Optimal Supervisory Control of Aircraft Propulsion: A Discrete Event Approach[†]

Murat Yasar[‡], Devendra Tolani, Asok Ray
Mechanical Engineering Department
The Pennsylvania State University
University Park, PA 16802

[‡]**Corresponding Author:** Email: myasar@psu.edu; Tel: (814) 865-8427

Keywords: Optimal Control, Discrete Event Systems, Supervisory Control, Gas Turbine Engine, Propulsion Systems

Abstract[†]

This paper presents an application of Discrete Event Supervisory (DES) control theory for intelligent decision and control of a twin-engine aircraft propulsion system. A dual layer hierarchical DES controller is designed to supervise and coordinate the operation of two engines of the propulsion system. The two engines are individually controlled to achieve enhanced performance and reliability, necessary for fulfilling the mission objectives. Each engine is operated under a continuously varying control system that maintains the specified performance and a local discrete-event supervisor for condition monitoring and life extending control. A global upper level DES controller is optimally designed for load balancing and overall health and mission management of the aircraft propulsion system.

1 Introduction

Discrete-event dynamical behavior of physical plants is often modeled as regular languages that can be realized by finite-state automata. This paper focuses on development of intelligent decision and control algorithms based on the theory of Discrete Event Supervisory (DES) control [1] [2] for a twin-engine aircraft propulsion system.

The DES control system is designed to be hierarchically structured in the following sense: Continuously varying control of an engine interacts with its own local DES controller for detailed health monitoring and intelligent control, and the operational information is abstracted and reported to the coordinator for propulsion level DES control that considers the operation of two engines. Furthermore, the supervisory control system at the propulsion level allows interactions with external inputs such as human operator and inputs from other units (e.g., flight control, structural control, energy management, and avionic systems) of the vehicle management system for flexibility of making on-line modifications in the mission objectives. The uniqueness of this DES control approach is that the control policy can be adaptively updated on-line at both engine and propulsion supervisor levels and that the system is tolerant of small component faults.

Although DES control has been developed for quite some time, there are only a few application examples. It is primarily because no quantitative analytical tool has been established to help design and evaluate the DES controllers. This is the first time hierarchical DES control is used on nonlinear complex dynamical systems like an aircraft engine.

The objective of the paper is to apply Discrete Event Supervisory (DES) control theory for intelligent decision and control of a twin-engine aircraft propulsion system. In this paper, it has been demonstrated that: (1) DES control can be used for intelligent decision and control of twin-engine aircraft propulsion systems e.g. the problem of load balancing between engines during various operating conditions. (2) Implementation of DES control can be used to reduce the engine damage and thus to extend the life of aircraft engine; (3) DES control is helpful to improve the overall mission and operational behavior.

The paper is organized in six sections including the present one. In Section 2, theoretical foundations of DES control theory is reviewed. Section 3 elaborates on the topic of simulation setup. Section 4 discusses the design of the engine level DES control and propulsion level DES control design. The simulation results are presented and discussed in Section 5. The paper is summarized and concluded in Section 6.

2 Review of Language Measure and Optimal Control

This section reviews the previous work about the optimal control policy based on language measure [3]. It provides the background information necessary to develop a performance index.

Let the dynamical behavior of a physical plant be modeled as a deterministic finite state automaton (DFSA) $G_i \equiv (Q, \Sigma, \delta, q_i, Q_m)$ with $|Q| = n$ and $|\Sigma| = m$.

[†] This work has been supported in part by the United States Army Research Office under Grant No. DAAD19-01-1-0646 and NASA Glenn Research Center under Grant Nos. NAG3-2448 and NNC04GA49G.

