Dedicated to
Professor Jakub Gutenbaum
on his 70th birthday

Control and Cybernetics

vol. **29** (2000) No. 1

Dynamic control of a class of discrete event systems using a state reconstruction algorithm[1]

by

Francesco Martinelli*, Salvatore Nicosia* and Paolo Valigi**

* Dipartimento di Informatica, Sistemi e Produzione
Università di Roma "Tor Vergata"
Via di Tor Vergata, 00133 Roma, Italy

** Dipartimento di Ingegneria Elettronica e dell'Informazione
Università di Perugia
Via G. Duranti, 93, 06125 Perugia, Italy

e-mail: {martinelli,nicosia}@disp.uniroma2.it
e-mail: valigi@diei.unipg.it

Abstract: The problem of dynamic control of Discrete Event Dynamic Systems (DEDS) is addressed in this paper as a dynamic optimization problem: some resources must be allocated to the system in order to optimize a performance function which is assumed time-varying. The control scheme exploits a state reconstruction algorithm to compute an estimate of the performance for perturbed sample paths.

The algorithm is based on the use of data extracted from the observation of the system and allows to accurately reconstruct its state behavior, for resource allocations different from the nominal one.

The proposed control scheme is then used for dynamic allocation of buffer capacities in manufacturing systems, such as Kanban systems. A parallel implementation of the whole algorithm is also mentioned.

Keywords: optimization, ordinal optimization, Kanban systems, sample path analysis

[1]This work has been supported by ENEA (MURST program) and University of Tor Vergata (ex 60%) funds.

1. Introduction

DEDS are important models of many man-made systems, for which on-line or dynamic control is now emerging as a key problem.

Several techniques based on the observed sample path have been proposed, aimed at estimating the behavior of a DEDS under perturbed values of some of its parameters (Glassermann, 1990; Ho and Cao, 1991).

Although the main motivations for the introduction of these techniques was reduction of simulation effort, they are also particularly useful in implementing on-line control schemes. As a matter of fact, in this case it is not possible to run several instances of the same system, with different values of the parameters, in order to assess its behavior. Such a type of on-line comparison instead, is made possible by sample path techniques, and by the one proposed here.

Classical perturbation analysis techniques (Glassermann, 1990; Ho and Cao, 1991), and the modified rules proposed in Liberatore, Nicosia and Valigi (1997) for the case of "weakly adjacent" events, attempt to estimate perturbation in the time of occurrence of an event by comparing nominal and perturbed timings, without consideration of the state evolution. The main difficulty in such a "timing comparison" approach is that several past and future events are required, thus making such a computation too complex, and, more important, non causal. The problem is usually solved by assuming that only sufficiently close events may change order due to parameter perturbation. This allows to obtain computation algorithms, which are approximate in nature, and whose accuracy is reduced when larger parameter variations are considered.

To overcome this problem, the state reconstruction scheme from Martinelli, Nicosia and Valigi (1997b) will be used here. A similar reconstruction approach is that of Rapid Learning (Cassandras, 1993b), which includes techniques like Augmented System Analysis (Cassandras and Strickland, 1989), Standard Clock (Vakili, 1991), and the recently proposed Time Warping Algorithm (TWA) (Cassandras and Panayiotou, 1996).

The main objective of this paper is the description, analysis and test of an algorithm for the dynamic control of DEDS, which uses as ingredient the sample path analysis approach of Martinelli, Nicosia and Valigi (1997b).

The control problem will be posed as an optimization problem: some resources must be allocated in order to optimize a performance function defined on the system. Changes in the stochastic description of the system are possible at any time: the time and the type of these changes are not known by the controller which should dynamically modify the resource allocation in reaction to these changes. The framework is very general: the proposed control scheme can be applied with any stochastic description. To operate the dynamic resource allocation, it is necessary to periodically take an estimate of the performance function corresponding to a given set of possible allocations, based on the measurements taken on the real system. In the more general case in which some measurements are not available, the inversion approach proposed in Park and

Chong (1995) could be used.

