automatic control, especially of continuous type, while the study of discrete dynamics have been relegated. In this work, an identification technique based on the evolutionary programming is proposed for Discrete-Event Dynamic Systems (DEDS), using finite state machines (like the machines of Mealy).

Keywords: Identification system, evolutionary programming, discrete-event dynamic systems

1. Introduction

The systems designed by the man present dynamic highly not lineal that cannot be modeled for continuous dynamic in many cases, reason why it is necessary to use Discrete-Event Dynamic Systems (DEDS). These systems are managed through transitions. When are modeled continuous dynamics, they are attached to physical laws. In the case of the discrete events dynamics, the models are formulated according to the experience of the "engineer", and frequently it is necessary to take into account some considerations and/or restrictions to obtain a model to events based on some points of view.

A Discrete-Event System (DES) describes the behavior by means of the enumeration (or occurrence) of events from an initial state, or by means of the history of the changes of states happened starting from an initial state. Several ways exist of studying the behavior of the DEDS. Particularly, we will remit ourselves to work with Sequential Machines.

On the other hand, the identification of systems tries to follow the behavior of inputs-outputs of an unknown plant using an appropriate model. In the continuous dynamic, the behavior of an unknown plant can be modeled as a non lineal function of the input sign, the sign of previous inputs and the previous outputs of the plant. The purpose is to select a model that can reproduce the behavior of the plant exactly.

Some intelligent techniques as the Neural Networks, the Fuzzy Logic and the Evolutionary algorithms [1, 7–9] have been used to create models of continuous plants, and this way to solve a great quantity of identification problems. But until now, the intelligent techniques applied to the identification problem of discrete dynamic are very limited.

Some interesting recent works in the domain of identification problems are: In [6] they present a survey of the state-of-the art about the identification of DES within the framework of Petri nets (identification of discrete event systems using Petri nets) [11] defines the identification problem for DES as the problem of inferring a Petri Net model using the observation of the events and the available output vectors, that correspond to the markings of the measurable places. Two cases

*Corresponding author. E-mail: aguilar@ula.ve.

ISSN 1327-2314/11/$27.50

are studied considering different levels of the system knowledge. In the first case the place and transition sets are assumed known. In the second case the transition and place sets are assumed unknown and only an upper bound of the number of places is given [12] presents a comparative study of identification approaches of DES. It focuses on three different approaches described in recent publications: (i) a progressive identification approach in which several algorithms have been proposed allowing the online identification of concurrent DES, (ii) an offline input-output approach, in which, through an efficient technique oriented to fault diagnosis, it is obtained a nondeterministic FA representing exactly the observed behaviour, (iii) and an offline approach based on an integer linear programming (ILP) technique, which leads to free-labelled PN models representing observed sequences. In [4] is presented a systems identification method, for discrete time linear systems, based on an evolutionary approach, which allows achieving the selection of a suitable structure and the parameters estimation, using non conventional objective functions. This algorithm incorporates parametric crossover and parametric mutation along a weighted gradient direction. In [16] they focus on the identification of large-scale DES for the purpose of fault detection. The properties of a model to be useful for fault detection are discussed. As appropriate model basis the nondeterministic autonomous automaton is chosen and metrics to evaluate the accuracy of the identified model are defined. An identification algorithm which allows setting the accuracy of the identified model is presented. In [10] they propose a new mathematical framework in the context of a Mealy machine and language theory, for system identification (SI) for DES. They part of the idea of identifying system dynamics from externally observed sample paths. The objective of SI is to derive minimal valid automata that duplicate the input-output relation in the observed sample paths, while ensuring minimal realization. An algorithm to compute minimal valid automata and special properties regarding SI are explained with illustrative examples. Real time identification and dependence of the expert knowlegement about the real system for the identification model are several of the limitations of these works, which we try to solve using intelligence techniques.

The purpose of this work is to apply a technique of the Evolutionary Computing to the study of Discrete-Events Dynamic Systems (DEDS). The technique to use is the Evolutionary Programming, and it is not more than a search and optimization method. The idea is to make evolve a group of individuals, each one representing a model of DES that in turn is a possible solution of the system that we like to identify. This way, through a finite number of generations these systems should adjust to the environment. That is, the Evolutionary Programming proposes a group of finite state machines that reproduce the general behavior of the real system.

This work is organized in the following way: An introduction is made to the theoretical aspects that are used in this paper. Then, the proposal is presented, and a methodology is exposed to use it in identification processes. Finally, some results are presented, particularly the identification of a Real System is analyzed.

2. Theoretical aspects

2.1. Discrete-event systems and identification problem

It is a system where its states space is a discrete set which is managed by events, such that its evolution of states depends exclusively of the asynchronous occurrence of discrete events in the time [18]. If we consider a DES as a generator of a formal language, we can introduce the characteristic of control when taking into account certain events or transitions that can be disabled by an external controller. But before being able to control any system, first we must have a model.

