Genetic Algorithms and Discrete Event Systems: An Application

Ricardo R. Gudwin Fernando A. C. Gomide
UNICAMP/FEE/DCA
C.P. 6101
13.081-970 Campinas - SP - Brazil

ABSTRACT: In this paper we propose an approach for discrete event systems control optimization, based on the theory developed by Ramadge and Wonham and on the Limited Lookahead Policy strategy proposed by Chung and Lafortune. By considering a performance index, i.e. a measure of how well a sequence of events meets its objectives, a genetic algorithm is derived to find optimal decisions for this class of systems. After introducing the theoretic background, an application example concerning the supervisory control of elevator systems is also included.

1.0 Introduction

A discrete event system (DES) is a dynamic system that evolves in accordance with the occurrence of physical events. This kind of system requires control and coordination to ensure the orderly flow of events. They exist in many technological applications, such as in manufacturing and communications systems. Many DES models have been proposed in recent years; examples include Markov chains and Markov jump processes, Petri nets, queuing networks, finitely recursive processes, models based on min-max algebra, discrete event simulation and generalized semi-Markov processes, etc.[1].

For our purposes, a particularly interesting approach based on theory of languages and automata has been initially developed by Ramadge and Wonham (1987, 1989) [4,5]. Their model requires an automaton model that provides a complete description of all future behavior of the discrete event process and an automaton model of the desired or legal behavior. These behaviors are specified as formal languages over the alphabet of events, while the automata are finite representations of these languages. In 1992, Chung & Lafortune proposed a supervisory control scheme, called *limited lookahead control policies* (LLP) [2], that instead of calculating off-line the complete control policy for the discrete event process (which could be infeasible sometimes), implements an on-line scheme where after the occurrence of an event, the next control action is determined on the basis of an N-step ahead projection of the behavior of the process. This procedure then repeats after the execution of the successive events. The choice of N can be dictated by the amount of future information available about the process, or by limitations of the controller in terms of processing power or memory available. Both Ramadge & Wonham and Chung & Lafortune provided in their schemes something they called a control optimization, in the sense that the control they implement are minimum restrictive relative to the open-loop behavior. But no considerations about system performance, in the sense that one legal sequence of events could be better than other legal sequence were done.

In this paper, we introduce DES with performance considerations. In this case, many legal sequence of events with different performance evaluation indices must be considered. The model developed by Chung & Lafortune is modified to include performance analysis. A genetic algorithm (GA) is derived for predictions and optimal decision. An application example concerning supervisory group of elevator systems is considered aiming at optimal car dispatch. The application herein described is a simplified version of an actual system being developed by the authors.

In the next section we provide the DES basic background followed by a review of the Chung & Lafortune model. We propose some modifications in their model to consider performance analysis. Next, in section 3 we develop the application example to show the usefulness of the proposed approach for elevator group control. Finally the conclusions and future work are addressed.

2.0 Discrete Event Systems

According to the RW's theory, a DES is modeled as a generator, i.e. a 5-tuple given by $G = (Q, \Sigma, \delta, q_0, Q_m)$, where Q is the set of states q, Σ is the alphabet or set of output events σ, $\delta : \Sigma \times Q \rightarrow Q$ is a partial transition function, q_o is the initial state and $Q_m \subseteq Q$ is a subset of marked states. G is interpreted as a device that starts in q_0 and executes state transitions (i.e. generate a sequence of events) by following its graph. The transition function δ is extended to comprise not only events, but strings of events. Being Σ^* the set of all finite strings s, including the empty string ε, an extended transition function is defined by $\delta : \Sigma^* \times Q \rightarrow Q$ according to $\delta(\varepsilon, q) = q$, $q \in Q$, and $\delta(s\sigma,q) = \delta(\sigma, \delta(s,q))$ whenever $q' = \delta(s,q)$ and $\delta(\sigma,q')$ are both defined. Any subset of Σ^* is a language over Σ. The strings of a language are often called words. The language L generated by G corresponds to the set of strings s for which $\delta(s,q_0)$ is defined. The language marked by G, L_m is the set of strings s for which $\delta(s, q_0) \in Q_m$. The language L is related to the set of all possible finite sequences of events that can occur, while L_m is a subset of these sequences that are marked. These marked sequences will correspond to desired *final* states, i.e., representing completed tasks (or sequences of tasks) carried out by the physical process modeled by G [4].