Definition 1: The characteristic function $\chi : Q \rightarrow [-1, 1]$ assigns a signed real weight to states and is defined as:

$$\chi_j \equiv \chi(q_j) \in \begin{cases} [-1,0) \; if \; q_j \in Q_m^- \\ \{0\} \quad if \; q_j \notin Q_m \\ (0,1] \; if \; q_j \in Q_m^+ \end{cases} \text{ independent of } q_i$$

The $(n \times 1)$ characteristic vector is denoted as:

$$\overline{\chi} \equiv [\chi_1 \; \chi_2 \cdots \chi_n]^T$$

Definition 2: The event cost is defined as the relative frequency of occurrence of an event given the DFSA state at which the event is generated, such that

- $\tilde{\pi}[\sigma_k | q_j] = 0$ if $\delta(q_j, \sigma_k)$ is undefined; $\tilde{\pi}[\varepsilon | q_j] = 1$;
- $\tilde{\pi}[\sigma_k | q_j] \equiv \tilde{\pi}_{jk} \in [0,1)$; $\sum_k \tilde{\pi}_{jk} < 1$;

The $(n \times m)$ event cost matrix is denoted as: $\tilde{\Pi} \equiv [\tilde{\pi}_{ij}]$.

Definition 3: The state transition cost of the DFSA is a function $\pi : Q \times Q \rightarrow [0,1)$ defined as the relative frequency of transition from state j to state k such that $\pi(q_k | q_j) = \sum_{\sigma \in \Sigma : \delta(q_j, \sigma) = q_k} \tilde{\pi}(\sigma | q_j) \equiv \pi_{jk}$ and $\pi_{jk} = 0$ if $\{\sigma \in \Sigma : \delta(q_j, \sigma)\} = \varnothing$. The $n \times n$ state transition cost matrix, denoted as Π, is defined as:

$$\Pi = \begin{bmatrix} \pi_{11} & \pi_{12} & \cdots & \pi_{1n} \\ \pi_{21} & \pi_{22} & \cdots & \pi_{2n} \\ \vdots & & \ddots & \vdots \\ \pi_{n1} & \pi_{n2} & \cdots & \pi_{nn} \end{bmatrix}$$

It has been shown in [3] that the measure of the language $L(G_i)$, where $G_i = (Q, \Sigma, \delta, q_i, Q_m)$ can be expressed as: $\mu_i = \sum_j \pi_{ij} \mu_j + \chi_i$. Equivalently, in vector notation: $\overline{\mu} = \Pi \overline{\mu} + \overline{\chi}$. Since Π is a contraction operator, the measure vector $\overline{\mu}$ is uniquely determined as: $\overline{\mu} = [I - \Pi]^{-1} \overline{\chi}$.

Fu et al. [4] have introduced the concept of unconstrained optimal control of regular languages based on the specified measure. In each iteration, the optimal control algorithm attempts to disable all controllable events leading to "bad marked states" and enable all controllable events leading to "good marked states."

Let G be the DFSA plant model and let the state transition cost matrix of the open loop plant be: $\Pi^{plant} \in \Re^{n \times n}$ and the characteristic vector be: $\overline{\chi} \in [-1,1]^{n \times 1}$. Starting with $k = 0$ and $\Pi^0 \equiv \Pi^{plant}$, the control policy is constructed by the following two-step procedure:

Step 1: For every state q_j for which $\mu_j^0 < 0$, disable controllable events leading to q_j . Now, $\Pi^1 = \Pi^0 - \Delta^0$, where $\Delta^0 \geq 0$ is composed of event costs corresponding to all controllable events that have been disabled at $k = 0$.

Step 2: Starting with $k = 1$, if $\mu_j^0 \geq 0$, re-enable all controllable events leading to q_j, which were disabled in Step 1. The cost matrix is updated as: $\Pi^{k+1} = \Pi^k + \Delta^k$ for $k \geq 1$, where $\Delta^k \geq 0$ is composed of event costs corresponding to all currently re-enabled controllable events. The iteration is terminated if no controllable event leading to q_j remains disabled for which $\mu_j^0 \geq 0$. At this stage, the optimal performance is $\overline{\mu}^* \equiv [I - \Pi^*]^{-1} \overline{\chi}$.

