SIAM J. CONTROL OPTIM.
Vol. 39, No. 2, pp. 512–532

ON THE SYNTHESIS OF OPTIMAL SCHEDULERS IN DISCRETE EVENT CONTROL PROBLEMS WITH MULTIPLE GOALS*

HERVÉ MARCHAND†, OLIVIER BOIVINEAU‡, AND STÉPHANE LAFORTUNE‡

Abstract. This paper deals with a new type of optimal control for discrete event systems. Our control problem extends the theory of [R. Sengupta and S. Lafortune, *SIAM J. Control Optim.*, 36 (1998), pp. 488–541] that is characterized by the presence of uncontrollable events, the notion of occurrence and control costs for events, and a worst-case objective function. A significant difference with [R. Sengupta and S. Lafortune, *SIAM J. Control Optim.*, 36 (1998), pp. 488–541] is that our aim is to make the system evolve through a set of multiple goals, one by one, with no order necessarily prespecified, whereas the previous theory only deals with a single goal. Our solution approach is divided into two steps. In the first step, we use the optimal control theory in [R. Sengupta and S. Lafortune, *SIAM J. Control Optim.*, 36 (1998), pp. 488–541] to synthesize individual controllers for each goal. In the second step, we develop the solution of another optimal control problem, namely, how to modify if necessary and piece together, or schedule, all of the controllers built in the first step in order to visit each of the goals with the least total cost. We solve this problem by defining the notion of a scheduler and then by mapping the problem of finding an optimal scheduler to an instance of the well-known traveling salesman problem (TSP) [E. L. Lawler, J. K. Lenstra, A. H. G. Rinooy Kan, and D. B. Shmoys, *The Traveling Salesman Problem*, John Wiley, 1985]. We finally suggest various strategies to reduce the complexity of the TSP resolution while still preserving global optimality.

Key words. discrete event systems, optimal control, scheduler, traveling salesman problem

AMS subject classifications. 93A99, 49-XX, 90C27, 90B35

PII. S0363012998341964

1. Introduction and motivation.

We are interested in a new class of optimal control problems for discrete event systems (DES). We adopt the formalism of supervisory control theory [16] and model the system as the regular language generated by a finite state machine (FSM). Our control problem follows the theory in [19, 20, 21] and is characterized by the presence of uncontrollable events, the notion of occurrence and control costs for events, and a worst-case objective function. A significant difference with the work in [21] and with the other works dealing with optimal control of DES [6, 11, 14, 24] is that we wish to make the system evolve through a set of marked states (or multiple goals) one by one, with no order necessarily specified a priori; in contrast, the previous theories only deal with a single marked state.

Our problem formulation is motivated by several application domains such as test objective generation in verification and diagnostics, planning in environments with uncertain results of actions, and routing in communication networks.

- In test objective generation, a given system has been designed to meet some specific requirements. However, it may happen that some of these require-

*Received by the editors July 10, 1998; accepted for publication (in revised form) May 11, 2000; published electronically September 15, 2000. This research was supported in part by INRIA and by the Department of Defense Research and Engineering (DDR&E) Multidisciplinary University Research Initiative (MURI) on Low Energy Electronics Design for Mobile Platforms and managed by the Army Research Office (ARO) under grant DAAH04-96-1-0377.

http://www.siam.org/journals/sicon/39-2/34196.html

†IRISA / INRIA - Rennes, F-35042 RENNES, France (hmarchan@irisa.fr). The work of this author was carried out at the Department of Electrical Engineering and Computer Science, University of Michigan.

‡Department of Electrical Engineering and Computer Science, University of Michigan, 1301 Beal Avenue, Ann Arbor, MI 48109-2122 (oboivine@eecs.umich.edu, stephane@eecs.umich.edu).

ments have been overlooked or neglected. Failures can occur as a consequence of negligence. Test objective generation is a way of (ideally exhaustively) checking for inconsistencies in the behavior of the system [1, 3, 17]. The marked states (the states of interest) would be some particular states in which the behavior of the system to be tested is suspected to be flawed. The method that we develop generates a behavior for the system that allows it to reach all these states in an optimal way, with respect to the given occurrence and control cost functions for the events. Each time a state of interest is reached, a behavioral test can be performed on this particular state in order to check if it meets the requirements and conforms to the designed or expected behavior.

- In artificial intelligence (AI), the behavior of an agent is often sought to be optimized with respect to an optimality criterion [5]. Moreover, dealing with multiple goals is an active area of research in AI [13]. The model and the methods that we develop in this work can easily be applied to an agent evolving in an environment where the results of its actions are not always the ones expected. Under certain restrictions, there is a mapping between partial controllability in DES and the notion of a nondeterministic environment[1] in AI [18]. The notion of an optimal scheduler that we define and construct can be used to do planning with multiple goals.
- Broadcasting and multicasting in a communication network is an instance of a multiagent system. Here, the marked states would represent the nodes of the network to which we would want the information to be sent. The uncontrollability of certain events would be interpreted as the uncertainty regarding the actual route that the information would take, since the entire route is not up to the decision of the single sending agent. The solution that we generate can be used to determine the number of duplicated messages that must be sent in parallel through the network in order for all the desired recipients to receive the piece of information.

Our solution approach consists of two steps. The starting point is an FSM which represents the desired behavior of a given system. From this FSM, we can generate a controller that verifies any property that we would wish to associate to it, from the set of acceptable controllers. The desirable property is often taken to minimize a quantitative performance measure. In our case, we generate a controller which verifies a range of properties. This is what has been called the DP-optimality property of an FSM [21]. DP-optimality stands for dynamic programming optimality. This comes from the fact that we use back-propagation from the goal state to generate the controller, based on event cost functions. The controller is represented as an FSM also. The theory of DP-optimal controllers has been developed in the restricted case of one unique marked state [19, 20, 21]. We use the theory in [21] to synthesize a set of optimal controllers corresponding to the different marked states, each treated individually. This yields a set of FSMs that are generated independently from each other. These controllers are synthesized in a manner that gives them an optimal substructure, consistent with the notion of DP-optimality of [21]. The objective function has a worst-case form. The total worst-case computational complexity of the first step is

[1]The notion of a nondeterministic environment in AI is different from the notion of a nondeterministic FSM in control of DES. In AI, a nondeterministic environment is one where the actions undertaken by the agent might not lead to the expected arrival state of the world, whereas in control of DES, a nondeterministic FSM is one in which there are identically labeled transitions that lead from one state to different states.

cubic in the number of states in the systems. At this point, the notion of a DP-optimal controller is replaced by the notion of a stepwise DP-optimal scheduler. By scheduler, we mean a sequence of behaviors that are modeled by FSMs. We develop the solution of a "higher-level" optimal control problem, where we use all the controllers built in the first step in order to visit each of the marked states with least total cost; we call this problem that of finding a "stepwise DP-optimal scheduler." We solve this problem by defining the notion of a scheduler and then by mapping the problem of finding a stepwise DP-optimal scheduler to an instance of the well-known traveling salesman problem (TSP) [8]. We finally suggest different strategies to reduce the computational complexity of this step while still preserving global optimality by taking advantage of some particular properties of the structure of stepwise DP-optimal schedulers.

One of the differences between DP-optimality and stepwise DP-optimality resides in the controller having an FSM structure, whereas the scheduler is a concatenation of FSMs. All the states appear only once in a controller, whereas states can appear several times in a scheduler, but under different circumstances, i.e., in different submachines. Also, another difference between DP-optimal controllers and a stepwise DP-optimal scheduler for an FSM is the existence of a unique maximal DP-optimal controller which contains all the other DP-optimal controllers as submachines, whereas there is no notion of a unique maximal stepwise DP-optimal scheduler.

This paper is organized as follows. In section 2, the necessary notations are introduced. In section 3, we recall the basic definitions and properties of the optimal control theory of DES of [19, 20, 21]. More precisely, we review the notion of a DP-optimal submachine of an FSM G. This definition is used as a springboard to section 4, where we introduce the enlarged problem in the case of multiple marked states. In section 5, we define the notion of an optimal scheduler; such a scheduler ensures that the system will visit each state in a given set of states at least once while minimizing a given cost function over the trajectories of the system. We then suggest possible simplifications that can be made to reduce the overall complexity of the computation of a stepwise DP-optimal scheduler. Section 6 illustrates this new notion of optimality with an example. Section 7 presents some possible applications of the theory that is developed throughout this paper. A conclusion and discussion on future works are presented in section 8.

2. Preliminaries. In this section, the main concepts and notations are defined (more definitions will be made when necessary in the following sections). The system to be controlled is modeled as an FSM defined by a 5-tuple $G = \langle \Sigma, Q, q_0, Q_m, \delta \rangle$, where Σ is the set of events, Q is the (finite) set of states, q_0 is the initial state, Q_m is the set of marked states, and δ is the partial transition function defined over $\Sigma^* \times Q \to Q$. The notation $\delta_G(\sigma, q)!$ means that $\delta_G(\sigma, q)$ is defined, i.e., there is a transition labeled by event σ out of state q in machine G. Likewise, $\delta_G(s, q)$ denotes the state reached by taking the sequence of events defined by trace s from state q in machine G. The behavior of the system is described by the prefix-closed language $\mathcal{L}(G)$ [2], generated by G. $\mathcal{L}(G)$ is a subset of Σ^*, where Σ^* denotes the Kleene closure of the set Σ [4]. Similarly, the language $\mathcal{L}_m(G)$ corresponds to the marked behavior of the FSM G, i.e., the set of trajectories of the system ending in one of the marked states of G.