To compute these estimates we use a state reconstruction algorithm, the Marking Algorithm, which is also presented in this paper and has similarities with TWA. Some characteristics, however, differentiate the two algorithms. A minor one concerns implementation. The reconstruction of perturbed sample paths in the Marking Algorithm is triggered periodically, at the end of every control interval: hence the decision as to which parameters to test can be taken at the end of the control interval, e.g., on the basis of paths already reconstructed on the same interval. TWA resumes the reconstruction every time there exists sufficient information to determine the next event. This allows to save time and memory space, but makes difficult the choice at the end of the control interval whose perturbed sample paths we may want to reconstruct.

Moreover, TWA is designed for DEDS whose randomness is only involved in event lifetimes, while state transitions are completely determined. On the contrary, the Marking Algorithm can be applied also to DEDS with random state transitions. In particular, when dealing with queueing systems, we will consider also random scheduling and routing policies and proper sequences will be introduced to store the random choices performed by the stochastic scheduling and routing policies.

A complete analytical study is offered for the Marking Algorithm to clarify that the state sequence reconstructed by this algorithm, under proper assumptions, is the longest that it is possible to reconstruct using a given set of information.

The control algorithm proposed in this paper is an extension of the basic idea presented in Liberatore, Nicosia and Valigi (1997). Here, the control problem for time-varying systems has been formally stated, and analytical results for the proposed control algorithm are given for a class of systems. Simulation results are reported for more general systems. A similar idea is also reported in Pepyne and Cassandras (1997).

In the sequel of the paper, presentation will be focused on a special class of DEDS, in particular those systems which can be modeled by means of queueing networks. As it will be shown in the paper, the extension to general DEDS is straightforward.

The paper is organized as follows: in Section 2 notation is introduced; in Section 3 the state reconstruction problem is stated and solved; in Section 4 the control scheme based on this state reconstruction algorithm is presented; the application of such a methodology to on-line control of manufacturing systems has been considered and tested by means of simulations in Section 5, together with a parallel implementation. Section 6 gives some remarks on the extension of the method to general DEDS. Section 7 contains conclusions.

2. Notation and system dynamics

In this paper we will consider a dynamic control problem for the class of systems that can be modeled by means of time-varying queueing networks with: (a) general service time, (b) general routing and scheduling policy at each node, (c) finite buffer capacity, (d) multi-class, (e) non-preemptive service, (f) infinite arrival rate sources and infinite capacity sinks. Assumption (f) is considered in order to guarantee that machines immediately downstream of sources are never starved and machines immediately upstream of sinks are never blocked. This is a classical assumption, and does not lead to loss of generality. As a matter of fact, any finite arrival source can be modelled as the series connection of an infinite arrival source followed by a node with a properly selected service time probability distribution. A similar reasoning applies to infinite capacity sinks.

Each node of the network comprises a multi-class server and a dedicated input queue is associated with each class. The set of nodes is denoted by $\mathcal{N}$ and comprises N_s elements. The set of classes serviced by node i will be denoted by $\mathcal{C}_i$.

The dynamics of a queueing network is driven by random processes, defined on a common probability space $(\Omega, \mathcal{B}, P)$, characterizing service durations, scheduling and routing policies. With each node i, implementing a random scheduling policy, is associated a random sequence $\{u_S(i)\}_{k_i}$, whose k_i-th entry is the class of the k_i-th customer serviced by server i. Similarly, with each node i implementing a random routing policy is associated a random sequence $\{u_R(i,\alpha)\}_{k_{i,\alpha}}$ for each class $\alpha \in \mathcal{C}_i$, whose $k_{i,\alpha}$-th element is the destination queue for the $k_{i,\alpha}$-th class α customer serviced by node i.

Finally, with each node i is associated a random sequence $\{u_D(i,\alpha)\}_{k_{i,\alpha}}$ for each class $\alpha \in \mathcal{C}_i$, whose $k_{i,\alpha}$-th element is the service duration of the $k_{i,\alpha}$-th class α customer serviced by node i.

Denote by $u_S(\cdot)$, $u_R(\cdot)$, and $u_D(\cdot)$ the realizations of the random sequences $\{u_S(i)\}_{k_i}$, $\{u_R(i,\alpha)\}_{k_{i,\alpha}}$, and $\{u_D(i,\alpha)\}_{k_{i,\alpha}}$, respectively, and let $u(\cdot) := \{u_S(\cdot), u_R(\cdot), u_D(\cdot)\}$; in the following $u(\cdot)$ will be referred to as *input sequence*, since it drives the network evolution. Finally, let $\mathcal{U}$ denote the set of all the *admissible input sequences*, that is, the set of all the deterministic sequences $u(\cdot) = \{u_S(\cdot), u_R(\cdot), u_D(\cdot)\}$ that are realizations of the corresponding stochastic sequences.