3 Implementation of the DES Control Concept

This section presents the implementation of DES control on a real-time simulation test bed of a twin-engine aircraft propulsion system where the engine model is similar in complexity and details to that reported by Diao and Passino [5]. The objective of such application is to validate the DES control techniques with a real world nonlinear complex dynamical system.

A DES controlled propulsion system is being designed and tested on a simulation test bed that consists of three networked computers using the client/server concept. One of the computer acts as the propulsion system coordinator for health monitoring of the engines and accordingly making intelligent decisions such as load balancing. The other two computers run separate copies (which may or may not be different depending on the health of the engine) of an aircraft gas turbine engine simulation model including its (continuously varying) control system and a local discrete-event supervisor. The test bed is capable of simulating different dynamics for individual engines due to non-uniform operating conditions. Each of the engine simulators integrates the event-driven discrete dynamics modeled by finite-state automaton as well as time-driven continuous dynamics modeled by ordinary differential equations through continuous-discrete and discrete-continuous interfaces.

In order to implement the DES control in tandem with existing turbofan engine simulation program which is written in FORTRAN language, a C++ code was written to wrap the FORTRAN simulation code. The C++ wrapper program takes the major inputs / outputs and makes them working in C++ environment.

Turbofan engine simulation is a stand-alone program with its own continuous time gain scheduling controller that is left untouched. With proper inputs, the FORTRAN program simulates the complex operation of the engine from transient to some steady state condition with large order differential and difference equations.

Figure 1 shows the architecture of the engine level plant and DES controller implementation, which has two replicas in two different computers. In Figure 2, the organization of the propulsion level DES controller together with its own event generator is presented. The latter structure is implemented on a third computer which uses the messaging interface to communicate with the other two computers.

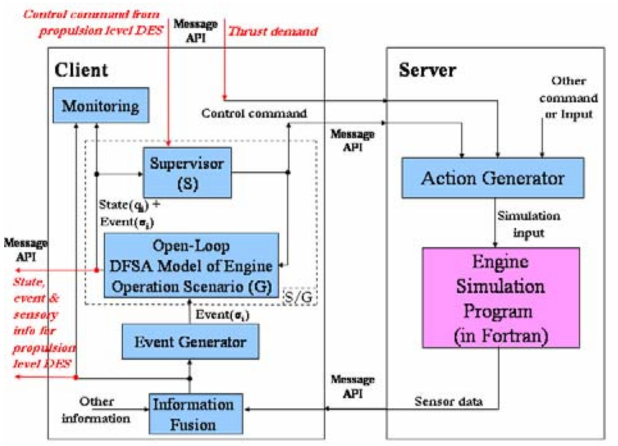

Figure 1. Engine Level Plant/DES Controller

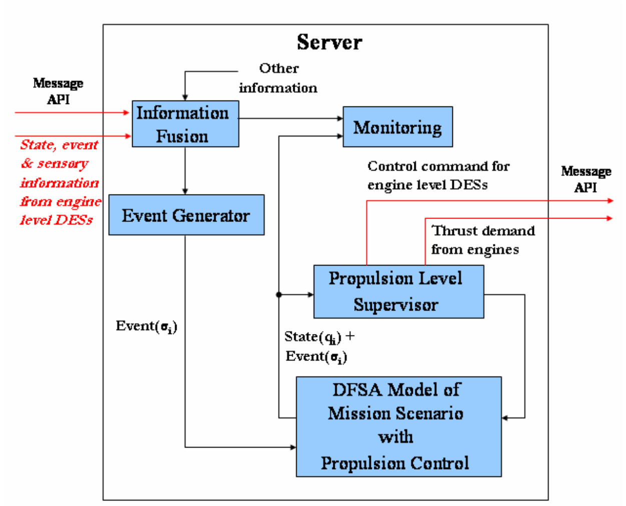
Server
Other information
Message API
Information Fusion
Monitoring
State, event & sensory information from engine level DESs
Control command for engine level DESs
Message API
Event Generator
Thrust demand from engines
Propulsion Level Supervisor
Event(σ_i)
State(q_k) + Event(σ_i)
DFSA Model of Mission Scenario with Propulsion Control