Some of the events in Σ are uncontrollable, i.e., their occurrence cannot be prevented by a controller, while the others are controllable. In this regard, Σ is partitioned as $\Sigma = \Sigma_c \cup \Sigma_{uc}$, where Σ_c represents the set of controllable events and Σ_{uc} represents the set of uncontrollable events. In what follows, we will only be interested

in *trim* FSMs (i.e., FSMs whose states are all accessible from q_0 and coaccessible to Q_m). For explicit mathematical definitions, the reader may refer to [2]. We say that an FSM $A = \langle \Sigma, Q_A, q_{0A}, Q_{mA}, \delta_A \rangle$ is a submachine of G if $\Sigma_A \subseteq \Sigma$, $Q_A \subseteq Q$, $Q_{m_A} \subseteq Q_m$, and $\forall \sigma \in \Sigma_A, q \in Q_A$, $\delta_A(\sigma, q)! \Rightarrow (\delta_A(\sigma, q) = \delta(\sigma, q))$. The statement $A \subseteq G$ denotes that A is a submachine of G. We also say that A is a submachine of G at q whenever $q_{0A} = q \in Q$ and $A \subseteq G$. For any $q \in Q$, we will use $\mathcal{M}(G, q, Q_m) = \{A \subseteq G : A$ is trim with respect to Q_{m_A} and $q_{0_A} = q\}$ to represent the set of trim submachines of G at q with respect to Q_m. This set has a maximal element in the sense that this maximal element contains all other elements as submachines. It is denoted as $M(G, q, Q_m)$. For convenience, we write $\mathcal{M}(G, q)$ and $M(G, q)$ when there is only one marked state, i.e., when $Q_m = \{q_m\}$.

As stated in [21], to take into account the numerical aspect of the optimal control problem, costs are associated with each event of Σ. To this effect, we introduce an occurrence cost function $c_e : \Sigma \to \mathbb{R}^+ \cup \{0\}$ and a control cost function $c_c : \Sigma \to \mathbb{R}^+ \cup \{0, \infty\}$. Occurrence cost functions are used to model the cost incurred in executing an event (energy, time, etc.). Control cost functions are used to represent the fact that disabling a transition possibly incurs a cost. The control cost function is infinity for events of Σ_{uc}. These cost functions are then used to introduce a cost on the trajectories of a submachine A of G. To this effect, we first define a projection p_j that, when applied to a trace of events $s = \sigma_1^s \sigma_2^s \dots \sigma_{\|s\|}^s$, gives the subtrace of s of length j starting from σ_1^s ($p_j(s) = \sigma_1^s \sigma_2^s \dots \sigma_j^s$ if $j \leq \|s\|$, and is undefined otherwise). We also introduce $\Sigma_d^G(A, q)$ as the set of disabled events at state q for the system to remain in submachine A of G.

DEFINITION 2.1. *Let A be a submachine of G, and let $\mathcal{L}_m(A)$ be the marked language of A. Then the following are defined.*

- *For any state $q \in Q_A$ and string $s = \sigma_1^s \sigma_2^s \dots \sigma_{\|s\|}^s$ such that $\delta_A^*(s, q)$ exists, the cost of the string s is given by*

$$c^g(q, A, s) = \sum_{j=1}^{\|s\|} c_e(\sigma_j^s) + \sum_{j=1}^{\|s\|} \sum_{\substack{\sigma \in \Sigma_d^G(A, q') \\ q' = \delta_A(p_j(s), q)}} c_c(\sigma). \tag{2.1}$$

- *The objective function denoted as $c_{sup}^g(.)$ is given by*

$$c_{sup}^g(A) = \sup_{s \in \mathcal{L}_m(A)} c^g(q_{0A}, A, s). \tag{2.2}$$

Basically, the cost of a trajectory is the sum of the occurrence costs of the events belonging to this trajectory to which is added the cost of disabling events on the way to remain in A. If an uncontrollable event is disabled, this renders the cost of a trajectory infinite because the second term of (2.1) becomes infinity. The notation $c_{sup}^g(A)$ represents the worst-case behavior that is possible in submachine A.

3. Review of the DP-optimal problem for one final state. In general, the purpose of optimal control is to study the behavioral properties of a system, to take advantage of a particular structure, and to generate a controller which constrains the system to a desired behavior according to quantitative and qualitative aspects. In the basic setup of supervisory control theory (see [15, 16] and Chapter 3 of [2]), optimality is with respect to set inclusion, and thus all legal behaviors are equally good (zero cost) and illegal behaviors are equally bad (infinite cost). The work in [21] enriches this setup by the addition of quantitative measures in the form of occurrence

and control cost functions, to capture the fact that some legal behaviors are better than others. The problem is then to synthesize a controller that is not only legal, but also "good" in the sense of given quantitative measures. Some other studies appear in [6, 11, 14, 24]. In this section, we present some results of [21] that are necessary for developing the solution procedure for optimal schedulers. Our aim here is not to describe in detail all the theory, which can be found in [19, 20, 21], but to present the principal notations and results that we use in what follows.

DEFINITION 3.1. *A submachine A of G is said to be controllable if $\forall\ q \in Q_A$, such that there exists $s \in \Sigma^*$ and $\delta(s, q_{o_A}) = q$, the following is satisfied:*

$$\forall\sigma((\sigma \in \Sigma_{uc}) \wedge (\delta(\sigma, q)!)) \Rightarrow \delta_A(\sigma, q)!$$

We now define the optimization problem for a single marked state q_m.

DEFINITION 3.2. *$\forall q \in Q$, $A_o \in \mathcal{M}(G, q)$ is an optimal submachine if*

$$c^g_{\sup}(A_o) = \min_{A \in \mathcal{M}(G,q)} c^g_{\sup}(A) < \infty.$$

For such a submachine A_o, $c^g_{\sup}(A_o)$ represents the optimal cost (in fact, the worst inevitable cost) necessary to reach q_m from q_0. It means that a submachine with a lower cost could not ensure the accessibility of q_m from q_0. The following lemma (Lemma 2.15 in [19]) is stated to note that optimal solutions lie within the class of controllable submachines.

LEMMA 3.3. *Let $A \in \mathcal{M}(G, q, Q_m)$. If $c^g_{sup}(A) < \infty$, then A is controllable.*

Theorem 4.2 of [19] gives necessary and sufficient conditions for the existence of optimal submachines as follows.

THEOREM 3.4. *An optimal submachine of G exists if and only if there exists a submachine A of G such that A is trim, controllable and $\forall s \in \mathcal{L}(G)$ and $q \in Q$ such that $\delta(s, q) = q$ we have $c^g(q, A, s) = 0$.*

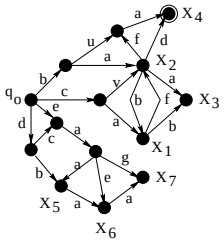
Intuitively, this theorem states that an optimal solution exists when there are controllable submachines of G in which all cycles have a zero cost. The controllability assumption ensures that the positive cost cycles can be broken using controllable events alone. We now introduce the notion of DP-optimal submachines. This kind of submachine will be used intensively in the next sections.

DEFINITION 3.5. *A submachine $A_{DO} \in \mathcal{M}(G, q)$ is DP-optimal if it is optimal and $\forall q' \in Q_{A_{DO}}$, $M(A_{DO}, q')$ is an optimal submachine in $\mathcal{M}(G, q')$.*

If a particular DP-optimal FSM includes all other DP-optimal FSMs as submachines of itself, then we call it the *maximal DP-optimal submachine.* The maximal DP-optimal submachine of a machine G at q with respect to the final marked state q_m will be denoted by $M^o_D(G, q, q_m)$. Note that all DP-optimal submachines are acyclic. The existence of a DP-optimal submachine of G is given by the following theorem (Theorem 4.3 of [19]).

THEOREM 3.6. *If an optimal submachine of G exists, then the unique maximal DP-optimal submachine $G^m_{des} = M^o_D(G, q_0, q_m)$ of G with respect to the final state q_m also exists.*

The cyclic DP-optimal algorithm. Consider an FSM $G = \langle \Sigma, Q, q_0, q_m, \delta \rangle$ with a unique initial state q_0 and a unique marked state q_m. Assume that all occurrence costs are strictly positive; then there exists an algorithm [21], named **DP-Opt**, with a worst-case complexity $\mathcal{O}(|Q|^2|\Sigma| \log(|\Sigma|) + |Q|^3|\Sigma|)$ (Theorem 6.10 of [21]), that constructs the desired maximal DP-optimal submachine of the FSM G with respect to q_0 and q_m, that we denote as G^m_{des}. The algorithm also returns the worst inevitable



Event	c_e	Remarks
a,f	1	Controllable
b,c,d,e	2	Controllable
g	4	Controllable
u	3	Uncontrollable
v	2	Uncontrollable

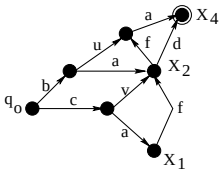


FIG. 3.1. *The initial system G, the event cost function, and the maximal DP-optimal submachine* G^4_{des}.

cost $c^g_{sup}(G^m_{des})$. Moreover, during the computation of the algorithm, we can recover the submachines $M^o_D(G, q, q_m)$ associated with $c^g_{sup}(M^o_D(G, q, q_m))$ for each state visited during the computation. A simplified version of this algorithm can be found in [10] (when the control cost function is reduced to the null function for controllable events).

Example of the DP-optimal problem. We conclude this section by illustrating the DP-optimal problem through an example that is reused in section 6. Let G be an FSM and $\Sigma = \{a, b, c, d, e, f, g, u, v\}$ such that a, b, c, d, e, f, and g are controllable; u and v are uncontrollable. G and the event cost function defined on Σ are as in Figure 3.1. We assume $c_c \in \{0, \infty\}$. Finally, the initial state is q_0 and the final state is X_4.