We will assume that in the general case the statistical properties of the random sequences characterizing service durations, scheduling and routing choices may change at some given times. In particular, we will suppose that there is a set $\{\gamma_1, \gamma_2, \ldots\}$, called *switching set*, possibly empty, of time instants, $\gamma_{i+1} > \gamma_i$, called *switching times*, such that the queueing network has the same stochastic description in every interval $[\gamma_l, \gamma_{l+1})$, called *steady interval*, but the stochastic properties could change at every switching time.

Let x be the *queueing state* (containing information about buffer contents and server state), and let $\mathcal{X}$ denote the queueing state space.

The completion time of the customer currently under service at node i, or the completion time of the last customer serviced by node i, if node i is currently starved or blocked, will be denoted by τ_i, and $\tau := (\tau_1, \tau_2, \ldots, \tau_{N_S})^T$ will denote the vector of all these completion times, where 'T' denotes transposition.

The complete state of a queueing network is given by the vector $z := (x^T, \tau^T)^T$, and will be referred to as the network *extended state*, with *extended state space* $\mathcal{Z}$. Then, the evolution of a queueing network can be expressed, based on the GSMP formalism (Cassandras, 1993a), as:

$$z(\cdot) = \Phi(z(0), u(\cdot)). \tag{1}$$

Function Φ will be referred to as the *extended state map*, while the sequence $z(\cdot)$ will be referred to as the *extended state sequence*. It is stressed that equation (1) is completely deterministic, i.e., given the initial extended state $z(0)$ and the input sequence $u(\cdot)$, its solution is completely determined. This means that once the (infinite length) input sequence has been recorded, the extended state sequence can be reconstructed; so eq. (1) can be seen both as a network description and as an ideal algorithm for reconstructing the evolution of the queueing network once the input sequence and the initial state have been recorded. This is the basic idea of the reconstruction algorithm which will be presented in the following (see Martinelli, Nicosia and Valigi, 1997a, for more details).

The behavior of a queueing network depends on a number of network parameters $\theta \in \Theta$, whose value may be changed during network operation to control network performance. It is assumed that the *parameter space* Θ has N elements and does not contain structural parameters, such as, e.g., the number of nodes or classes in the network. For simplicity, we will also assume in the following that the values of these parameters do not affect the input sequences. If this is not the case the problem becomes much more involved and is not treated in this paper: in this case even a stationary system could become time varying when controlled. Then, to reflect such a dependence on parameter θ, equation (1) will be written as:

$$z(\cdot) = \Phi(z(0), u(\cdot), \theta). \tag{2}$$

For queueing networks, the values the state may assume are not independent of network parameters: e.g., if the entries of vector θ comprise the buffer capacities, then these entries are also the maximum values for buffers content.

Let $\mathcal{X}(\theta)$ denote the *admissible state space under parameter* θ, that is, the set of all the values the queueing state vector may assume, for the value θ of the network parameter; then $\mathcal{X} = \cup_{\theta \in \Theta} \mathcal{X}(\theta)$. A given queueing state x is an *admissible queueing state under parameter* θ if $x \in \mathcal{X}(\theta)$.

3. The state reconstruction problem

The solution to the state reconstruction problem considered in this paper is based on the assumption that it is possible to directly measure the input

sequence over a finite time interval and the initial extended state. The observability issue, in the sense of Park and Chong (1995), is not considered here, and direct availability of the required data is assumed. For more general cases, the approach in Park and Chong (1995) can be considered.

Apart from the termination issue, explored in detail in this section, the Marking Algorithm proposed here to solve the reconstruction problem, consists of the execution of the system evolution equations using the measured input sequence and initial state. Observe that for the general framework considered in this paper, where the stochastic process characterizing the system is unknown, this kind of approach is classical.