Figure 2. Propulsion Level DES Controller

The DES control system has two important components. One is Event Generator and the other is Action Generator, as seen in Figure 1. Event Generator is the module which receives continuous time sensor data from the plant. The data along with other information like estimated state and external inputs are used by this module to generate events, which in turn are inputs to open-loop DFSA model of engine operation. The open-loop DFSA model is constructed based on the operation scenario. The details of the modeling are discussed in Section 4. The state-based DFSA model serves as state estimator and provides important state and event (both controllable and uncontrollable) information for the discrete-event supervisor to take proper action.

The supervisor represents the control policy applied to the DFSA model of engine operation, and it could be an ad hoc DES controller based on experience or a supervisor designed by the control techniques discussed in Section 2. The DES controller takes estimated state as input and generates control commands as outputs. The control commands are sent through Message API communication routine to Action Generator. The primary task of the action generator is to convert control commands from supervisor into necessary simulation input for the continuously varying plant.

The software architecture of the simulation test bed is flexible to adapt arbitrary DFSA models and controller designs and to fit other complex dynamic systems.

4 DES Control System Modeling

The DES control scheme has a two layer hierarchical structure where the open-loop discrete event dynamics are modeled as two DFSA, based on the postulated engine operation scenario. The modeling may vary for different mission scenarios. The propulsion level DFSA plant model assumes that an aircraft equipped with a twin turbofan jet engine is carrying out a routine surveillance mission. Abortion of the mission is allowed at certain states according to the mission scenario. Each engine of the aircraft is equipped with a continuous time controller which is supervised by a local DES controller. The primary objective of the engine level DES controller is to strike the right balance between the conflicting demands of higher performance from upper level supervisor and limiting the damage to the engine. The propulsion level DES controller redistributes the load depending on the health of the engine and thrust demand of the pilot.

4.1 Engine Level DES Control

The supervisor is derived from the engine DFSA model. The engine can operate in two regimes, one is high performance regime where the damage rate is also high and other is low performance regime where the damage rate is low.

In the high performance regime the engine has a tendency of going to the state where engine variables like combustor temperature have been observed to have oscillatory behavior. These oscillations are extremely harmful in terms of engine health. Engine level controller chooses the regime of operation depending on two factors: (1) Thrust requirement of upper level (2) Health of the engine, which is determined by the damage accumulated over the period of operation. Damage accumulation has two components. It is formulated as a function of high-pressure turbine gas inlet temperature and shaft speed; and in addition, at random time intervals spikes (sudden jumps) are introduced to simulate real world damage impacts in an engine.

4.2 Propulsion Level DES Control

One of the main tasks of the upper level DES controller is to redistribute the load between two engines depending on the current health condition of each engine and the thrust demand of the pilot. Other important motive of the propulsion level DES controller is to obtain better mission performance. Thus, mission scenario elements, like "successful mission" and "mission abort" are elaborately used in the modeling stage. In this hierarchical control scheme propulsion level supervisor should be able to carry out the operational properties of the engine level supervisors. Thus, operating regimes of individual engines are included in the propulsion level model as Cartesian product. However, model order reduction is needed to avoid the expansion of number of states in the upper level due to unrealizable states.

5 Simulation Experiments: Results and Discussion

The experimentation setup for the DES simulator is designed on the engine simulation test bed to validate the DES control concept. Upon successful implementation of the software modules several sets of experiments were conducted. The first set of experiments were performed on the engine level DES controller to validate that, implementation of DES control reduces the engine damage and thus extends engine life. Both unsupervised and supervised plants were excited by the same predetermined fixed input shown in Figure 3. To make a comparison between the supervised and the unsupervised cases several outputs of the engine simulation were observed. Figure 4 and 5 shows the simulation outputs for the unsupervised and supervised cases respectively.