To control a DES, certain events of the system are considered to be disabled when desired. Then the set of events Σ is partitioned into uncontrollable and controllable events: $\Sigma = \Sigma_u \cup \Sigma_c$. The events in Σ_c can be disabled at any time, while those in Σ_u can never be disabled. A control input for G consists of a subset $\gamma \subseteq \Sigma$ satisfying $\Sigma_u \subseteq \gamma$. If $\sigma \in \gamma$, then σ is enabled by γ, otherwise σ is disabled by γ. A DES represented by the generator G plus the set of control inputs $\Gamma = \{\gamma\}$ (the set of all possible control inputs γ) is called a controlled DES (CDES). Control of a CDES G consists of switching the control inputs through a sequence of elements γ, γ', γ'', γ''', ... in Γ, in response to the observed string of previously generated events. Such a controller is called a supervisor. Formally, a supervisor is a map $f: L \rightarrow \Gamma$ specifying for each possible string of generated events s the control input f(s) to be applied at that point [5]. We see that, for control purposes, we can consider only strings from a language, being the corresponding generator only a realization of this language. We can work then only in terms of a physical behavior, given by a language L, and a desired behavior, given by a specification language K (which corresponds to the language L_m defined by the marked states of the generator).

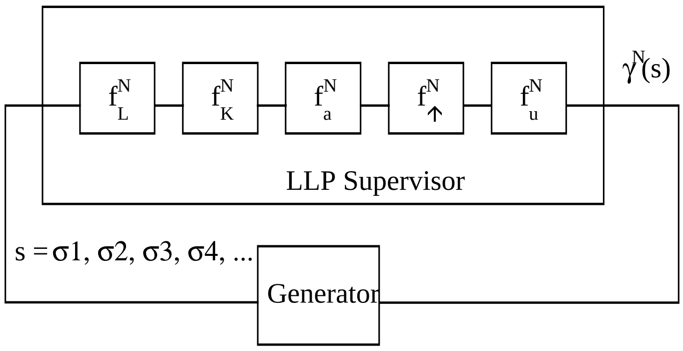


Figure 1 - LLP Supervisor

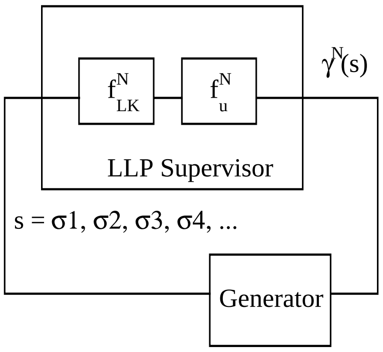


Figure 2 - Simplified LLP Supervisor

Limited lookahead supervisors (LLS) constitute a class of supervisory control systems which can be represented as shown in figure 1. The generator provides LLS with input string s, carrying the sequence of events occurred. The first module f^N_L provides a N-step projection of the possible system behavior, represented by an N-level tree. The branches of this tree is tested for membership in a legal language K, by the module f^N_K. The next module, f^N_a, provides an *attitude*, i.e. the assumption that for a given N-level tree, the next event will lead the system to a legal state or an illegal state. Based on this attitude, the system may work with either a N-level string or (N-1) level strings.[2]. Next module, $f^N_\uparrow$, calculates the supremal controllable sub language of the language, i.e. it will cut off the strings with prefixes that could lead the system to an illegal state due to an uncontrollable event that could happen at the state achieved by this prefix [2,6]. The last module, f_u^N, provides the control input $\gamma^N(s)$, based on the N-level cut out tree.

In this paper we assume module f_u^N as being one which chooses the path in the N-level tree with the best performance index. Only the first event in the optimal string is enabled in the current control cycle. Differently, Chung & Lafortune consider all paths in the N-level tree as equally suitable, enabling the first event of each path. The purpose of the Chung & Lafortune strategy is to generate the minimum restrictive control relative to the supremal controllable sub language of a given language. In our approach, the aim is to find the optimal control in the sense of a given performance index.

In the particular class of applications we are interested, we may assume that all events are controllable and an optimistic attitude [2] is adopted. Hence, the supervisor can be represented as shown in figure 2, where f_{LK}^N is a module which generates the legal sub language, directly. The LLP of figure 2 is implemented by means of a GA. Note that this approach supports adaptive behavior.