Let $\bar{z}_0$ be the measured or *observed initial extended state*, $\bar{u}(\cdot)$ be the measured or *observed input sequence*, and $\bar{z}(\cdot) = \Phi(\bar{z}_0, \bar{u}(\cdot), \bar{\theta})$ the measured (or observed) extended state sequence, corresponding to the nominal network parameter vector $\bar{\theta}$. To reconstruct the extended state sequence for perturbed values of the network parameter, the system dynamics (2) could be used. Notice that the two sequences $\bar{u}(\cdot)$ and $\bar{z}(\cdot)$ have infinite length, hence any associated reconstruction problem can be solved only at a conceptual level.

The extended state sequence we would have observed if the network parameter was $\hat{\theta}$ in place of $\bar{\theta}$ can be reconstructed using eq. (2) if the observed initial extended state $\bar{z}_0 = ((\bar{x}_0)^T \ (\bar{\tau}_0)^T)^T$ is such that $\bar{x}_0 \in \mathcal{X}(\hat{\theta})$. Formally, we can write:

$$\hat{z}(\cdot) = \Phi(\bar{z}_0, \bar{u}(\cdot), \hat{\theta}). \tag{3}$$

In the Time Local Control Algorithm we present next, and in any other real control problem, we are interested in reconstruction problems only using finite input sequences. This is particularly true here, where attention is on non stationary systems.

Given a finite input sequence, qualitatively the problem is that of finding the longest possible extended state sequence which can be generated by such an input sequence (the concept of length of a sequence will be better specified in the following). To formally state this problem, we need notation given below.

Let $\tilde{u}|_k(\cdot)$ denote a finite input sequence measured by observing the system for a finite time. Here k is the counter vector, that is – the vector containing the lengths of all the sequences in $\tilde{u}|_k(\cdot)$. The sequence $\tilde{u}|_k(\cdot)$ will be called a *k-length finite subsequence* (briefly, *k-subsequence*) since it can be considered as a subsequence of the infinite sequence $\bar{u}(\cdot)$, which would have been measured if the system were observed for infinite time. Given $\tilde{u}|_k(\cdot)$, let $\mathcal{U}(\tilde{u}|_k)$ denote the set of all the admissible infinite sequences having $\tilde{u}|_k(\cdot)$ as the common k-length finite initial subsequence:

$$\mathcal{U}(\tilde{u}|_k) := \{u(\cdot) \in \mathcal{U} : u|_k(\cdot) = \tilde{u}|_k(\cdot).\} \tag{4}$$

Notice that, in general, for a given $\tilde{u}|_k$, the set $\mathcal{U}(\tilde{u}|_k)$ has an infinite number of elements. Of course, the infinite sequence we would have measured when observing the system for infinite time belongs to such a set.

Let a k-subsequence $\tilde{u}|_k(\cdot)$ be given, together with a network parameter vector $\tilde{\theta}$ and an initial extended state $\tilde{z}_0 = ((\tilde{x}_0)^T\ (\tilde{\tau}_0)^T)^T$ such that $\tilde{x}_0 \in \mathcal{X}(\tilde{\theta})$. Then, the set $\mathcal{S}(\tilde{z}_0, \tilde{u}|_k(\cdot), \tilde{\theta})$ can be introduced, comprising all the infinite extended state sequences that can be generated by (2) starting from the initial value $\tilde{z}_0$ under all the admissible infinite input sequences having $\tilde{u}|_k(\cdot)$ as the common k-length finite initial subsequence. Formally:

$$\mathcal{S}(\tilde{z}_0, \tilde{u}|_k(\cdot), \tilde{\theta}) := \{z(\cdot) : z(\cdot) = \Phi(\tilde{z}_0, u(\cdot), \tilde{\theta}),\ u(\cdot) \in \mathcal{U}(\tilde{u}|_k)\}. \tag{5}$$

Given an extended state sequence $z(\cdot)$, the sub-sequence obtained by taking the first $\tilde{k}$ terms (the $\tilde{k}$-th term is the value that the extended state assumes when the counter vector is equal to $\tilde{k}$) will be denoted by $z|_{\tilde{k}}(\cdot)$ and will be referred to as the $\tilde{k}$-subsequence of the infinite sequence $z(\cdot)$. Given the set $\mathcal{S}(\tilde{z}_0, \tilde{u}|_k(\cdot), \tilde{\theta})$, a subsequence $z^*|_n(\cdot)$ is a common n-subsequence of $\mathcal{S}$ $(\tilde{z}_0,\ \tilde{u}|_k(\cdot),\ \tilde{\theta})$ if $z|_n(\cdot) = z^*|_n(\cdot)$ for all $z(\cdot) \in \mathcal{S}(\tilde{z}_0, \tilde{u}|_k(\cdot), \tilde{\theta})$. Then, the state reconstruction problem for finite input sequences can be formally stated as follows:

PROBLEM 1 *For the queueing network Σ, with observed initial extended state $\bar{z}_0 = ((\bar{x}_0)^T\ (\bar{\tau}_0)^T)^T$, and observed k-length input sequence $\bar{u}|_k(\cdot)$, given a perturbed value $\theta = \hat{\theta}$ of the network parameter vector such that $\bar{x}_0 \in \mathcal{X}(\hat{\theta})$, find the longest common finite subsequence of $\mathcal{S}(\bar{z}_0, \bar{u}|_k(\cdot), \hat{\theta})$.*

A slightly different notion of the longest finite sequence could be introduced considering each single node instead of the whole system. Problem 1 will be solved under the following assumption, which is not too restrictive.

ASSUMPTION 1 *The probability distributions of all the random variables determining service duration σ, for each server, are such that, for all $\epsilon > 0$, $Prob\{0 < \sigma \leq \epsilon\} > 0$.*

Problem 1 is initially solved under a simplifying assumption.

ASSUMPTION 2 *The queueing network Σ only comprises servers with deterministic scheduling policy.*

Under assumptions 1 and 2, Problem 1 can be solved by the following algorithm, as shown in the subsequent Theorem 1.

ALGORITHM 1 *Deterministic Marking Algorithm*

- Given the k-length finite observed input subsequence $\bar{u}|_k(\cdot)$, let $\bar{u}|_\infty(\cdot)$ be the infinite sequence obtained from $\bar{u}|_k(\cdot)$ by adding "mark" symbols after the last element of each sequence comprising $\bar{u}|_k(\cdot)$; e.g. let the mark symbol be $+\infty$.
- Apply the algorithm given by $\hat{z}(\cdot) = \Phi(\bar{z}_0, \bar{u}|_\infty(\cdot), \hat{\theta})$ and terminate the extended state reconstruction as soon as a server which needs a new service duration extracts a mark symbol.

THEOREM 1 *If Assumptions 1 and 2 hold for system Σ, then Problem 1 is solved by the Deterministic Marking Algorithm.*

Proof. Let $\hat{z}|_n(\cdot)$ be the finite extended state n-subsequence obtained by the Deterministic Marking Algorithm, where n is the number of its terms. The proof is completed if the two following conjectures are correct:

- $\hat{z}|_n(\cdot)$ is a common n-subsequence of $\mathcal{S}(\bar{z}_0, \bar{u}|_k(\cdot), \hat{\theta})$;
- a common m-subsequence of $\mathcal{S}(\bar{z}_0, \bar{u}|_k(\cdot), \hat{\theta})$, with $m > n$, does not exist.

The proof of the first conjecture is straightforward: the Deterministic Marking Algorithm performs only deterministic operations over a given finite input subsequence, common to all the elements of the set $\mathcal{U}(\bar{u}|_k)$.

To prove the second conjecture, let $z^A|_a(\cdot)$ be a common a-subsequence of $\mathcal{S}(\bar{z}_0, \bar{u}|_k(\cdot), \hat{\theta})$ and suppose by contradiction that $a > n$. Take the first $n+1 \leq a$ terms of $z^A|_a(\cdot)$, and denote this subsequence with $z^A|_{n+1}(\cdot)$. Obviously $z^A|_{n+1}(\cdot)$ is a common $(n+1)$-subsequence of $\mathcal{S}(\bar{z}_0, \bar{u}|_k(\cdot), \hat{\theta})$ and $z^A|_n(\cdot) = \hat{z}|_n(\cdot)$. So $z^A|_{n+1}(\cdot)$ differs from $\hat{z}|_n(\cdot)$ only by having one more term, the $(n+1)$-th one.