The comparison of two figures indicates that the DES controller applied in the engine level quenches the high frequency oscillations which are observed in the unsupervised case. High frequency oscillations are modeled as the primary cause of damage to the turbine blades and adversely affect the overall health of the engine.

The propulsion level DES controller has two main tasks. One is the intelligent decision making and control of twin-engine aircraft propulsion systems and the second is to improve the overall mission and operational behavior. The issue of load balancing becomes even more important when the health conditions of two engines are significantly different (one can be in "bad condition" and the other in "good condition"). If the situation comes to this point, then the aim of the DES controller is judicious redistribution of the load between two engines such that the "bad engine" carries lower load than the "good" one, subject to the condition that the total thrust output of the engines remains same.

Figure 6 and 7 shows the simulation outputs of two engines: In the start of the simulation, both engines are in "good" health condition, as the mission progresses one engine deteriorates and starts to run at "bad" health condition, and after some time the other engine also deteriorates. Thus, the load distribution of the engines varies in three regions. In the beginning, the load is distributed equally (region 1), then the "good engine" (Engine 2) takes more responsibility (region 2), and at the end both engines are again loaded equally (region 3).

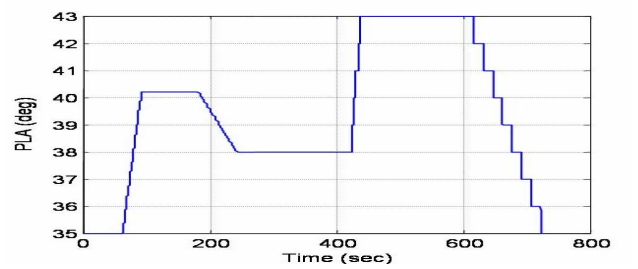

Figure 3. Power Lever Angle input

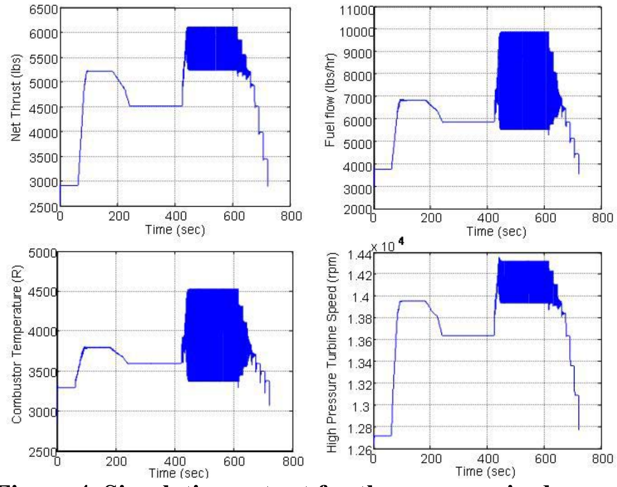

Figure 4. Simulation output for the unsupervised case

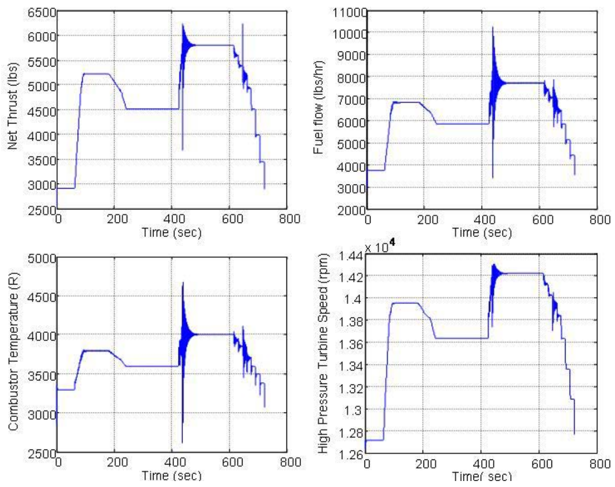

Figure 5. Simulation output for the supervised case

To observe the effect of the propulsion level DES controller on the overall mission behavior, the concept of language measure is used. Language measure gives a theoretical performance measure for the controllers, given the state weights and the state transition probabilities. To form the state transition probability matrix Π, 50 experimental runs were conducted for both supervised and unsupervised cases.