It is not always possible to obtain that model analyzing the process, it can be mathematically difficult, or not all the parameters of the real system can be known. Then, we need to carry out the system identification. The problem of Identification refers to have a black box, where we can only observe the change of the output due to the change of the input [7–9]. The objective is to find a description of the behavior that is observed. If we don't have some idea of what could be in the box, the identification task is more difficult, for example, it is more complex to interpret the obtained data. In real situations, what is usually made is to prove with a finite number of experiments and to increase more and more the complexity of the model, until being able to explain or to justify the resulting data.

2.2. Finite state machine (FSM) and systems identification

A FSM is a transducer which operates on a finite group of input symbols (alphabet), it possesses a finite number of internal states, and it produces output symbols from a finite alphabet [3,17]. It is essentially a computer program: it represents a sequence of instruc-

Table 1
Table of transition

Current state/ next state	A	B	C
A	1 /β	0 /β	
B		0 /δ	1 /α
C	1 /δ	0 /β	

tions to be executed, where each instruction depends on the current state of the machine and of the current stimuli.

The possible inputs to the system are a sequence of symbols selected from a finite group I of input symbols, and the resulting outputs are sequences of symbols chosen from a finite group Z of output symbols. More formally, a FSM is described like:

$$M = (Q, I, Z, S, O)$$

Where: Q is the group of states; I is the group of symbols of input; Z is the group of output symbols; $S(QxI) -> I$ is the function of the next state; $O(QxI) -> Q$ is the function of the next output. The behavior of a FSM is described as a sequence of events that happen in discrete instants $t = 1, 2, 3$, etc.

Two techniques are commonly used to represent the analytic properties of a machine: the transition diagrams and the transition tables. Next, the Table 1 presents an example of a transition table for a machine with three states (A, B, C) whose alphabet of input symbols is 0 and 1, and the alphabet of output symbols is α, β and δ. For example, in this machine when the system is in the state A, and there is an input 0, the system change to state B and its output symbol is β; an so for the rest of states.

Generally, to carry out the identification of a FSM we settle down a group of conditions that should be satisfied to be able to determine the properties of the unknown machine. First, it is assumed that the whole group of input symbols is defined, if this information is not possessed it cannot be defined the transition functions completely (output and next state).

The final condition that should be completed to identify a FSM is that this should be strongly connected. The problem of a machine that is not strongly connected resides basically in the fact that it is not possible to obtain a sequence that evolves through the different states of a machine. Two examples of FSM do not strongly connected are when the machine possesses special states (recurrent, etc.), or when there are states non-connected (the FSM possesses one or more unreachable states). This makes that the system is not globally controllable. In some works that present methods to identify machines, besides of the previous restrictions, they also restrict the identification to machines that have not equivalent states.

The identification of FSM has been carried out through experiments, which require that the engineer makes use of its knowledge of sequential machines and DES to select the appropriate input sequence in each stage of the identification. The basic idea is to apply a sequence and to observe the answer of the machine. Then, another input sequence is applied and the answer is used to extend or to eliminate the previously formed partial state diagrams. This way the process continues until is obtained a totally defined state diagram that describes the machine that we search.

2.3. Evolutionary computing

Under the term of Evolutionary Computing it is included to a wide group of technical of resolution of complex problems based on the emulation of the natural processes of evolution [1,5,13–15]. That is, these techniques use as elements some mechanism of the theory of the evolution for their design and implementation.

The main contribution of the Evolutionary computing to the methodology of resolution of problems consists on the use of mechanisms of selection of solutions potentials, and of construction of new candidates' for recombination of characteristic of other already present, in a similar way to like it happens in the evolution of the natural organisms.

The purpose of the Evolutionary Algorithms is to guide a stochastic search making evolve a group of structures and selecting iteratively the most capable. An Evolutionary Algorithm is executed on a population of individuals that represent to a group of candidates solutions of a problem; these individuals are subjected to a series of transformations and later to a selection process that favors the best individuals. It is expected from the Evolutionary Algorithm that after certain number of generations (iterations) the best individual is reasonably close to the searched solution.

The Evolutionary Operators are those that carry out the population's changes during the execution of an Evolutionary Algorithm. Classically, two genetic operators exist: mutation (it is an operator unary that simulates the evolutionary process that happens in the individuals when they change their genetic structure), and crossover (it is usually a binary operator that allows to represent the processes of natural mating. they take diverse components of different individuals to generate with them new individuals). However, there are other operators, for example: Dominance, Segregation, among others. The steps of an Evolutionary Algorithm are the following:

– Generation of an initial population, generally randomly.
– Evaluation of the individuals
– Selection of some individuals.
– Modification of the genes of the selected progenitors using the genetic operators.
– Generation of a new population.
– Verification of the convergence of the Algorithm, or returns to the step 3, if it is the case.

The main techniques of the Evolutionary Computation are [1]: the Genetic Algorithms, the Evolutionary Strategies, the Genetic Programming, and the Evolutionary Programming.