As shown below, if such a $z^A|_a(\cdot)$ exists, it is possible to construct an $(n+1)$-subsequence of a sequence in $\mathcal{S}(\bar{z}_0, \bar{u}|_k(\cdot), \hat{\theta})$ that differs from $z^A|_{n+1}(\cdot)$ by the last term: this is in contradiction with the fact that $z^A|_{n+1}(\cdot)$ is a common $(n+1)$-subsequence of $\mathcal{S}(\bar{z}_0, \bar{u}|_k(\cdot), \hat{\theta})$, hence with the fact that $z^A|_a(\cdot)$ is a common a-subsequence of $\mathcal{S}(\bar{z}_0, \bar{u}|_k(\cdot), \hat{\theta})$.

To construct this subsequence proceed as follows. Let $\delta > 0$ be the time interval between the two final transitions of the extended state sequence $z^A|_{n+1}(\cdot)$ and choose an $\epsilon \in \mathbb{R}$, $0 < \epsilon < \delta$. Replace the mark symbol of the sequence $u|_\infty(\cdot)$ that has determined the termination of the Deterministic Marking Algorithm while computing $z^A|_n(\cdot)$, with a service duration equal to $\eta \leq \epsilon$, which is an admissible service duration, in view of Assumption 1. Then consider the extended state $(n+1)$-subsequence obtained by considering the first $(n+1)$ iterations of the Deterministic Marking Algorithm, applied to such a modified input sequence. This subsequence contains $n+1$ elements and differs from $z^A|_{n+1}(\cdot)$, as was to be proved. ■

The extension to the case of servers with random scheduling is not difficult and is given in the following Theorem and Algorithm. To simplify notation, assembly nodes are not allowed. In the following algorithm, $u_{D,i,\alpha}(k_{i,\alpha})$ denotes the observed service duration of the $k_{i,\alpha}$-th customer of class α serviced at node i, $N_{C,i}$ denotes the number of classes that node i can provide service to, $u_{S,i}(k_i)$ denotes the class of the k_i-th customer serviced at node i, and $x_{L,i,\alpha}$ denotes the state entry corresponding to the current length of queue α in node i.

ALGORITHM 2 *Marking Algorithm*

- Given $\bar{u}|_k(\cdot)$, let $\bar{u}|_\infty(\cdot)$ be the infinite sequence obtained from $\bar{u}|_k(\cdot)$ by adding an infinite number of mark symbols after the last terms of all the sequences in $\bar{u}|_k(\cdot)$.
- Associate with each server i implementing a random scheduling policy a set F_i, initially empty, and $N_{C,i}$ flags $f(i,\alpha)$, $\alpha \in \mathcal{C}_i$, initially set to zero.
- Apply the algorithm given by $\hat{z}(\cdot) = \Phi(\bar{z}_0, \bar{u}|_\infty(\cdot), \hat{\theta})$, and, in addition, at each state transition update flags $f(i,\alpha)$, $\alpha \in \mathcal{C}_i$, and sets F_i, for all nodes i implementing a random scheduling policy, according to the following rules:

$$f(i,\alpha) := \begin{cases} 1 & \text{if } u_{D,i,\alpha}(k_{i,\alpha}) = \text{mark symbol} \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

$$F_i = \{\alpha \in \mathcal{C}_i : f(i,\alpha) = 1 \text{ and } x_{L,i,\alpha} > 0\}, \tag{7}$$

and mark all the servers for which the three following conditions are satisfied all together:

1. F_i is not empty;
2. server i has completed a service at the current transition, or is starved;
3. $u_{S,i}(k_i) \in F_i$ or $u_{S,i}(k_i) =$ mark symbol.

- The extended state reconstruction terminates as soon as a server with deterministic scheduling which needs a new service duration value extracts a mark symbol or a server implementing a random scheduling policy is marked.

Theorem 2, which can be proved similarly to Theorem 1, summarizes the results of the solution to Problem 1, in the general case in which both deterministic and random scheduling policies are allowed.

THEOREM 2 *Assume the queueing network Σ does not contain assembly nodes and that Assumption 1 holds, then Problem 1 is solved by the Marking Algorithm.*

Proof. Like for Theorem 1, assume $\hat{z}|_n(\cdot)$ be the finite extended state n-subsequence obtained by the Marking Algorithm and assume by contradiction that it is possible to determine a common a-subsequence $z^A|_a(\cdot)$ of a sequence in $\mathcal{S}(\bar{z}_0, \bar{u}|_k(\cdot), \hat{\theta})$ with $a > n$. Also in this case we show that it is possible to construct an $(n+1)$-subsequence of a sequence $\mathcal{S}(\bar{z}_0, \bar{u}|_k(\cdot), \hat{\theta})$ that differs from $z^A|_{n+1}(\cdot)$ for the last term.