Using the **DP-Opt** program, we obtain the maximal DP-optimal submachine of G, denoted G^4_{des}, for which the worst inevitable cost is equal to $c^g_{sup}(G^4_{des}) = 6$.

We can observe all the properties of the generated submachine. First, it is controllable, since from any state, there exists a path that leads surely to the goal X_4. Also, it is optimal, since all the paths leading to X_4 have a finite and minimized worst-case cost (notably, no uncontrollable event at state X_4 needs to be disabled). Finally, the DP-optimality property can be observed. From every state q of G^4_{des}, the path from q to X_4 which has the highest cost contains an uncontrollable event u that cannot be disabled.

We have reviewed the optimal control problem and the notion of DP-optimal submachines when only one marked state is present in the system. We now turn our attention to the case of multiple marked states and present our results for this new problem. This will require the introduction of a new, more comprehensive, optimality criterion.

4. The optimal control problem with multiple marked states. In the previous section, we were interested in finding a DP-optimal submachine of G that makes the system evolve from an initial state q_0 to a final state q_m by minimizing a cost function along the various trajectories of the system. Here, our goal is different. We consider an FSM G with a set of multiple marked (or final) states $\mathcal{X} = (X_i)_{i\in[1,\ldots,n]}$. Our aim is now to have the system reach each and every one of the states of $\mathcal{X}$. To account for the fact that it may not be possible to find such a path, we assume in the following the possibility of *resetting* the system to its initial state q_0, when the system has evolved in one of the states of $\mathcal{X}$. The *Reset* event that is added in this section is much more than an artifact for developing the theory. Indeed, many interpretations can be associated with it. First, there are physical systems that can actually be reset to their initial state (like a World Wide Web browser, for example). Second,

the *Reset* event can be seen as an event whose occurrence signals the impossibility of visiting all the states of $\mathcal{X}$ without visiting the initial state q_0 more than once. This apparent impossibility can be alleviated by having multiple systems perform in parallel. For example, in the case of a communication network, a message that is sent cannot be brought back to the initial state. However, it can be regenerated, and then the number of *Reset* events can be regarded as an indicator of the number of copies of the message that must be generated and sent in parallel in a broadcast or a multicast (See section 7).

4.1. Stepwise DP-optimality definition. Due to the *Reset* event, the system is now represented by the following FSM $G = \langle \Sigma \cup \{Reset\}, Q, q_0, \mathcal{X}, \delta \rangle$, with $\delta(Reset, X_i) = q_0 \ \forall X_i \in \mathcal{X}$. As in the previous section, we introduce cost functions that take into account the particular *Reset* event: the occurrence cost function $c_e : \Sigma \cup \{Reset\} \to \mathbb{R}^+ \cup \{0\}$ such that $\forall \sigma \in \Sigma, c_e(\sigma) \geq 0$ and $c_e(Reset) = 0$, and the control cost function $c_c : \Sigma \cup \{Reset\} \to \mathbb{R}^+ \cup \{0, \infty\}$ such that $\forall \sigma \in \Sigma, c_c(\sigma) \geq 0$ and $c_c(Reset) = 0$.

DEFINITION 4.1. *Let $s \in \mathcal{L}_m(G)$. The trajectory s is said to be* valid *if there exists at least n prefixes of s, $(s_i)_{i\in[1,\dots,n]}$, such that $\delta(q_0, s_i) = X_i \in \mathcal{X}$.*

In other words, a trajectory is valid if it makes the system evolve into each of the marked states in $\mathcal{X}$. Note that the definition does not require that the trajectory visit each marked state exactly once. Besides, due to the *Reset* event, the system has the possibility of coming back in its initial state along the trajectory. The set of valid trajectories of the FSM G will be denoted as $\mathcal{S}$.

Given that our primary interest is in the states of $\mathcal{X}$, we introduce the notion of a *valid state trajectory*.

DEFINITION 4.2. *Let s be a valid trajectory in $\mathcal{S}$, such that $s = t_1^s \dots t_l^s$, with $l > n$ and $\delta(q_0, t_1^s \dots t_k^s) = X_k^s \in \mathcal{X} \cup \{q_0\}$. We define the function D from $\mathcal{S}$ into $\{q_0\}(\mathcal{X}^*\{q_0\})^*$, such that $D(s) = (X_k^s)_{k\in[1,\dots,l]}$.*[2] *Such a trajectory is called a* valid state trajectory *with respect to $\mathcal{X}$. We denote as $\mathcal{D}$ the set of valid state trajectories in G, with respect to the set of valid trajectories $\mathcal{S}$: $\mathcal{D} = D(\mathcal{S})$.*

A valid state trajectory $d \in \mathcal{D}$ corresponds to a trajectory in $\{q_0\}(\mathcal{X}^*\{q_0\})^*$ that contains all the states of $\mathcal{X}$ (with possible repetitions).

Since we must deal with a set of marked states rather than with a single marked state, we need to introduce a model that comprises all the states of the set $\mathcal{X}$ and that accounts for the global behavior of the system. It is not possible to use a classical merge operation ($\oplus$, Definition 6.2 in [21]), because states might appear in different submachines in different contexts, i.e., with different partial transition functions associated with them. Therefore, instead of using a merge, we introduce the notion of a scheduler. A scheduler can be thought of as a concatenation of (DP-optimal, in our case) submachines. The role of the scheduler is then to make the system evolve according to one submachine at a time and to account for switching between them at appropriate instants. In what follows, the symbol "$\circ$" will denote the concatenation of two submachines A and A' of G. It is defined in terms of languages. Let $\mathcal{L}_m(A)$ and $\mathcal{L}_m(A')$ be the marked languages of A and A'. Then $\mathcal{L}_m(A \circ A') = \{st \ : \ s \in \mathcal{L}_m(A), \ t \in \mathcal{L}_m(A')\}$. Note that $\mathcal{L}_m(A \circ A') \subseteq \mathcal{L}_m(G)$ if and only if $Q_{m_A} = \{q_{0_{A'}}\}$ and $Q_{m_{A'}} \subseteq Q_{m_G} = \mathcal{X}$. Also note that, due to possible cycles in the FSM G, $A \circ A'$ is in general no longer a submachine of G since some state q of G may be shared by the two submachines A and A' but without the same transitions.

[2] This function allows the "extraction" of the state trajectory in G from the valid trajectory s.

DEFINITION 4.3. *Let $d = (X^d_{k'})_{k' \in [0,\dots,l]} \in \mathcal{D}$ be a valid state trajectory of $\mathcal{X} \cup \{q_0\}$, and let $(A_k)_{k \in [1,\dots,l]}$ such that $l \geq n$ and $A_k \in \mathcal{M}(G, X^d_{k-1}, X^d_k)$ $\forall k \in [1,\dots,l]$; then the structure $A = A_1 \circ A_2 \circ \cdots \circ A_l$ is called a* scheduler *with respect to G and $\mathcal{X}$. The set of schedulers with respect to G and $\mathcal{X}$ is denoted as $\mathcal{M}^{sc}(G, \mathcal{X})$.*

In this particular case, for each submachine of the scheduler, there is only one initial state and one final state. Hence, for two consecutive submachines A_i and A_{i+1}, we have $q_{m_{A_i}} = q_{0_{A_{i+1}}}$. Note that for a scheduler $A = A_1 \circ A_2 \circ \cdots \circ A_l$, some A_k may be simply reduced to the simple FSM $(X^d_k \xrightarrow{Reset} q_0)$. This FSM is clearly a DP-optimal submachine from X^d_k to q_0. Besides, in some cases, $\mathcal{M}^{sc}(G, \mathcal{X})$ can be reduced to $\emptyset$. The cost associated with a scheduler $A = A_1 \circ A_2 \circ \cdots \circ A_l$, denoted as $C^{sc}_{sup}(A)$, is given by

$$C^{sc}_{sup}(A) = \sum_{i=1}^{l} c^g_{sup}(A_i). \tag{4.1}$$

The following definition extends the notion of DP-optimality to the notion of stepwise DP-optimality.

DEFINITION 4.4. *Let $A \in \mathcal{M}^{sc}(G, \mathcal{X})$ be a scheduler, such that A makes the system evolve through a valid state trajectory $d = (X^d_{k'})_{k' \in [0,\dots,l]}$ of $\mathcal{D}$. $A = A_1 \circ A_2 \circ \cdots \circ A_l$ is said to be* stepwise DP-optimal *if each of the submachines $A_k \in \mathcal{M}(G, X^d_{k-1}, X^d_k)$ is DP-optimal with respect to its initial state X^d_{k-1} and final state X^d_k, and if the following condition is satisfied:*

$$C^{sc}_{sup}(A_o) = \min_{A \in \mathcal{M}^{sc}(G,\mathcal{X})} C^{sc}_{sup}(A) < \infty.$$

We wish to draw attention to the following assumption.

Assumption 4.1. From now on, we assume that the DP-optimal submachines under consideration, with the exception of $(X^d_k \xrightarrow{Reset} q_0)$, are maximal. This is done for two main reasons. First, the algorithm **DP-Opt** (see Appendix A of [10]) outputs exactly the maximal DP-optimal submachines. Second, taking the maximal DP-optimal submachines allows the system greater freedom. Indeed, it contains all the other DP-optimal submachines; therefore, it has more possible paths from the initial state to the final marked state. In most applications, it is desirable to lower the probability of taking the worst-case cost path, which is the intent of taking the maximal DP-optimal submachine for $(G^i_{des})_{i \in [1,\dots,n]}$. The more possible paths there are, the less likely it is for the system to take the worst-case cost path. Note that the *Reset* machine $(X^d_k \xrightarrow{Reset} q_0)$ need not be maximal (this can only happen if occurrence costs cannot be equal to zero); in this case, however, given our interpretation of the *Reset* event, we will include the single transition $(X^d_k \xrightarrow{Reset} q_0)$ in the scheduler.