2.3.1. Evolutionary Programming (*EP*)

The Evolutionary Programming was originally conceived by Lawrence J. Fogel in 1960 [1,13–15]. It is a mechanism that makes evolve a group of finite state machines. This technique develops a group of solutions which show a good behavior with respect to an environment and an objective function. The steps of the EP are:

– InitPopulation P(t): It begins with a population of finite state machines that represent solutions to the problem in question.
– Evaluation of P(t): Each machine is evaluated by means of a specific function that calculates the individual's capacity to solve the problem.
– Mechanism of Selection: The selection of the machines that will become parents of the next generation is made, that is, all the individuals are selected.
– P'(t) = mutate (P(t)): the population's member is altered through a mutation process, the crossing is not used. That is, each machine generates a descendant through a mutation process that is applied on it Classically five types of random mutations exist:
  * To change an output symbol.
  * To change the transition a state to another.
  * To add a state.
  * To eliminate a state.
  * To change the initial state.
– $P(t+1)$ = survive (P'(t)): the individuals are chosen that will survive for the following generation. Normally, we chose a number k of the best individuals of the total machines population. The population's size generally remains constant, but there is not restriction in this case.
– Stopping Criterion: The process finishes when the solution reaches a predetermined quality, when a specific number of iterations has been obtained (generations), or some other stop approach.

3. The evolutionary programming in the identification of dynamic systems

In this section, our approach to obtain discrete event models is presented. We require two phases: a) Characterization of the system according to the EP: in this first part the structure of the individuals is designed, the parameters values and the cost function are defined, as well as the initials population. b) Search of the FSM using the EP: in this phase the EP searches the FSM that better follows the behavior of the real system. This is a global search through different generations, treating to avoid local optimals.

3.1. Characterization of the system according to the EP

3.1.1. Design of the Structure of the Genotype

The Genotype in the Evolutionary Programming represents a FSM, it should also be able to generate an output sequence due to one input. This requirement is indispensable for the fact that we like to make an identification process. Two models of finite state machines can be used (or two forms of sequential machines) [3, 17]: Mealy and Moore.

We used the Mealy model by their computacional perfomance, since it requires generally less states than the Moore model to recognize an input signal and to generate the output corresponding. As we require working with great populations, then we do not like to diminish the search speed due to the introduction of very complex individuals.

The model of Mealy is represented by the following expression:

$$M = (Q, I, Z, S, O, q(0))$$

Where: Q is the group of states; I is the group of input symbols; Z is the group of output symbols; $S(QxI) -> Q$ is the function of the next state; $O(QxI) -> Z$ is the function of the next output; $q(0)$ is the initial state.

A machine of Mealy only generates an output symbol when an input symbol is presented. The procedural semantics of the model of Mealy is the following: the machine is in a state $q(0)$, when it receives a literal of

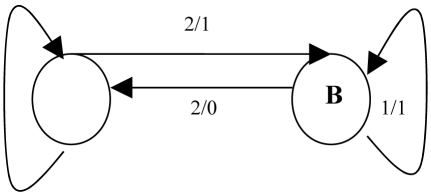

Fig. 1. Diagram of transition of a machine of mealy.

Table 2
Table of transition of a machine of mealy

States / inputs	1	2
A	A/0	B/1
B	B/1	A/0

input, then it emits the output symbol and it passes to the new state. An example of a machine of Mealy is shown in the Fig. 1 (the transition diagram) and in the Table 2 (the table of transition), respectively.

This table is encoded to describe each individual of the EP algorithm how a FSM.

3.1.2. Initialization of the machines

This process is only executed once, and it is at the beginning of the algorithm. In the system two ways were implemented of initializing of the population of finite state machines:

- Randomly: an initial state is assigned, later it is chosen a value that will be the number of states that possesses the machine that should be smaller or similar to the maximum number of possible states. Then, we begin to fill each one of the cells of the Table of transitions; the quantity of columns is previously defined, since it is the group of input symbols. In each state with transition symbol intersection is the cell that indicates the state will be passed and which will be the corresponding output symbol. These two elements are also generated randomly. Lastly, a vector is believed that will contain the labels respective to identify the states. This vector cannot contain repeated elements (repeated states), and is updated together with the Table of transitions when a mutation is made on the individual (only if the mutation is to add or to eliminate a state).
- The other way is to initialize it from a population existent.

3.1.3. Mutation

The five mutations are used randomly.

3.1.4. Costs function

It possesses two attributes that are two chains of characters, the input sequence and the output sequence that are shared for the population's individuals. The capacity of each machine is measured comparing the output sequence that the individual generates with the sequence of the output reference, when it is faced to the input sequence. This procedure is executed in the following way: the first input symbol is applied over the individual that is in an initial state, if that symbol is not recognized by the FSM it is penalized with a big positive value. Otherwise, if the symbol is recognized the output symbol generated is compared with the symbol of output reference. If the symbols are different the individual is penalized with a small positive value; but, if the output symbols coincide the scheme is rewarded with a negative value. Then, there is a transition of state and we repeat the process for the following input-output symbol until we reach the end of the sequences. According to our cost function those individuals with more negative values will be the most capable in the generation.

We also incorporated in the cost function the capacity to reward those FSM that have fewer states. This is optional and it is with the purpose of obtaining FSM that not only follow an input sequence, but that also are simple (from the point of view of the size).

3.2. Search of the FSM using the EP

Next, the process to search FSM of real systems is described. We consider a set of steps that are going to allow the identification of a real system.