For the propulsion level DES controller, the weights of the states are selected according to each state's importance (contribution) to the mission management as $\overline{\chi} = [0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad -1.0 \quad -0.2 \quad 0.3 \quad 0 \quad 0]$.

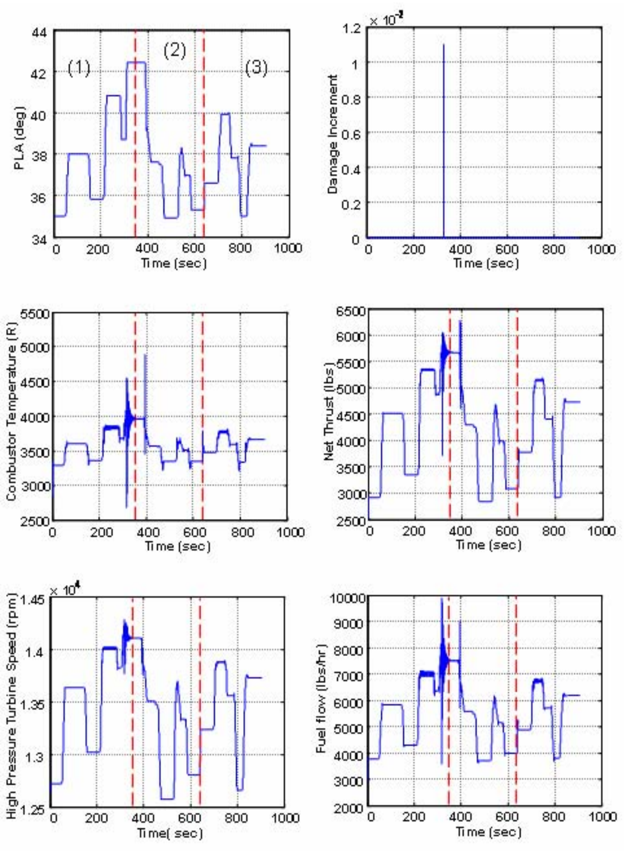
Figure 6. Effect of DES controller on Engine 1

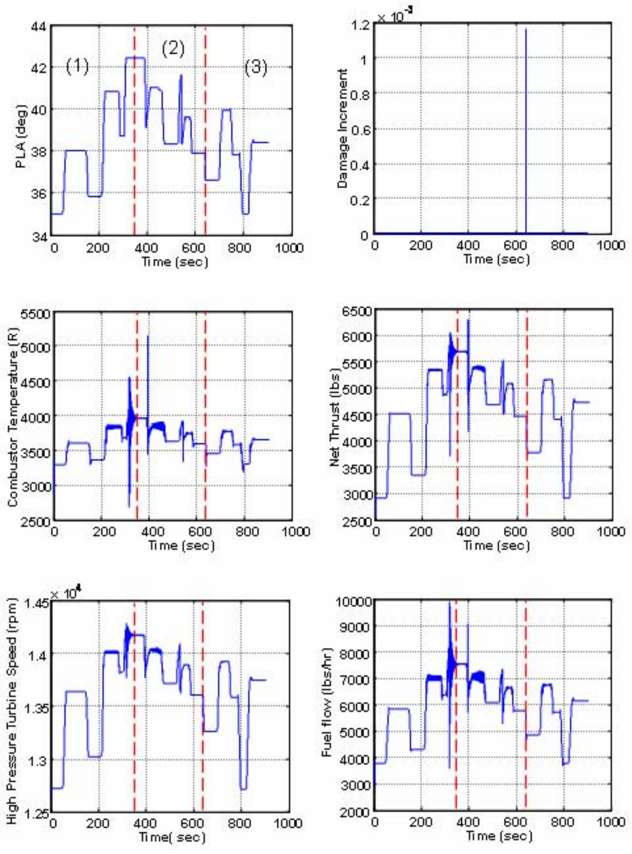
Figure 7. Effect of DES controller on Engine 2