Under this assumption, the following property is a direct consequence of Definition 4.4.

PROPERTY 4.2. *Let G be an FSM, and let $\mathcal{X}$ be the set of marked states of G. Let A be a stepwise DP-optimal scheduler, such that $A = A_1 \circ A_2 \circ \cdots \circ A_l$. Let $d = (X^d_k)_{k \in [0,\dots,l]}$ of $\mathcal{D}$ be the associated valid state trajectory. Then $\forall k \in [1,\dots,l]$, $A_k = M^o_D(G, X^d_{k-1}, X^d_k)$. Furthermore, the global cost of the scheduler is*

$$C^{sc}_{sup}(A) = \sum_{k=1}^{l} c^g_{sup}(M^o_D(G, X^d_{k-1}, X^d_k)) < \infty. \tag{4.2}$$

This property states that if a stepwise DP-optimal scheduler exists, then all the submachines constituting this scheduler are the respective $M_D^o(G, X_{k-1}, X_k)$. Moreover the cost of the scheduler is then simply equal to the sum of the costs of these DP-optimal submachines. We will refer to this important result as the *additivity property* of the stepwise DP-optimal scheduler. In what follows, the set of all schedulers A such that all the submachines of A are of the form $M_D^o(G, X_i, Xj)$ for $X_i, X_j \in \mathcal{X} \cup \{q_0\}$, is denoted $\mathcal{M}_D^{sc}(G, \mathcal{X})$.

Now that we have defined the notion of a stepwise DP-optimal scheduler and given some of its properties, we need to give necessary and sufficient conditions for its existence. The next subsection gives these conditions and also proves desirable properties of such a scheduler.

4.2. Existence of a stepwise DP-optimal scheduler. Theorem (4.7) presented below gives necessary and sufficient conditions for the existence of a stepwise DP-optimal scheduler. First we prove the following lemma.

LEMMA 4.5. *If the DP-optimal submachines $M_D^o(G, X_i, X_j)$ and $M_D^o(G, X_j, X_k)$ of G exist, then there exists a DP-optimal submachine $M_D^o(G, X_i, X_k)$. Moreover, we have the following* triangular inequality:

$$c_{sup}^g(M_D^o(G, X_i, X_k)) \leq c_{sup}^g(M_D^o(G, X_i, X_j)) + c_{sup}^g(M_D^o(G, X_j, X_k)). \tag{4.3}$$

Proof. Assume the existence of $M_D^o(G, X_i, X_j) = \langle \Sigma_{ij}, Q_{ij}, X_i, X_j, \delta_{ij} \rangle$ and of $M_D^o(G, X_j, X_k) = \langle \Sigma_{jk}, Q_{jk}, X_j, \{X_k\}, \delta_{jk} \rangle$. Consider the intersection of the states of these two submachines as being $Q_{ij} \cap Q_{jk} = \{X_j, q_1, \ldots, q_n\}$. Note that this intersection might be reduced to $\{X_j\}$. We construct a new submachine $G_{ik} = \langle \Sigma_{ik}, Q_{ik}, q_{0_{ik}}, Q_{m_{ik}}, \delta_{ik} \rangle$ from these submachines:

$$G_{ik} = \begin{cases} \Sigma_{ik} = \Sigma_{ij} \cup \Sigma_{jk}, \; Q_{ik} = Q_{ij} \cup Q_{jk}, \\ q_{0_{ik}} = X_i, \qquad Q_{m_{ik}} = \{X_k\}, \\ \delta_{ik}(\sigma, q) = \begin{cases} \delta_{jk}(\sigma, q) \text{ if it exists and } q \in Q_{jk}, \\ \delta_{ij}(\sigma, q) \text{ if it exists and } q \in Q_{ij} - \{X_j, q_1, \ldots, q_n\}, \\ \text{undefined otherwise.} \end{cases} \end{cases}$$

This submachine G_{ik} is well defined. Any possible ambiguity has been eliminated by separately dealing with the states $\{X_j, q_1, \ldots, q_n\}$ in the definition of δ_{ik}. G_{ik} is obtained by always following the partial transition function of $M_D^o(G, X_j, X_k)$ as a default behavior, and following the partial transition function of $M_D^o(G, X_i, X_j)$ otherwise whenever possible. First, the machines $M_D^o(G, X_i, X_j)$ and $M_D^o(G, X_j, X_k)$ are trim. Second, G_{ik} is constructed by forward propagation; therefore, all the states of G_{ik} are accessible with respect to the initial state X_i and are coaccessible with respect to the marked state X_k. Therefore, G_{ik} is trim.

Moreover, G_{ik} is controllable. Indeed, the partial transition function δ_{ik} says that as long as the system has not reached a state of the set $\{X_j, q_1, \ldots, q_n\}$, it follows the partial transition function of δ_{ij}. Due to the DP-optimality of $M_D^o(G, X_i, X_j)$, the system will always reach a state of the set $\{X_j, q_1, \ldots, q_n\}$ with a finite cost. Indeed, if the system never visits a state in $\{q_1, \ldots, q_n\}$, it will eventually reach X_j. Let us call q the first state of the set $\{X_j, q_1, \ldots, q_n\}$ that is visited by the system as it evolves. At this point, the default partial transition function becomes δ_{jk}; therefore, the system will eventually reach the marked state X_k with a finite cost since the submachine $M_D^o(G, X_i, X_j)$ is DP-optimal. Since the cost of reaching X_k from q is

finite, the overall cost of reaching X_k is necessarily finite. From Lemma 3.3, G_{ik} is controllable.

Finally, G_{ik} has no positive cost cycles. $M_D^o(G, X_i, X_j)$ and $M_D^o(G, X_i, X_k)$ do not have positive cost cycles (by definition of DP-optimality). As we have described previously, before the system reaches a state of $\{X_j, q_1, \ldots, q_n\}$ for the first time, it will not complete a positive cost cycle (from the DP-optimal nature of $M_D^o(G, X_i, X_j)$). After the system reaches a state of $\{X_j, q_1, \ldots, q_n\}$ for the first time, it will not complete a positive cost cycle either (from the DP-optimal nature of $M_D^o(G, X_j, X_k)$). Therefore, no new cycles have been introduced. The only cycles that may exist in G_{ik} are those of $M_D^o(G, X_i, X_j)$ and $M_D^o(G, X_j, X_k)$.

Given that G_{ik} is trim, controllable, and contains no cycles of positive cost in G, FSM G_{ik} satisfies the preconditions of Theorem (3.4), and there exists an optimal submachine of G_{ik}. Following Theorem 3.6, there also exists a DP-optimal submachine $M_D^o(G, X_i, X_k)$ of G_{ik}.

The proof of the *triangular inequality* relies on what we have said previously. The cost of reaching a state of the set $\{X_j, q_1, \ldots, q_n\}$, from the initial state X_i, is less than $c^g_{sup}(M_D^o(G, X_i, X_j))$ (equality is possible but not necessary when X_j is reached). Once one of the states $\{X_j, q_1, \ldots, q_n\}$ has been reached, the cost for the system to reach the marked state X_k is less than $c^g_{sup}(M_D^o(G, X_j, X_k))$ (equality is possible but not necessary when the system visits X_j) because the corresponding machine is DP-optimal. More formally, let us take a trace s of events that leads from the initial state X_i to the final state X_k, i.e., such that $\delta_{ik}(s, X_i) = X_k$. As seen earlier, s visits at least one state of the set $\{X_j, q_1, \ldots, q_n\}$. Let us call it q again. We can now subdivide s into s_1 and s_2 such that $s = s_1 s_2$, $\delta_{ik}(s_1, X_i) = q$, and $\delta_{ik}(s_2, q) = X_k$. From the DP-optimality of the two submachines $M_D^o(G, X_i, X_j)$ and $M_D^o(G, X_j, X_k) \forall s$ such that $s = s_1 s_2,\ \delta_{ik}(s_1, X_i) = q,\ \delta_{ik}(s_2, q) = X_k$, we can compare

$$\begin{cases} c^g(X_i, M_D^o(G, X_i, X_j), s_1) \leq c^g_{sup}(M_D^o(G, X_i, X_j)), \\ c^g(q, M_D^o(G, X_j, X_k), s_2) \leq c^g_{sup}(M_D^o(G, X_j, X_k)). \end{cases}$$

Since this is true for all traces leading from X_i to X_j, we can deduce the triangular inequality. □

The following corollary uses the construction in the proof of Lemma 4.5 to introduce a necessary condition for the existence of a stepwise DP-optimal scheduler. The proof is straightforward and can be found in [10].

COROLLARY 4.6. *If G^k_{des} does not exist, then there exists no subscheduler that makes the system evolve from q_0 to X_k, should it be indirectly via states of $\mathcal{X}$.*

As a consequence of these results, we can ensure that a state X_k is accessible in an optimal way if and only if G^k_{des} exists. We are now able to give the necessary and sufficient conditions of the existence of a stepwise DP-optimal scheduler. This is stated by Theorem 4.7.