- Obtain the succession of events of the DEDS to identify. It is the fundamental step of all process of identification of dynamic systems (continuous or discrete); the sequence should be the most representative possible.
- Analyze the complexity of the system to identify, in order to determine if it is necessary to modulate in different phases the identification process. Each phase search to adapt the individual to different patterns, some phases more drastic than others, but with the same purpose, the important is that each new phase is a succession of the previous ones, incorporating new characteristics.
- Begin to generate prototypes that are possible results, of phases, or of the totality of the experiment.

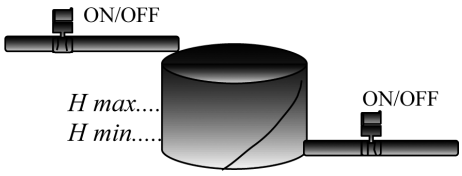

Fig. 2. Real system.

– It is advantageous to make first tests where we are interested that the individual recognizes the entirety succession of events (input signs), without caring for the quantity of states that this possesses.
– Then, to initialize the population with the resulting individuals of the previous step, but in this new execution of the algorithm we fix that the machines with less states are rewarded, in order to eliminate their redundant states.
– Verify the resulting models, if they don't present satisfactory behaviors to return to the step 3, or to analyze the input and output sequences to determine the possibility to make the evolution with new data (return to the step 1).
– If the individuals are the sufficiently capable, in this step they should be confronted to a validation sequence, with which we can analyze the behavior of the model, and this way to generate an error function. If the machines don't approve with success this phase, new prototypes should be studied.

4. Case of study

We propose a system compose by a tank where a liquid is stored and a couple of valves to its input and output that regulate the level (see Fig. 2). It is obvious that it can be modeled as a discrete events system; however we should take into account a series of specifications. Before beginning the identification process, it is indispensable to have the sequence of input and output events (input and output signs, respectively). Since we have not the plant, it is necessary to have a model to obtain some data to start the identification process; it will also serve as reference to compare it with the model to discrete events generated by the identification process (see Fig. 3). We use the EP tool proposed in [2] to built FSM.

4.1. Modeling the system

The internal behavior of the plant is described by a group of twelve (12) states, an input alphabet composed by eight (8) symbols, and an output alphabet of three (3) symbols. It is assumed that the tank can reach three states:

– Low level: it happens when the liquid in the tank is below a reference mark H(tank)<H(minimum).
– Normal level: the height of the liquid in the tank is between two reference limits, indicating that it is below of the maximum level allowed, but above of the minimum level, (minimum)<H(tank)<H(maximum).
– High level, in this state the level of liquid in the tank has overflowed the maximum height allowed, H(tank)>H(maximum).

Also, the valves are assumed of type (ON/OFF), restricting them to only two possible states. According to that, it is obtained three states for the tank and two for each valve, being twelve possible states in that the plant can be in a given moment (see Fig. 3).

The Symbols of the input Alphabet are shown in Table 3.

The Symbols of the Alphabet of Output are shown in Table 4.

The events here presented were selected since they make to pass the system for all the states mentioned. On the other hand, it is considered that the flow that

Table 3
Symbols of the input Alphabet

Input	Description
α	To empty the Tank, there is only liquid leaving but not entering
β	To fill the Tank, there is only liquid entering but not leaving
γ	To give liquid, a normal flow is wanted where the liquid continually enters and leaves
δ	To restart or to Turn off the system, they close the two valves
ε	The height of liquid in the tank passed of minimum level to normal
ζ	The height of liquid diminished, passing from the normal level to the minimum
η	The level of liquid in the tank overflows the maximum height allowed
θ	The liquid passed from the maximum height to the normal level of operation

Table 4
Symbols of the outputt Alphabet

Ouput	Description
A	After being generated the event, the height of the liquid in the tank passed at a minimum level
B	After being generated the event, the height of the liquid in the tank passed at the normal level
C	After being generated the event, the height of the liquid in the tank passed at the maximum level