The states which have weights other than zero are State Q_8: *Both engines failed*, State Q_9: *Mission abort*, State Q_{10}: *Mission successful*. They have relative weights of -1.0, -0.2 and 0.3 respectively. The language measures (the theoretical performance) of the propulsion level DES control for the unsupervised and supervised cases are computed to be $\mu_{unsupervised} = 0.0646$ and $\mu_{supervised} = 0.1374$ respectively. It can be concluded that the DES controller has a positive effect on the mission behavior of the overall system.

5.1 Parameter Identification

Similar to continuously varying dynamical systems (CVDS), techniques of system identification are used to identify the language measure parameters of the DFSA plant model, i.e. the entries $\tilde{\pi}_{ij}$ of the event cost matrix $\tilde{\Pi}$. As the number of experiments increases, the identified event costs tend to converge in the Cauchy sense. For stationary operation of the engine, since conditional probabilities of the events can be assumed to be time-invariant, the identified event costs and their uncertainty bounds can be determined. As a typical case, Figure 8 presents identification of event costs at State Q_5 (Both engines in low performance operation). During 50 experiments, the states visited and the events occurred were monitored and plotted.

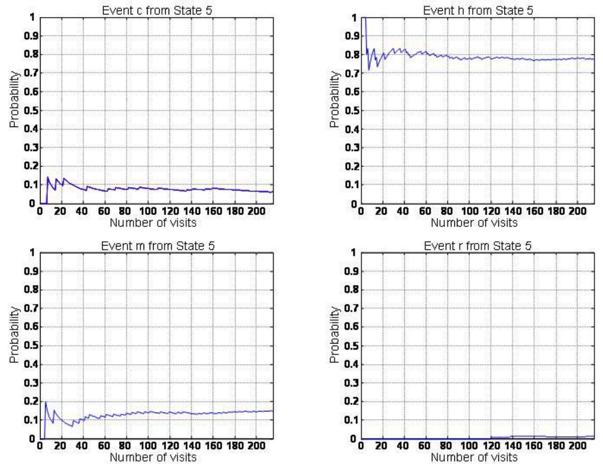

Figure 8. Convergence of event cost identification

After identifying the event cost matrix $\tilde{\Pi}^0$ of the open loop plant *G*, the next step is to obtain the state transition cost matrix Π^0 using Definition 3.

5.2 Optimal DES Controller Synthesis

Given the state transition cost matrix Π^0 of the open loop plant automaton and the state characteristic vector $\bar{\chi}$, the optimal DES controller can be synthesized. The characteristic values are assigned based on the designer's perception of the importance of terminating on specific marked states. As an example, the bad marked state, *Both engines failed*, assigned the characteristic value of -1.0 because the aircraft will most likely be destroyed if the DFSA terminates on this state. On the other hand, the good marked state, *Mission successful*, is assigned the characteristic value of +0.3 based on its relative importance to the loss of the aircraft. Another bad marked state, *Decision to abort mission*, has a negative characteristic value which signifies the importance of this state relative to a successful mission and a possible loss of the aircraft.

Table 1 lists the iterations of optimal control synthesis for the propulsion level supervisory control. The performance measure of the unsupervised plant is negative at the states Q_6, Q_7, Q_8, Q_9, Q_{11}, and Q_{12} as indicated by bold script. All controllable events leading to these states are disabled and the resulting performance measure at Iteration 1 shows a sign change at states Q_7, Q_{11}, and Q_{12} as indicated by italics. All controllable events leading to these states are now re-enabled for further increase in performance at Iteration 2. However, there is no sign change in the performance vector between Iteration 1 and Iteration 2, which immediately shows that the algorithm converged to the optimal solution after this iteration.