THEOREM 4.7. *There exists a corresponding stepwise DP-optimal scheduler $A \in \mathcal{M}_D^{sc}(G, \mathcal{X})$ if and only if the n DP-optimal submachines G^i_{des} of G exist $\forall X_i \in \mathcal{X}, i \in [1, \ldots, n]$.*

Proof. The necessary condition is given by Corollary 4.6, which states that if there is a state X_i of $\mathcal{X}$ such that there does not exist a DP-optimal submachine G^i_{des}, then there is no way to reach this state with a finite cost (thus in a DP-optimal way) and the goal cannot be achieved. All the states of $\mathcal{X}$ cannot be visited, since one of them cannot be visited. The condition is sufficient since FSM A, such that

$A = G^1_{des} \circ (X_1 \xrightarrow{Reset} q_0) \circ G^2_{des} \circ (X_2 \xrightarrow{Reset} q_0) \circ \cdots \circ G^n_{des} \circ (X_n \xrightarrow{Reset} q_0)$, visits all the states of $\mathcal{X}$. A is then a possible scheduler allowing the achievement of the goal. □

This theorem implies that the stepwise DP-optimal problem has a solution when there exists a DP-optimal submachine for each of the X_i. Besides, if a stepwise DP-optimal solution exists, it need not be unique in general. There is no notion of a maximal stepwise DP-optimal scheduler, as in the DP-optimal problem [21]. The problem of finding one of the optimal schedulers is now explored.

5. Determination of a stepwise DP-optimal scheduler. In this section, we need to assume that the occurrence costs are strictly positive: $\forall \sigma \in \Sigma,\ c_e(\sigma) > 0$. This assumption is necessary when we use the **DP-Opt** algorithm in order to ensure polynomial complexity. We also assume that a DP-optimal submachine exists for all the states $X_i \in \mathcal{X}$. From here on, G^i_{des} will denote the maximal DP-optimal submachine of the particular FSM $G_i = \langle \Sigma, Q, q_0, X_i, \delta \rangle$ output by the **DP-Opt** algorithm. We take advantage of the DP-optimal structure of each of the G^i_{des}. We explore the possibility of starting the system at q_0, reaching a state X_i, and instead of doing a *Reset*, continuing the graph to a state X_j. To do so, we convert the problem to a path-cost minimization problem on a graph equivalent to a TSP.

5.1. Modeling of the problem. In order to convert the stepwise DP-optimal problem into a path-cost minimization problem, we use the **DP-Opt** algorithm. This algorithm computes for each $X_i \in \mathcal{X}$ the DP-optimal submachine G^i_{des}. During this computation, a state X_j belonging to $\mathcal{X}$ may be reached. Due to the DP-optimality definition, the algorithm also gives the DP-optimal submachine between X_j and X_i. The worst inevitable case cost between these two states can be collected as well and placed in a matrix $C \in \mathbb{R}^{n+1} \times \mathbb{R}^{n+1}$ that has the following form (see [10] for the algorithm and further details):

$$\begin{array}{ll} \bullet\ C[i,i] = \infty, & \bullet\ C[i,0] = 0, i \neq 0, \\ \bullet\ C[0,i] = c^g_{sup}(G^i_{des}), i \neq 0, & \bullet\ C[k,i] = \begin{cases} c^g_{sup}(M^o_D(G, X_k, X_i)) \text{ if it exists,} \\ \infty \text{ otherwise.} \end{cases} \end{array} \tag{5.1}$$

From additivity Property 4.2, the cost of a scheduler $A = A_1 \circ A_2 \circ \cdots \circ A_l$ of $\mathcal{M}^{sc}_D$ is equal to

$$C^{sc}_{sup}(A) = \sum_{k=1}^{l} c^g_{sup}(A_k) \ = \ \sum_{k=1}^{l} c^g_{sup}(M^o_D(G, X_{d_{k-1}}, X_{d_k})) \ = \ \sum_{k=1}^{l} C[d_{k-1}, d_k]. \tag{5.2}$$

Considering (5.2), the new optimization problem is now reduced to finding a path with a minimal cost in the directed graph associated with the matrix C. This closely resembles the TSP with the slight difference that multiple visits to states of $\mathcal{X}$ are possible. In this new problem, the "cities" are represented by the set of nodes $\mathcal{X}$, and the "streets" are represented by machines $(G^i_{des})_{i \in [1,\ldots,n]}$ and $M^o_D(G, X_i, X_j)$ when these are available. The costs of these paths are given by the maximum costs for each machine, i.e., the $(c^g_{sup}(G^i_{des}))_{i \in [1,\ldots,n]}$ and the $(c^g_{sup}(M^o_D(G, X_i, X_j)))_{i,j \in [1,\ldots,n]}$. Figure 5.1 illustrates this conversion from the graph of the FSM to the reachability graph.

Note that some elements of C might be equal to ∞ after all G^i_{des} have been computed, which does not mean that the corresponding DP-optimal submachines do not exist. This means that they have not been computed in the algorithm **DP-Opt**.

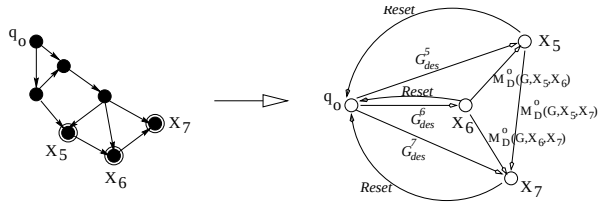


FIG. 5.1. *Conversion from the FSM to a reachability graph on the marked states.*

Indeed, let us suppose that some $M_D^o(G, X_i, X_j)$ has not been computed (and that therefore $C[i, j] = \infty$). It means that it is less costly to perform a *Reset* from state X_i to state q_0, and to reach X_j through G_{des}^j. Another way of seeing this is to look at what the **DP-Opt** algorithm does. It backtracks from the marked state, say X_j. If it reaches X_i before q_0, this means that the cost from X_i to X_j is less than the cost from q_0 to X_j, in which case it is less costly to go directly from X_i to X_j than to reset the system. On the other hand, if state X_i is not reached when the algorithm reaches q_0 during its backtracking, it means that the cost to go from X_i to X_j is greater than the cost of resetting the system (0 in our case) and taking the DP-optimal submachine G_{des}^j. This explains why these paths are not taken into account as possible paths and are directly replaced in the matrix C by an infinite cost.

5.2. Generation of the stepwise DP-optimal scheduler. The problem of finding a stepwise DP-optimal scheduler A_0 has been brought down to solving an instance of the TSP, a classic combinatorial optimization problem. Many methods exist to solve the problem in an acceptable amount of time [8]. We specify once more the conditions in which we solve the TSP. The costs of the paths are all nonnegative. The nodes of $\mathcal{X}$ must be visited *at least once*. One requirement of the TSP is that the salesman come back to the city he started from. This condition does not change anything in our problem since this maps to a *Reset*, which has null cost in our model. Finally, note that the cost matrix C, given in section 5.1, is not necessarily symmetric.

The first step is to transform our modified version of the TSP, where we can visit a node more than once (but at least once), into an ordinary TSP where we must visit each node exactly once. This is typically done by transforming the matrix C into a matrix C', called the all-pairs shortest-paths matrix [8]. Many techniques exist to perform such a computation. Among them is the Floyd–Warshall algorithm [8], which runs in a worst case of $\mathcal{O}(n^3)$, where n represents the number of vertices. Once the all-pairs shortest-paths matrix C' is obtained, we can feed it to a TSP solver.

C and C' have the same dimension but represent different features of the graph. C contains, as noninfinite elements, the costs of the links that actually exist in the graph of the TSP. C' contains the minimum costs necessary to go from one marked state to another, along DP-optimal submachines. C' is a reachability matrix, whereas C is a connectivity matrix. Notably, C' shows if states can be reached by using the *Reset* event. Concretely, to obtain C' from C, one only needs to replace any infinite value in C by the value in the same column in the first line (the cost of the G_{des}^i associated with column i).

Resolution of the TSP. The actual solving of the TSP from matrix C' can be done by using several methods. The most common method is the branch and bound method (see Chapters 9 and 10 of [9] and [12, 7]). An outline of the method can be found in Appendix B of [10]. The worst-case complexity for solving the TSP is

$(n+1)!$. However, the branch and bound method is expected to give a solution to the TSP in a tolerable amount of time. (To give a feel of the time complexity of this method, a 1,000 node fully-connected TSP can be solved in about 20 minutes on a standard workstation.)

The principle of the branch and bound method is quite natural. A branching strategy and a bounding strategy are used alternatively. The branching strategy consists of forcing a supplementary constraint to the system, usually by forcing a set of subpaths in the graph. This allows us to find a solution that is suboptimal in general but that is sometimes optimal. The bounding strategy focuses on finding a lower bound on the cost of the optimal solution by relaxing one of the constraints of the problem (usually by relaxing the constraint that the solution must be a tour). The branching yields a search tree, and the bounding yields a way of quickly finding a suboptimal solution which is close to the optimal solution of the problem. The final solution is optimal.

5.3. Restitution of the stepwise DP-optimal scheduler. From a solution of the TSP, we now build a corresponding stepwise DP-optimal scheduler. The resolution of the TSP provides an optimal solution that gives the ordering in which the states should be visited so as to minimize the worst-case cost. A solution is under the form of a set of $n+1$ pairs (there are $n+1 = |\mathcal{X} \cup \{q_0\}|$ states), in which each state appears exactly once as an initial state and exactly once as a final state of a pair. For pairs (X_i, X_j) that represent a physically existing submachine $M_D^o(G, X_i, X_j)$, i.e., for which $C[i,j] < \infty$, it is sufficient to map these pairs to their associated submachine. As for the pairs (X_i, X_j) that do not map to an existing DP-optimal submachine, i.e., those for which $C[i,j] = \infty$ and $C'[i,j] < \infty$, they are divided into two pairs, namely, (X_i, q_0) and (q_0, X_j). The first is mapped to a *Reset* to the initial state, and the second is mapped to the DP-optimal submachine G_{des}^j.