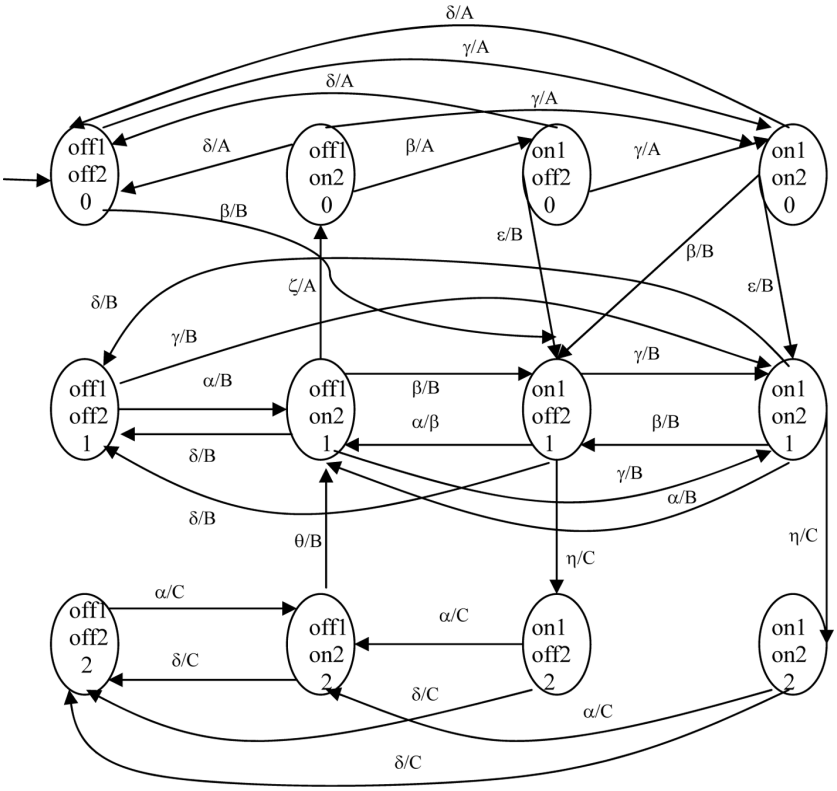

Fig. 3. Automate machine that models the Real System.

goes by the input valve is bigger to the output flow; this difference is used to allow that the FSM has events that are caused internally, as the events of level change.

The outputs can be confused with the states, but this can be used to favor. A sample would be when one obtains a model result of the identification process, it is able to give information of the level in that the liquid is (assuming that one doesn't know the internal configuration of the system), valuable information to the moment to present an external event to the system. For example, it should not empty the tank when the previous output indicated that the level is minimal.

Of the real FSM the sequences of events are acquired that will be used in the identification algorithm, besides during the validation.

4.2. Process of identification

The procedure to make the identification is based on the propose methodology in the previous section. The fundamental characteristic of the methodology is that each phase constitutes a concatenation of the previous one, and in each step the training sequences become more extensive. The reason to use this approach is because we like to find the machine the simplest. In each one of the phases we introduce new elements of the alphabet to allow that the EP learns these elements (a pattern) in that phase. In this way, we can obtain a simple FSM of a complex system.

The main parameters of the experiment are: Size of the Population = 10; Size of the Group of Input = 8 symbols; Size of the Group of Output = 3.

4.2.1. First phase

In this stage the population is initialized randomly, and the algorithm is executed according to the previous parameters, for this first case we select a sequence of events, which allows passing for some states of the system (not all). Also, it is chosen the option of gratifying the population's individuals that recognize the input word and that possess few states. The used sequence is: of Input = γ, ε, β, α, ζ, δ, γ, ε, β, α, ζ, δ, β, γ, β, α, ζ, δ, β γ, β, α, ζ, δ, β, α, ζ, β, ε, α, γ, α, ζ, β, ε, α, δ; and of Output = A, B, B, B, A, A, A, B, B, B, A, A, B, B, B, B, A, A, B, B, B, B, A, A, B, B, A, A, B, B, B, B, A, A, B, B, B, A, A, B, B, B, B, A, A, B, B, B.

One can observe that the sequence of input and of output events have not all the elements of the alphabet.

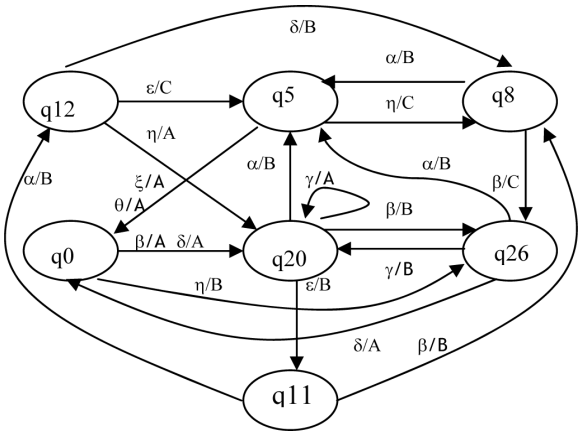

Fig. 4. Generated machine in the First Phase.

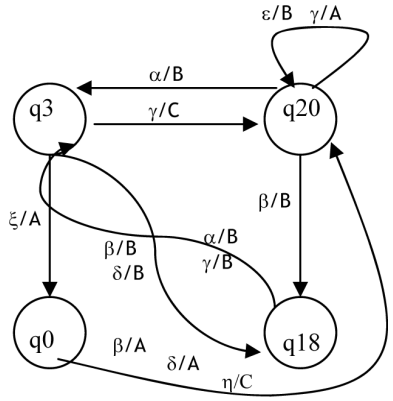

Fig. 5. Generated machine in the Second Phase.

However, the algorithm has as parameter the total number of symbols, because in the following phases we are going to incorporate the rest of the events.

In this phase we have not still used a sequence of validation events. The result of this first identification stage is shown in Fig. 4.

For this first step the machine generated by the identification algorithm reached a value of objective function of -423, in a total of 822 generations. That mean, this FSM, does not follow the FSM of the real system (see Fig. 3) and has less states and transitions because the initial sequence of events is less that from the original system. The computation effort for these 822 generations is of 45 seconds and the quality of the FSM is very good (The model generated accepts the sequence presented).