Table 1. Optimal Controller Synthesis Iterations

	Unsupervised plant	**Iteration 1**	**Iteration 2**
State Q_1	0.0646	0.2452	0.2452
State Q_2	0.0653	0.2477	0.2477
State Q_3	0.0660	0.2502	0.2502
State Q_4	0.0795	0.2617	0.2617
State Q_5	0.0947	0.2785	0.2785
State Q_6	**-0.3233**	-0.1238	-0.1238
State Q_7	**-0.2287**	***0.0000***	0.0000
State Q_8	**-1.0000**	-1.0000	-1.0000
State Q_9	**-0.2310**	-0.0353	-0.0353
State Q_{10}	0.3640	0.5428	0.5428
State Q_{11}	**-0.0961**	***0.1132***	0.1132
State Q_{12}	**-0.1212**	***0.0919***	0.0919

The performance of the optimal controller was compared with that of unsupervised plant and supervised plant (designed using the conventional procedure, thus suboptimal). Theoretical performance of the supervisors can be associated with the language measure of each supervisor. The language measure of the unsupervised plant and that of the DES controllers are listed in Table 2. Note that the optimal controller not only yields the best mission performance compared to other cases, but also functions as an intelligent coordinator at the propulsion level for the load balancing between two engines.

Experimental outcomes, shown in Table 3, can also be used to evaluate the DES controller performance directly by multiplying each state visit with the relative weight of the state. The performance of the null and the two supervisors are compared based on observations of mission execution on the simulation test bed. For each controller, 50 missions were simulated, and the mission outcomes were recorded. The experimental performances of the controllers are presented in Table 2.

Table 2. Performance under different supervisors

	Unsupervised	**Supervised**	**Optimal**
Language Measure (μ)	0.0646	0.1374	0.2452
Experimental Performance	6.2	9.7	12.4

Table 3. Simulated performance of controllers

State	**Description of state**	**Relative weight of state**	**Number of state visits**		
			Unsupervised	**Supervised**	**Optimal**
Q_8	Plane destroyed	-1	1	1	2
Q_9	Mission Abort	-0.2	15	8	0
Q_{10}	Mission Successful	0.3	34	41	48

It is seen that the theoretical performance (μ) values of the supervisors are in qualitative agreement with the experimental results, presented in Table 2. Optimal DES controller, synthesized using the algorithm described in Section 2, has the highest language measure value, among all controllers. The results of the simulation experiments concur with the theoretical measure of the controllers.

6 Summary and Conclusions

This paper presents a quantitative method for synthesis of optimal discrete-event supervisory (DES) control systems for aircraft propulsion. The DES control law has been validated on a networked simulation test bed. The plant dynamics in the simulation test bed is built upon the model of a generic turbofan gas turbine engine. The software architecture of the simulation test bed is flexible to adapt arbitrary DFSA models and controller designs and to fit other complex systems. The supervisory control laws are quantitatively analyzed using a language measure.

References

[1] P.J. Ramadge, and W. M. Wonham, "Supervisory control of a class of discrete event processes", *SIAM J. Control and Optimization,* Vol. 25, No. 1, January 1987, pp. 206-230.

[2] C.G. Cassandras, and S. Lafortune, *Introduction to Discrete Event Systems*, Kluwer Academic, 1999.

[3] A. Ray, V.V. Phoha, and S. Phoha, *Quantitative Measure for Discrete Event Supervisory Control*, Springer, 2005

[4] J. Fu, A. Ray, and C. Lagoa, "Unconstrained Optimal Control of Regular Languages", *Automatica*, Vol. 40, No. 4, April 2004, pp. 639-646.

[5] Y. Diao, and K.M. Passino, "Stable Fault-Tolerant Adaptive Fuzzy/Neural Control for a Turbine Engine," *IEEE Transactions on Control Systems Technology*, Vol.9, No. 3, May 2001, pp. 494-509.