3.0 Application Example : Elevator Supervisory Control

In traffic control of elevator systems two different control problems must be solved by a corresponding two level control hierarchy. The lower level task is to command each elevator to move up or down, to stop or start and to open and close the door. The higher level task is to coordinate the movement of a group of elevators through a strategy which aims at improving the system's performance. This problem is solved by means of a group supervisory control system with the aid of a group supervisory control policy. The main requirements of a group control system in serving both car and hall calls should be to provide even service to every floor in a building; to minimize the time spent by passengers waiting for service; to minimize the time spent by passengers to move from one floor to another and to serve as many passengers as possible in a given time [7].

A practical method widely used in group supervisory control systems consists in allocating cars to serve the building hall calls. Usually new calls are allocated to a car as soon as they appear, remaining fixed once made. This method is known as call allocation strategy. An alternative scheme can be implemented, where each car decides to attend a hall call. In this case some hall call remain unallocated until a car chooses to attend it. This method is known as car allocation strategy. Since each car can serve only one call at a time, the second strategy is considered to be more flexible and more suited for our purposes. To model the behavior of such system, a generator as shown in figure 3 was constructed. This generator corresponds to a building with 6 floors.

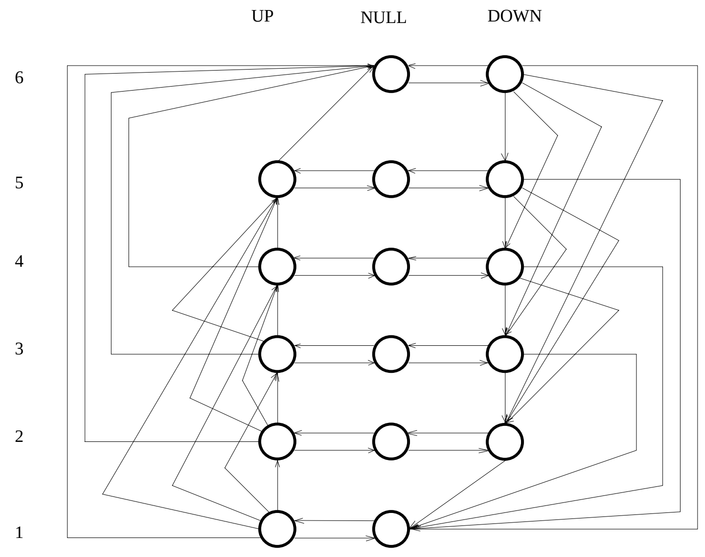


Figure 3 - Generator for an Elevator in a 6 floor Building

The states of the generator are associated with 6 floors of the building, and the directions of movement (up, down or null). That is, each generator state corresponds to a floor and a direction of movement. The events correspond to changing a position (from a floor to any other floor) in a direction, or changing direction, as indicated in figure 3. The change of floor, is the sum of sub-events such as closing door, starting acceleration, achieving cruise velocity, starting disacceleration, stopping the car and opening door. For each event a corresponding execution time is considered and stored in a table. All events are considered to be controllable. Then, for a given initial state, a language L corresponding to the physical behavior is given by all generator's possible transitions, for each elevator. For a given set of hall calls, the main system requirement is to attend each hall call. This is equivalent to sequences of events where at least one car stops at the hall call floor with the hall call direction of movement. The strings corresponding to such sequences of events comprises the legal language K. From the language L we can only select those strings of the sub-language $K \subseteq L$, associated with the control requirement. Moreover, the system must find a sequence of events which minimizes the average waiting time. Then, we associate a performance index (the average waiting time for the hall calls) to each string. Assuming an optimistic attitude and all events controllable, we can use the simplified LLP of figure 2.

Instead of first generating the possible paths and then restricting them to the paths that attend all hall calls, we develop a codification scheme for the GA, which guarantees that all hall calls are attended by at least one car. This corresponds in generating the strings from language K at once, bypassing the generation of L. The codification is as follows. First, we list all hall calls, associating each hall call to a gene in a chromosome. The value of each gene should be a car number, corresponding to the car that will attend the call. This codification guarantees that at least one car will attend a call. This string (the chromosome, called *genetic string*, GS, to avoid confusion with the strings of DES) is translated into DES strings for each car, assuming that following its initial direction of movement, the cars make up and down cycles, while attending the hall calls. For each hall call, the waiting time and the predicted attending time are computed, as well as the average waiting time for all hall calls. The average waiting time is set as the performance index for this GS. A population of GSs is then generated randomly [3].