4.2.2. Second phase

For this step of the identification process, the algorithm is executed again, but in this occasion the initial population is loaded with the file that contains the machines of state finite resultants of the first part. Now, the individuals are subjected to a new sequence of events, for what we need to recalculate the value of the objective function of each machine when beginning the algorithm. The input word and their corresponding output are: Input $= \gamma$, β, α, δ, α, β, γ, β, α, δ, γ, δ, γ, δ, α, ζ,δ, γ, ε, β, α, δ, α, ζ, δ, γ, ε, β, α, δ; output $=$ B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, A, A, A, B, B, B, B, B, A, A, A, B, B, B.

The sequence of events inserted in this phase of the experiment doesn't introduce a new symbol, but it provides a different path to the studied in the first phase. It is importance to link is sequence of events to the one introduced in the previous phase, to force the individual to not forgetting the behavior acquired previously; although it can happen that the algorithm in their search of the best machine for the word inserted in this execution phase causes deteriorations achieved in the first phase.

The result of the second phase of the experiment is shown in Fig. 5, in this part the FSM possesses three states less than the machine generated in the previous step, however, if the two models are compared, it is possible to notice that the most recent machine eliminated five (5) states and added two (2) new with regard to the initial individual.

In this second phase the individuals obtained have a value of objective function of -726, in a total of 2245 generations, being a FSM "more simple" that the one obtained in the previous phase. The computation effort for these generations is of 69 seconds. Additionally, the model generated in this step accepts the sequence presented in the first phase completely. Also, it is not still pertinent to submit the model to a validation word because the experiment has not been completed.

4.2.3. Third phase

Again, the population in this phase of the experiment is initialized with the resulting individuals of the previous phase. The sequence introduces new events, two input symbols and an output symbol, with these are introduced all the elements of the input and output alphabets.

Next, we present the input and the output word, the new symbols are η, θ and C, of the input and output alphabets, respectively. Input $= \alpha$, β, δ, α, β, δ, α, ζ, δ, β, α, δ, γ, β, δ, γ, β, δ, γ, η, α, θ, δ, γ, η, α, θ, δ; Output: B, B, B, B,B, B, B, A, A, B, B, B, B, B, B,B, B, B, B,C, C, B, B, B, C, C, B, B.

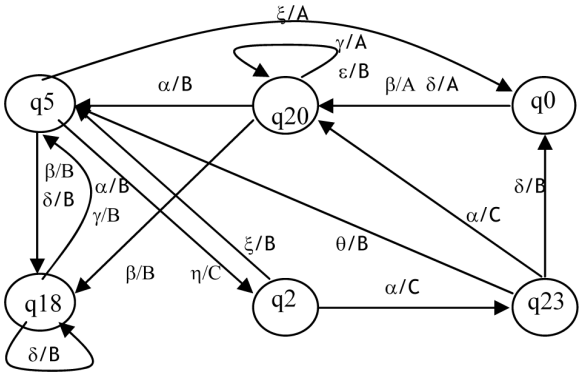

Fig. 6. Machine generated in the Third Phase.

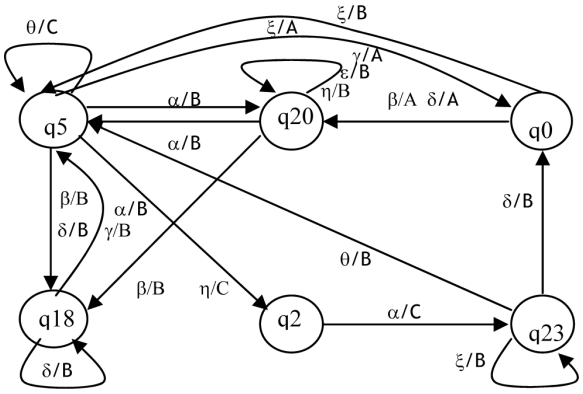

Fig. 7. Result of the Fourth Stage of Identification.

The obtained FSM behaves efficiently for the successions of events presented until the moment (Fig. 6). One can observe that the algorithm has introduced three new states (q23, q2, q5) and one has eliminated (q3), considering this way this individual like the most capable according to the environment to which was submitted. The model obtained in this step also possess more states that in the previous phase, What did it Happen?; the fundamental mission of the developed algorithm is to get a model that reconstructs the input-output sign, without important that this implies to add more states, then those that are redundant will leave eliminating.

In this stage the FSM reach a value of objective function of -1004, after having evolved along 2457 generations. The computation effort is of 81 seconds and the FSM generated accepts almost all the sequences presented until now (the success rate is the 96%, that mean recogmize almost all the events in all the states).

4.2.4. Fourth phase

This phase was only designed so that the model that one comes studying recognizes a transition especially, not for the fact that that transition is important, but to show with which easiness and speed the FSM adapts to this sequence. The model obtained before developing this phase doesn't recognize a certain event ((α/B) when this it happens after ((γ/B), assuming that the machine is in the state q18.

A sequence of events was developed that made pass the machine until that state (this sequence is also obtained of the model). The sequence is: Input $=$ α, β, γ, α, β, γ, δ, α, β, γ, α, β, γ, δ; Output $=$ B,B,B,B,B,B,B,B,B,B,B,B,B,B. The resulting machine is shown in Fig. 7.