THEOREM 5.1. *Given a solution of the TSP, by adopting the previous mapping, the obtained scheduler is stepwise DP-optimal.*

Proof. The initial solution of the TSP with respect to the matrix C' is given by a tour of the form $\{(q_0, X_{i_1}); (X_{i_1}, X_{i_2}); \dots (X_{i_j}, X_{i_{j+1}}); \dots; (X_{i_n}, q_0)\}$ with a corresponding cost $TSP(C') = C'[0, i_1] + C'[i_1, i_2] + \cdots + C'[i_j, i_{j+1}] + \cdots + C'[i_n, 0]$. Consider now the transformation previously adopted. If the pair (X_i, X_j) originally exists, i.e., $C[i,j] < \infty$, then the path is admissible in the original problem and we replace the pair by the submachine $M_D^o(G, X_i, X_j)$, where the corresponding cost $c_{sup}^g(M_D^o(G, X_i, X_j))$ is equal to $C[i,j]$. If the pair (X_i, X_j) does not map to an existing DP-optimal submachine, i.e., $C[i,j] = \infty$, then we need to *Reset* the system before directly going to X_j through G_{des}^j. The *triangular inequality* of Lemma (4.5) ensures that in this case, $C'[i,j] = c_{sup}^g(G_{des}^j)$. The pair is then replaced by the subscheduler $(X_i \xrightarrow{Reset} q_0) \circ G_{des}^j$, with the corresponding cost equal to $c_{sup}^g(G_{des}^j)$.

We then obtain a new sequence of pairs, with a cost equal to $TSP(C')$ but for which all the submachines actually exist in the original problem. The DP-optimality of each submachine of the scheduler is given by construction, since we only consider submachines of the form $M_D^o(G, X_i, X_j)$ or G_{des}^j. The minimal cost of the scheduler is ensured by the optimality of the TSP solution and by the fact that the mapping does not add new costs. □

An interesting property is given next. It states that all the submachines that constitute a stepwise DP-optimal scheduler are directly derived from all the DP-optimal submachines built during the computation of the matrix C (see section 5.1 and (5.1)).

PROPOSITION 5.2. *A stepwise DP-optimal scheduler A_o obtained by the TSP solution is composed of exactly n different DP-optimal submachines (not counting the possible Resets of the system). Moreover, all these submachines are obtained from the DP-optimal submachines $(G^i_{des})_{i\in[1,\dots,n]}$ computed during the matrix generation step (see* (5.1)*).*

Proof. The general solution of the TSP for the matrix C' is a tour of the form $\{(q_0, X_{i_1}); (X_{i_1}, X_{i_2}); \dots; (X_{i_j}, X_{i_j}); \dots; (X_{i_n}, q_0)\}$. Note that there are exactly $n+1$ pairs in this tour (but the last pair is a trivial one, i.e., a *Reset*). If a pair (X_i, X_j) originally exists, i.e., if $C[i,j] < \infty$, then the path is in the original problem and it is replaced by the submachine $M^o_D(G, X_i, X_j)$. If not, i.e., if $C[i,j] = \infty$, then the system is reset before directly going to X_j through G^j_{des}. The pair is then replaced by the subscheduler $(X_i \xrightarrow{Reset} q_0) \circ G^j_{des}$. The n pairs are then replaced by either the DP-optimal submachine $M^o_D(G, X_i, X_j) = Trim(G^j_{des}, X_i, X_j)$, or by the subscheduler $(X_i \xrightarrow{Reset} q_0) \circ G^j_{des}$. The final solution of our problem has then exactly n nontrivial submachines that can be obtained from the n DP-optimal submachine $(G^i_{des})_{i=[1,\dots,n]}$ of G, by a trim operation. □

COROLLARY 5.3. *In a stepwise DP-optimal scheduler obtained by the TSP solution, the states of $\mathcal{X}$ are visited exactly once by the stepwise DP-optimal scheduler.*[3]

We wish to draw attention to the following fact. The stepwise DP-optimal scheduler visits each marked state exactly once when it is obtained from the TSP solution. However, the system itself, through its evolution described by the FSM G, may visit a marked state of G more than once. This comes from the fact that the scheduler is constructed on C', whereas the behavior of the system modeled by the FSM G should be observed at a less abstract level, namely at the level of the FSM, G.

Remark 5.1. In [10], we also presented the resolution of the stepwise DP-optimal problem in the case of a nonzero occurrence cost for the *Reset* event (see section 5.3 of [10] for further details).

5.4. Some simplifications of the TSP resolution. In order to solve the stepwise DP-optimal problem, we have to solve the corresponding TSP for the matrix C. The TSP is an NP-complete problem. It is then greatly advantageous to find some simplification methods, taking advantage of the special structure of a stepwise DP-optimal scheduler, in order to reduce the computational complexity of the corresponding TSP without loss of global optimality. Proofs and algorithms are omitted in this section due to lack of space. They can be found in the companion paper [10].

5.4.1. Divide and conquer. In some cases, it is possible to divide the matrix C into several smaller ones. In such cases, it suffices to solve the TSP on each of these submatrices. The following proposition states the necessary and sufficient conditions for this simplification.

PROPOSITION 5.4. *Assume there exists a partition of $\mathcal{X} = \cup_{k\in[1,\dots,l]}(\mathcal{X}_k)$ such that $\forall k_1, k_2 \in [1,\dots,l]$, $\forall X_i \in \mathcal{X}_{k_1}$, and $\forall X_j \in \mathcal{X}_{k_2}$, the submachine $M^o_D(G, X_i, X_j)$ is not defined.*

If A_o is a stepwise DP-optimal scheduler with respect to $\mathcal{X}$, then it is possible to find a set of schedulers $A_{\mathcal{X}_k}$, where each $A_{\mathcal{X}_k}$ is stepwise DP-optimal with respect to $\mathcal{X}_k$ with an optimal cost $c^{sc}_{sup}(A_{\mathcal{X}_k})$ to visit of all the states of $\mathcal{X}_k$, and such that $A_o = \circ^l_{k=1} A_{\mathcal{X}_k}$ with $c^{sc}_{sup}(A_o) = \sum^l_{k=1} c^{sc}_{sup}(A_{\mathcal{X}_k})$.

[3]The proof is omitted and can be found in [10].

In view of Proposition 5.4, the global problem can be solved on each submatrix C_k, $k \in [1, \ldots, l]$, corresponding to the particular set of states $\mathcal{X}_k \cup \{q_0\}$. The necessary computation to find the connected components before applying Proposition 5.4 can be performed in $\mathcal{O}(n + E)$, where E is the number of vertices of the directed graph associated to matrix C (see [10] for details).

5.4.2. Terminal path simplification. We address here a property of the scheduler that can lead to a simplification on the matrix C. This property states that if there exists a kind of "dead-end" in the graph of the matrix, then it is always better to follow this path until the end than to perform a *Reset* and come back to visit the end of this path later.

PROPOSITION 5.5. *Assume that there exists a subset $\mathcal{X}_i = (X_{i_k})_{k \in [1,\ldots,m]}$ of $\mathcal{X}$, with $m < n$ and such that*

1. $\forall k \in [1, \ldots, m-1]$, $M_D^o(G, X_{i_k}, X_{i_{k+j}})$ *exists for* $j \in [1, \ldots, m-k]$,
2. $\forall k \in [2, \ldots, m]$, $M_D^o(G, X_{i_k}, X_{i_{k-j}})$ *does not exist for* $j \in [1, \ldots, k-1]$,
3. $\forall k \in [1, \ldots, m]$ *and* $\forall X_l \in \mathcal{X} - \mathcal{X}_i$, $M_D^o(G, X_{i_k}, X_l)$ *is not defined.*

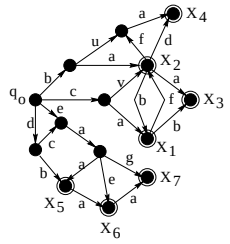
Under these assumptions, the submachines $(G_{des}^{i_k})_{k \in [2,\ldots,m]}$ do not belong to the stepwise DP-optimal scheduler A_o.

Proposition 5.5 deals with situations where a dead-end occurs. By dead-end, we mean a set of states $\{q_1, \ldots, q_n\}$ in which $\forall i \in [1, \ldots, n]$, $(q_j)_{j>i}$ are the only states coaccessible from q_i. If there exists a dead-end in the graph of marked states, the system will never enter that dead-end directly through one of the G_{des}^i but will only enter indirectly from the initial state q_0. This means that no direct submachine of the type G_{des}^i will be used by a scheduler to enter a dead-end. Any visit to a state of the dead-end is done via a visit to a state that does not belong to the dead-end.

An algorithm that performs this simplification on the matrix C according to the three assumptions in Proposition 5.5 is presented in [10]. Its complexity is linear in the number of states of $\mathcal{X}$. With this simplification, the paths of the form $q_0 \to X_{i_k}$ do not constitute valid paths any longer and, consequently, will not be taken into account as possible solutions in the corresponding TSP solution. This terminal path simplification can narrow down the search space when solving the TSP.

5.4.3. Predefined partial order for the visit of $\mathcal{X}$. Throughout section 5, we have assumed that we had no prespecified order in which to visit the marked states in $\mathcal{X}$. This may be the case in several applications. However, in other applications, such as test-generation, we may be interested in the path taken by a system more than in the final state it reaches. The designer may want to enforce the system to follow a given path. The path would be characterized by the states it traverses, which would be marked. This would yield an ordering, not on the marked states, but on the subpaths themselves. A possible extension of this predefined partial order assumption would be to consider our problem in a hierarchical setting in the same spirit as in [23, 22].