We first select pairs of GSs, using the Roulette Wheel algorithm [3], applying then the crossover operator. The crossover is such that 50% of the genes, randomly selected, are changed between the parent mates. This type of crossover was chosen because it avoids codification influences in the algorithm's performance, and allows manipulation of the rate of change between mates (in this case 50%). After generating new individuals, they are added to the original population. The new population is ordered by the fitness values, and only the original length population remain alive. For example, if we originally had a population of 50 individuals, and if after mating 50 more individuals were generated (summing 100), then only the best 50 of the 100 will remain for future evolution. After a given number of steps, we pick the GS with best fitness and consider it to be the solution.

This GS is further translated into its associated event strings (one for each elevator), and the first event of each string is enabled in this control cycle.

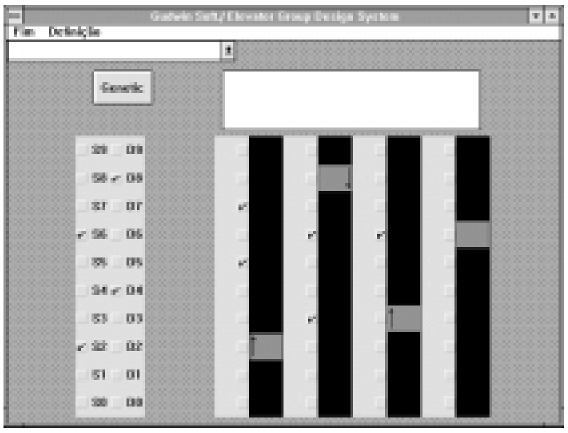
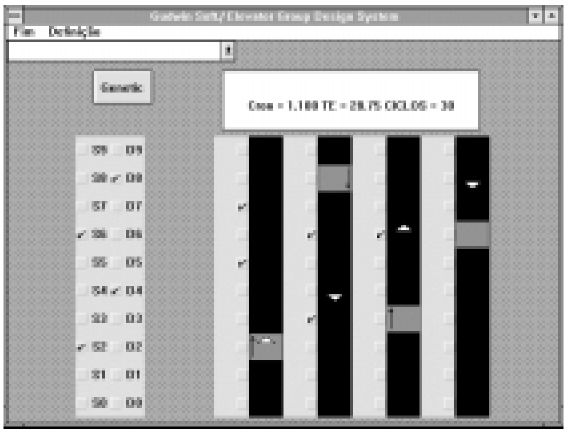


Figure 4 - Program Input



```
Cnon = 1.100 TE = 28.75 CICLOS = 30
```

Figure 5 - Program Output

The GA procedure developed was implemented in C++ language and tested with different data sets (number of floors, cars, hall calls and car calls), including different population length and evolution cycles for each data set. A sample of the results obtained are shown in figures 4 (initial state) and 5 (state after car allocation by the procedure).

4.0 Conclusions

In this paper an approach aiming at optimal control for discrete event systems was introduced. An application in supervisory control of elevator systems was addressed to show the usefulness of the proposed approach. Currently we are studying the real-time performance of the procedure developed aiming at a future experimentation in a real world system.

5.0 References

[1] Cao, Xi-Ren ; Ho, Yu-Chi - "Models of Discrete Event Dynamic Systems" - IEEE Control Systems Magazine, June 1990, pp 69-76.

[2] Chung, Sheng-Luen ; Lafortune, S. - "Limited Lookahead Policies in Supervisory Control of Discrete Event Systems"- IEEE Transactions on Automatic Control, vol. 37, n. 12, December 1992.

[3] Goldberg, D. E. - "Genetic Algorithms in Search, Optimization, and Machine Learning" - Addison-Wesley Publishing Company, Inc. - 1989

[4] Ramadge, P.J. ; Wonham, W.M. - "Supervisory Control of a Class of Discrete Event Processes"- SIAM Journal Control and Optimization, vol.25, n. 1, January 1987.

[5] Ramadge, P.J. ; Wonham, W.M - "The Control of Discrete Event Systems" - Proceedings of the IEEE, vol. 77, n. 1, January 1989.

[6] Wonham, W.M.; Ramadge P.J. - "On the Supremal Controllable Sub language of a Given Language" - SIAM Journal Control and Optimization, vol. 25, n. 3, May 1987.

[7] Barney, G.C.;dos Santos, S.M. - "Elevator Traffic Analysis Design and Control" - IEE Control Engineering Series 2 - Peter Peregrinus Ltd. London UK 2nd. Edition (1985)

Acknowledgments: R. Gudwin acknowledges CNPq - The Brazilian National Research Council for a fellowship; F. Gomide acknowledges CNPq for grant #300729/86-3