The resulting FSM is approximately similar to that of the previous phase, with some small discrepancies that do not deteriorate the behavior of the model, the

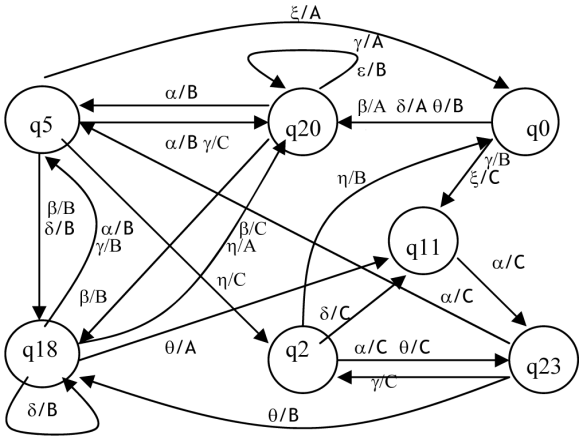

Fig. 8. Generated machine in the fifth stage.

only difference is that now it understand satisfactorily the special case that was studied in this stage. This is achieved in only 326 generations. To conclude this phase, we can see that using our approach we can submit a machine to small transformations, to achieve a specific objective, without making changes of more relevance in the behavior of the model.

4.2.5. Fifth phase

In this phase we try to improve the behavior of the model before the events introduced in the third phase, because they were handled very superficially.

The sequences used for the fifth phase are, Input: γ, η, δ, α, θ, δ, γ, η, δ, α, θ, δ; Output $=$ B, C, C, C, B, B, B, C, C, C,B, B. For the sixth is: Input: γ, η, α, θ, β, η, α, θ, β, η, α, θ, δ, γ, η, α, θ, β, η, α, θ, β, η, α, θ, δ; Output: B, C, C, B, B, C, C, B, B, C, C, B, B, B, C, C, B, B, C, C, B, B, C, C, B,B.

In the phase five the model increased their number of states at seven. The FSM of the Fig. 8 is the result of

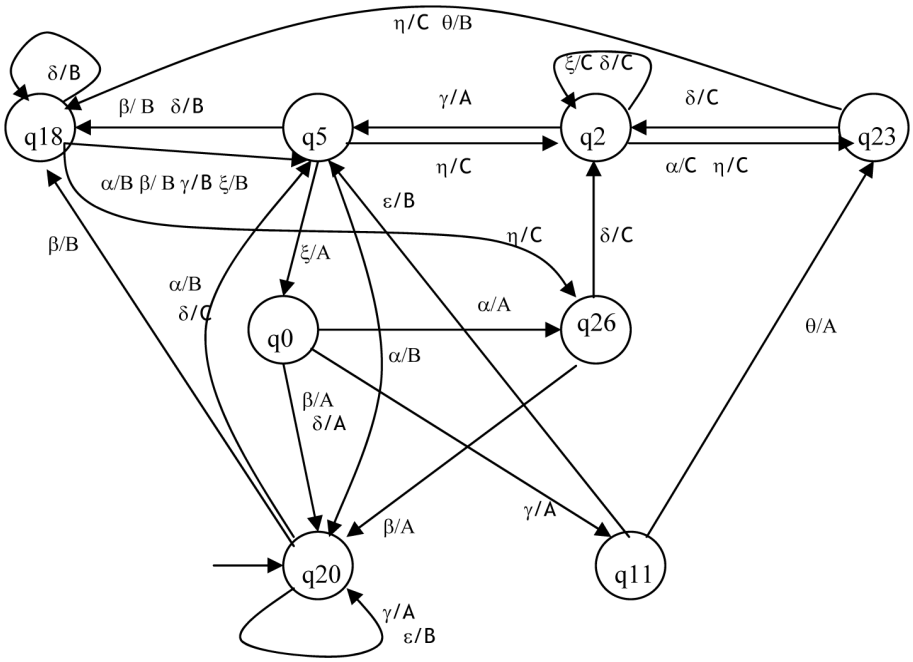

Fig. 9. Identified machine of the real system.

evolving along 1780 generations, to reach cost function of -1263. The computation effort is of 65 seconds and the FSM generated accepts all the secuences until know studied.

4.2.6. Last phase

This phase shows a sequence where all the events defined in the input alphabet are included, in the same way all the output symbols are analyzed. Because this is the last phase, it is wanted to improve the model generated until the previous phase. This becomes possible inserting to the algorithm a word that presents a succession of all the events defined in the input alphabet. Input $= \alpha$, β, η, δ, α, δ, α, θ, δ, α, β, η, δ, α, δ, α, θ, δ, ζ, γ, ε, η, α, θ, δ, α, ζ, γ, ε, η, α, θ, δ; Output: B, B, C, C, C, C, C, B, B, B, B, C, C, C, C, C, B, B, B, A, A, B, C, C, B, B, B, A, A, B, C, C, B, B.

It is possible to observe that the final FSM adds a new state (q11) (see Fig. 9). It is feasible to continue designing experiments, but when concluding this phase we have obtained a model that presents a quite approximate behavior to the real system (it accepts all the sequences presented until now with a success rate of 100%). This FSM has less states and transitions than the FSM of the Fig. 3, but can follow the behavior of the real system. To finish, we only need to validate the identified machine.