Let us consider a simple example. Assume that we have the marked states $\{X_1, \ldots, X_{10}\}$ to visit in an optimal way. If we do not prespecify the order in which they should be visited, the TSP will be solved on a 10×10 matrix. The designer may want to observe the behavior of the system when it visits states X_1 through X_3 in that order, X_4 through X_7 in that order, and X_7 through X_{10} in that order. This would reduce the TSP to a 3×3 matrix, abstracting away from the ten states to four macrostates: $\{q_0\}, \{X_1, X_2.X_3\}, \{X_4, X_5, X_6, X_7\}$, and $\{X_8, X_9, X_10\}$. The solution thus obtained will not be stepwise DP-optimal per se. It will be optimal given the additional constraints imposed by the designer.



Event	c_e	Remarks
a,f	1	Controllable
b,c,d,e	2	Controllable
g	4	Controllable
Reset	0	Controllable
u	3	Uncontrollable
v	2	Uncontrollable

FIG. 6.1. *The initial system G and the event cost function.*

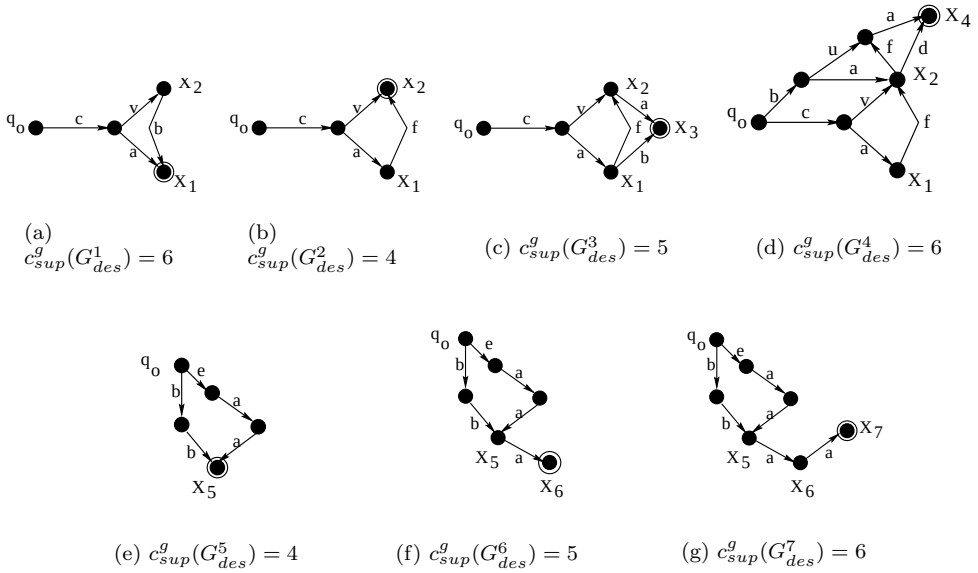


FIG. 6.2. *The DP-optimal FSMs for the different state of* $\mathcal{X}$.

6. Example. The following example is constructed to illustrate the essential stages of the optimal control problem for multiple marked states to visit. For the sake of simplicity, we have assumed that all control costs have zero cost for controllable events and infinite costs for uncontrollable events. We here consider a system modeled by the FSM G, which represents its legal behavior. In this example, there are seven states denoted $(X_i)_{i\in[1,\dots,7]} = \mathcal{X}$ to visit in no particular order. Some costs are allocated to each of the events of the FSM G. The event costs and their status (controllable or not) are as depicted in Figure 6.1.

Note that in Figure 6.1, the *Reset* events are not represented but exist between each of the $(X_i)_{i\in[1,\dots,7]}$ and the initial state q_0. The first phase of the algorithm consists of computing the various DP-optimal submachines $(G^i_{des})_{i\in[1,\dots,7]}$ for each of the final states of $\mathcal{X}$. This part is performed using the **DP-Opt** algorithm (see Appendix A of [10]). The seven figures given next (Figure 6.2) correspond to the DP-optimal submachines for each of the final states $(X_i)_{i\in[1,\dots,7]}$. We also give the worst inevitable cost for each submachine $(G^i_{des})_{i\in[1,\dots,7]}$.

According to (5.1) in section 5.1, we obtain the matrix C, encoding the worst inevitable cost between two states X_i and X_j.

	q_0	X_1	X_2	X_3	X_4	X_5	X_6	X_7
q_0	∞	**6**	**4**	**5**	**5**	**4**	**5**	**6**
X_2	0	**2**	∞	**1**	**2**	∞	∞	∞
X_3	0	∞	∞	∞	∞	∞	∞	∞
X_4	0	∞	∞	∞	∞	∞	∞	∞
X_5	0	∞	∞	∞	∞	∞	**1**	**2**
X_6	0	∞	∞	∞	∞	∞	∞	**1**
X_7	0	∞	∞	∞	∞	∞	∞	∞

Following Proposition 5.4, we can see that $\mathcal{X}_1 = \{X_1, X_2, X_3, X_4\}$ and $\mathcal{X}_2 = \{X_5, X_6, X_7\}$ form a partition of the set of states $\mathcal{X} = (X_i)_{i\in[1,\dots,7]}$. Moreover, the states of $\mathcal{X}_2$ satisfy Proposition 5.5. Thus, in order to solve the TSP, we can now consider the two following matrices. Note that in the second one, we have replaced $C[0,6]$ and $C[0,7]$ by ∞ as stated by Proposition 5.5.

	q_0	X_1	X_2	X_3	X_4
q_0	∞	**6**	**4**	**5**	**5**
X_1	0	∞	**1**	**2**	**3**
X_2	0	**2**	∞	**1**	**2**
X_3	0	∞	∞	∞	∞
X_4	0	∞	∞	∞	∞

	q_0	X_5	X_6	X_7
q_0	∞	**4**	∞	∞
X_5	0	∞	**1**	**2**
X_6	0	∞	∞	**1**
X_7	0	∞	∞	∞

One solution (there are several) of the TSP for each submatrix is

$$\{(q_0, X_4); (X_4, X_1); (X_1, X_2); (X_2, X_3); (X_3, q_0)\},$$
$$\{(q_0, X_5); (X_5, X_6); (X_6, X_7); (X_7, q_0)\}.$$

This is the output of the TSP resolution method run on each of the subproblems. The optimal worst-case costs are 13 and 6, respectively. From Theorem (5.1), there exist two stepwise DP-optimal schedulers A_{0_1} and A_{0_2}. We need to retrieve them from the output of the resolution of the TSP and build a global stepwise DP-optimal scheduler A_0.

We look at each one of the pairs and see if they correspond to a DP-optimal submachine. In this case $(q_0, X_4), (X_1, X_2), (X_2, X_3)$, and (X_3, q_0) correspond to G^4_{des}, $M^o_D(G, X_1, X_2)$, $M^o_D(G, X_2, X_3)$, and $X_3 \xrightarrow{Reset} q_0$, respectively. (X_4, X_1) does not have any associated DP-optimal submachine. We decompose it: (X_4, X_1) becomes $(X_4 \xrightarrow{Reset} q_0)$ concatenated with G^1_{des}. An optimal DP-optimal scheduler A_{o_1} is the following:

$$A_{o_1} = G^4_{des} \circ (X_4 \xrightarrow{Reset} q_0) \circ G^1_{des} \circ M^o_D(G, X_1, X_2) \circ M^o_D(G, X_2, X_3) \circ (X_3 \xrightarrow{Reset} q_0).$$

For the second subproblem of the divide and conquer method, we also map the pairs to original DP-optimal submachines, yielding scheduler A_{o_2}.

$$A_{o_2} = G^5_{des} \circ M^o_D(G, X_5, X_6) \circ M^o_D(G, X_6, X_7) \circ (X_7 \xrightarrow{Reset} q_0).$$

Note that we use exactly four DP-optimal submachines for the first subproblem and three for the second, as expected from Proposition 5.3. We finally generate the

global stepwise DP-optimal scheduler A_o as the concatenation of the two subschedulers A_{o_1} and A_{o_2}, yielding the following scheduler, A_o.

$$\begin{aligned} A_o &= A_{o_1} \circ A_{o_2} \\ &= G^4_{des} \circ (X_4 \overset{Reset}{\longrightarrow} q_0) \circ G^1_{des} \circ M^o_D(G, X_1, X_2) \circ M^o_D(G, X_2, X_3) \circ (X_3 \overset{Reset}{\longrightarrow} q_0) \\ &\quad \circ G^5_{des} \circ M^o_D(G, X_5, X_6) \circ M^o_D(G, X_6, X_7) \circ (X_7 \overset{Reset}{\longrightarrow} q_0). \end{aligned}$$

In fact, this scheduler is actually composed of three different nontrivial subschedulers. The stepwise DP-optimal A_o can be rewritten as

$$\begin{aligned} A_o &= G^4_{des} \circ (X_4 \overset{Reset}{\longrightarrow} q_0) \circ A_1 \circ (X_3 \overset{Reset}{\longrightarrow} q_0) \circ A_2 \circ (X_7 \overset{Reset}{\longrightarrow} q_0). \\ \text{(6.1)} \quad where \begin{cases} A_1 &= M^o_D(G, X_1, X_2) \circ M^o_D(G, X_2, X_3), \\ A_2 &= G^5_{des} \circ M^o_D(G, X_5, X_6) \circ M^o_D(G, X_6, X_7). \end{cases} \end{aligned}$$

This last expression shows the minimum number of *Reset*s that are necessary to visit all the X_i in an optimal way (three, in this case).

7. Potential applications of the theory. Applications of the theory that we have elaborated cover various fields of engineering. One application that can be developed from the theory is test objective generation. In test objective generation, the goal is to check whether a particular system meets the expectations or the requirements that are associated with it. In this framework, the states of interest may be states in which the system is suspected to behave incoherently or incorrectly, or states in which misbehavior could be dramatic or dangerous. These would be the states that would be marked. The theory that we developed allows us to visit all these states and to test the behavior of the system in each one. Once the system has reached one of the marked states, all the known events can be disabled to check if the system stops or enters a forbidden state. A timeout can be set, for example. If the system has not behaved incoherently after that timeout, we can decide to pursue the visit of the marked states. Other more involved strategies can be applied to determine whether a state is faulty or not. For each state, either the behavior of the system is acceptable, or it is not. In the first case where the state is flawless, the next submachine of the scheduler is activated in order to make the system evolve in the next state of interest to be tested. In the case where a failure has been detected in the state, either we stop since the system is faulty and does not correspond to the awaited specifications, or we proceed to determine other possible faults. To do so, we *reset* the system to its initial state q_0, and go directly to the next state, say X_i, through its direct DP-optimal submachine, namely G^i_{des}, and the process continues.

Another application area is planning in the case of multiple goals in AI. Several search algorithms exist when one unique goal is sought (see part II of [18]). Planning in the case of multiple goals remains challenging and interesting. The framework in which we have developed the theory allows goals to be independent or related. Once again, the *Reset* event has an interesting interpretation in AI. It represents the impossibility to meet all the goals without returning to the initial state. It may represent the possibility of using several agents to achieve the goals, each running in parallel. The number of *Reset* events gives the necessary and sufficient number of agents that are needed to perform the goal of reaching all the subgoals in parallel, without any conflicts. These applications constitute interesting further work.

We give a last potential application example of our theory: routing in a communication network. In the same way that several agents can perform in parallel to

achieve different tasks, in a communication network a message can be broadcast by generating multiple copies of it and sending these copies in parallel, along the stepwise DP-optimal paths. These paths are actually the stepwise DP-optimal schedulers seen as stepwise DP-optimal subnetworks. The marked states represent the agents to whom the messages are destined. The costs may be the energy consumed for each transmission between nodes. The uncontrollability of certain events may reflect the possibility of other agents changing the terminal path to certain nodes, based on their own view of the network.

Example 7.1. Consider the example of section 6 as a routing problem in a communication problem. Relation (6.1) (i.e., the solution of the stepwise DP-optimal problem) highlights the manner in which the information would need to be sent through the communication network. Given that there are exactly three *Reset* events, the sender should generate exactly three messages and send them in parallel. For each message, the sender can specify (in each header, for example) the desired route that each message should take, according to the sender's view of the network and calculations. (This routing is actually given by the corresponding stepwise DP-optimal subscheduler.) However, the uncontrollability is represented by the fact that other intermediate routing nodes may have a view of the network that is different from that of the sender, in which case the former might decide on a new route.

8. Conclusion. In this paper, we have introduced a new type of optimal control for DES. Previous work in optimal control deals with numerical performances in supervisory control theory when the goal to achieve is a unique state of interest. In contrast, our goal was to make the system evolve through a set of goals one by one, with no order necessarily specified a priori. The order in which the states are visited was part of the optimization problem since it had an influence on the cost of visiting all the goal states.

The system to be controlled is represented by an FSM with a set of multiple marked states $\mathcal{X} = (X_i)_{i\in[1,\dots,n]}$ representing the states of interest. Our aim was to have the system reach each and every one of the $(X_i)_{i\in[1,\dots,n]}$. To do so, we have introduced the notion of a scheduler. A scheduler can be thought of as a concatenation of submachines. The role of the scheduler is to make the system evolve according to one submachine at a time and account for switching between them at appropriate instants, i.e., when one of the states of interest has been reached. We have then introduced the notion of a stepwise DP-optimal scheduler of an FSM G with respect to the set $\mathcal{X}$. This particular type of scheduler is custom made given the system on which the optimization is to be run. It has the particularity of being composed of DP-optimal submachines which allow optimality from state of interest to state of interest (stepwise). Moreover, the ordering of these DP-optimal submachines allows global optimality in the sense that the total worst-case cost of visiting all the states of $\mathcal{X}$ is minimized.

We gave a necessary and sufficient condition for the existence of a stepwise DP-optimal scheduler, namely, the existence of n DP-optimal submachines between the initial state q_0 and each state of the n states of $\mathcal{X}$. This condition is not very restrictive, since if it does not hold, that means that one of the states is not reachable in a controllable manner, i.e., not surely reachable from the initial state q_0. In such a case, it is obvious that the state in question will never be reachable with a surely finite cost.

From a computational point of view, we showed that our optimal problem could be brought down to an instance of the TSP. The solution of this particular TSP

gives a direct access to both the structure of a stepwise DP-optimal scheduler and the worst-case cost for visiting all the states of interest. Considering the high computational complexity of this step, we also gave ways of taking advantage of some particular properties of the structure of a stepwise DP-optimal scheduler, leading to the reduction of the computational complexity of the corresponding TSP without loss of global optimality.

Finally, besides the possible applications briefly presented in section 7, future work will most probably extend the theory to the case of a system where the events are partially observable.

Acknowledgment. The authors wish to thank the reviewers for their relevant comments.

REFERENCES

[1] E. Brinskma, *A theory for the derivation of tests*, Protocol Specification, Testing and verification, 7 (1988), pp. 63–74.

[2] C. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1999.

[3] J. Fernandez, C. Jard, T. Jéron, and C. Viho, *Experiment in automatic generation of test suites for protocols with verification technology*, Sci. Comput. Programming, 29 (1997), pp. 123–146.

[4] J. E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, Reading, MA, 1979.

[5] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, *Planning and acting in partially observable stochastic domains*, Artificial Intelligence, 101 (1998), pp. 99–134.

[6] R. Kumar and V. Garg, *Optimal control of discrete event dynamical systems using network flow techniques*, in Proceedings of the 29th Allerton Conference on Communication, Control, and Computing, Champaign, IL, 1991, pp. 705–714.

[7] V. Kumar and L. N. Kanal, *A general branch and bound formulation for and/or graph and game tree search*, in Search in Artificial Intelligence, Springer-Verlag, New York, Berlin, 1988, pp. 91–130.

[8] E. L. Lawler, J. K. Lenstra, A. H. G. Rinooy Kan, and D. B. Shmoys, *The Traveling Salesman Problem*, John Wiley, New York, 1985.

[9] E. L. Lawler and D. E. Wood, *Branch-and-bound methods: A survey*, Oper. Res., 14 (1966), pp. 699–719.

[10] H. Marchand, O. Boivineau, and S. Lafortune, *On the Synthesis of Optimal Schedulers in Discrete Event Control Problems with Multiple Goals*, Tech. Report n° CGR-98-10, Control Group Reports, College of Engineering, University of Michigan, Ann Arbor, MI, 1998. Available via ftp. from ftp://ftp.eecs.umich.edu/techreports/systems/control_group/lafortune/.

[11] H. Marchand and M. Le Borgne, *On the optimal control of polynomial dynamical systems over z/pz*, in Proceedings of the 4th IEE International Workshop on Discrete Event Systems, Cagliari, Italy, 1998, pp. 385–390.

[12] K. Murty, *Operations Research: Deterministic Optimization Models*, Prentice-Hall, Upper Saddle River, N.J., 1995.

[13] D. J. Musliner, E. H. Durfee, and K. G. Shin, {*Circa*}: {*A*} *cooperative intelligent real time control architecture*, IEEE Trans. Systems, Man, and Cybernetics, 23 (1993), pp. 1561–1574.

[14] K. Passino and P. Antsaklis, *On the optimal control of discrete event systems*, in Proceedings of the 28th IEEE Conference on Decision and Control, Tampa, FL, 1989, pp. 2713–2718.

[15] P. J. Ramadge and W. M. Wonham, *Supervisory control of a class of discrete event processes*, SIAM J. Control Optim., 25 (1987), pp. 206–230.

[16] P. J. G. Ramadge and W. M. Wonham, *The control of discrete event systems*, Proceedings of the IEEE, 77 (1989), pp. 81–98.

[17] A. Rouger and M. Phalippou, *Test cases generation from formal specifications*, in Proceedings of the ISS'92, Yokohama, Japan, 1992, p. C10.2.

[18] S. Russel and P. Norvig, *Artificial Intelligence: A Modern Approach*, Prentice-Hall, Upper Saddle River, NJ, 1995.

[19] R. Sengupta and S. Lafortune, *A Deterministic Optimal Control Theory for Discrete Event Systems: Computational Results*, Tech. Report n° CGR-93-16, Control Group Reports, College of Engineering, University of Michigan, Ann Arbor, MI, 1993. Available via ftp. from ftp://ftp.eecs.umich.edu/techreports/systems/control_group/lafortune/.

[20] R. Sengupta and S. Lafortune *A Deterministic Optimal Control Theory for Discrete Event Systems: Formulation and Existence Theory*, Tech. Report n° CGR-93-7, Control Group Reports, College of Engineering, University of Michigan, Ann Arbor, MI, 1993. Available via ftp. from ftp://ftp.eecs.umich.edu/techreports/systems/control_group/lafortune/.

[21] R. Sengupta and S. Lafortune, *An optimal control theory for discrete event systems*, SIAM J. Control Optim., 36 (1998), pp. 488–541.

[22] G. Shen and P. Caines, *Control consistency and hierarchically accelerated dynamic programming*, in Proceedings of the 37th IEEE Conference on Decision and Control, Tampa, FL, 1998, pp. 1686–1691.

[23] G. Shen, P. Caines, and P. Hubbard, *Control Consistency and Hierarchically Accelerated Dynamic Programming*, Tech. report, Department of Electrical Engieering, McGill University, Montreal, Quebec, Canada, 1997.

[24] E. Tronci, *Optimal state supervisory control*, in Proceedings of the 35th IEEE Conference on Decision and Control, Kobe, Japan, 1996, pp. 2237–2242.