The validation sequence for the Final machine is: Sequence of Input $= \beta$, η, δ, α, θ, β, η, α, θ, δ, α, ζ, β, ε, α, ζ, γ, ε, η, δ; Sequence of Output $=$ B, C, C, C, B, B, C, C, B, B, B, A, A, B, B, A, A, B, C, C. The sequence of validation allows concluding that our model has a similar behavior to the system "in study".

To conclude, it is possible to have introduced the complete sequence and to wait certain number of generations to that the algorithm give a FSM that recognizes the entirety word. However, the machines generated by the identification process can not be the simplest. For that reason, in some cases when we have a very complex system and need an elaborated DEDS, it is advisable to modulate the procedure, in the way explained here. The sequences of events presented in each phase are the new patterns inserted in the phase, in a way that the EP can learn these one whitout forget the previous one.

5. Conclusions

It has been analyzed along this investigation the incidence of the EP for the resolution of DES problems. In particular, the identification of this type of dynamic was studied, obtaining satisfactory results and demonstrating this way the efficiency of the EP in the search of models that adapt to the presented characteristics. A

methodology of System Identification is exposed that shows a combination of simplicity, facility, rapidity and effectiveness, which allows it to compete with any other way to analyze DEDS.

It is evident the efficient form like the EP generates possible candidates to identify the system in study; it is made optimizing dynamically the behavior that each machine presents through evolutive changes.

Although the technique of EP for the identification of DEDS is not exempt of inconveniences, these complications are not bigger than those that are presented when identifying the systems for conventional methods, which require a coarse domain of the theory of sequential machines, among other things.

References

[1] J. Aguilar and F. Rivas, eds, *Introducción a las Técnicas de Computación Inteligente*, Mérida, Venezuela, MERITEC, 2001.

[2] J. Aguilar, La Programación Evolutiva en la Identificación de Sistemas Dinámicos a Eventos Discretos, *Revista IEEE Latinoamerica Transactions* **5**(5) (2007), 301–310.

[3] I. Aleksander and F. Keith Hanna, *Automata Theory: An Engineering Approach*, *Computer Systems Engineering Series.* Crane, Russak & Company, New York, 1975.

[4] E. Altamiranda, R. Calderon and E. Colina, *An Evolutionary Algorithm for Linear Systems Identification Source*, Proceedings 6th WSEAS International Conference on Signal Processing, Robotics and Automation, Greece, pp. 225–229, 2007

[5] P. Angeline and D. Fogel, n evolutionary program for the identification of dynamical systems, *IEEE Trans on Evolutionary Computation* (1997).

[6] M. Anti and C. Seatzu, "TTT". Proceeding 9th Iintl. Workshop on Discrete Event Systems, WODES 2008.

[7] M. Cerrada, J. Aguilar, A. Titli and E. Colina, *Dynamical Adaptive Fuzzy Systems: An Application on System Identification*, Proceedings of the 15th Triennial World Congress of the International Federation of Automatic Control, pp. 1006–1011, Barcelona, España, 2002

[8] M. Cerrada, J. Aguilar, A. Titli and E. Colina, Identificación de un Sistema No-lineal Variante en el Tiempo usando Sistemas Difusos Adaptativos Dinámicos, Revista Técnica de la Facultad de Ingeniería, *Universidad del Zulia* **25**(3) (2002), 171–180.

[9] M. Cerrada and J. Aguilar, *Genetic Programming-Based Approach for System Identification*, in Advances in Fuzzy Systems and Evolutionary Computation, Artificial Intelligence (A Series of Reference Books and Textbooks), (Ed. N. Mastorakis), World Scientific and Engineering Society Press, pp. 329–334, 2001.

[10] S. Chung, J. Wu and C. Li, System Identification of Discrete Event Systems, *Journal of the Chinese Institute of Engineers* **27**(2) (2004), 203–210.

[11] M. Dotoli, M. Fanti and A. Mangini, Real Time Identification of Discrete Event Systems Using Petri. Nets, *Automatica Volume* **44**(5) (2008), 1209–1219.

[12] A. Estrada, E. López and J. Lesage, A Comparative Analysis of Recent Identification Approaches for Discrete-Event Systems, *Mathematical Problems in Engineering* **21** (2010), 2010.

[13] D. Fogel, Evolutionary Computation, *IEEE Press* (1995).

[14] D. Fogel, Handbook of Evolutionary Computation, Oxford University, 1997.

[15] J. Heitkotter and D. Beasley, *Hitch Hiker's Guide to Evolutionary Computation* (2000).

[16] S. Klei L. Litz and J. Lesage, *Fault Detection of Discrete Event Systems Using and Identification Approach*, Proceedings 16th IFAC world Congress, Tchèque République, 2005.

[17] Z. Kohavi, *Switching and Finite Automata Theory*, McGraw-Hill, New York, 1970.

[18] A. Tornambue, Discrete-Event System Theory: An Introduction, *World Scientific*, Singapore, 1995.