Let $\delta > 0$ be the time interval between the two final transitions of the extended state sequence $z^A|_{n+1}(\cdot)$. If the Marking Algorithm was terminated by a deterministic scheduling node, it is possible to repeat the procedure reported in the proof of Theorem 1. Otherwise, if i is the marked node which terminated the algorithm, select an element from F_i and set $u_{D,i,\alpha}(k_{i,\alpha}) = \eta < \delta$. If just one of the conditions to mark the node was not met, it was not possible to apply the above procedure. The reason for which, in the perturbed path, $u_{S,i}(\cdot)$

can be not all used while $u_{D,i,\alpha}(k_{i,\alpha})$ =mark symbol, is that node i may have worked more parts of type α than in the nominal path. ■

Before concluding the section, we want to give some indications on the complexity of the Marking Algorithm. The computational complexity is proportional to the number of the events recorded in the input sequence and to the complexity of the update of the state when an event occurs. The quantity of memory required by the algorithm must be enough to store the input sequence.

4. Dynamic control of time-varying queueing networks

The dynamic control of a time-varying queueing network will be formulated as a dynamic optimization problem: a performance index $\mathcal{I}$, which is affected by the parameter vector θ, is defined on the network and the control objective is that of on-line choosing the value of the parameter vector which optimizes such a performance index.

Suppose for now that the switching set defined in Section 2 is empty (stationary scenario). A sample of the network performance taken in $[t, t+T]$ will be denoted by $L(\theta, \omega, t, t+T)$, where ω is a particular realization of the stochastic process characterizing the input sequence. The following ergodicity assumption allows to define a performance index $\mathcal{I}(\theta)$:

ASSUMPTION 3 *In the stationary scenario, the queueing network is such that* $\lim_{T\to\infty} L(\theta, \omega, t, t+T)$ *is well defined for all time* t *and* $\theta \in \Theta$, *and its value is the same for all* ω *and* t, *except possibly for* ω *in a* 0*-measure set (almost sure convergence of the limit).*

If Assumption 3 holds, we will assume as performance index the following steady state value

$$\mathcal{I}(\theta) = \lim_{T\to\infty} L(\theta, \omega, t, t+T).$$

In the non-stationary scenario define

$$\mathcal{I}(\theta, t) = \widehat{\lim}_{T\to\infty} L(\theta, \omega, t, t+T)$$

where the notation $\widehat{\lim}$ is used to specify that the limit is computed as if the stochastic process characterizing the queueing network would maintain after t the same description it has at time t and that such a fictitious stationary system satisfies Assumption 3. The dynamic optimization problem is that of finding:

$$\theta^*(t) = arg \max_{\theta\in\Theta} \mathcal{I}(\theta, t).$$

In the stationary case, if we observe the system for an infinite time, we are eventually able to choose θ^* which solves our problem. A control algorithm, however, has to take a decision in a finite time, as in the following algorithm,

which is the basis for the control scheme proposed next. The basic idea is that of applying a control action (i.e., a parameter modification) every T_c time units, at the end of a control interval $[t_k, t_{k+1}]$, where $t_k = kT_c$.

ALGORITHM 3 *Basic Control Algorithm.*

- *Initialization: Choose a duration T_c for the control intervals, an initial value for the parameter vector θ_0, and set $k = 0$ and $t_0 = 0$.*
- *Iteration k:*
  1. *by applying the marking algorithm, reconstruct the extended state sequence $\hat{z}_\theta(\cdot)$ using the input sequence recorded up to t_{k+1}, for all θ in Θ;*
  2. *based on the $\hat{z}_\theta(\cdot)$, compute $L(\theta, \omega, 0, t_{k+1})$, $\forall\ \theta \in \Theta$.*
  3. *choose as new parameter*

$$\theta_{k+1} = \bar{\theta} = arg \max_{\theta \in \Theta} L(\theta, \omega, 0, t_{k+1}).$$

The reconstruction at Step 1 can be implemented in a more efficient manner. At the end of each iteration k, we store for all $\theta \in \Theta$ the extended state reached by the queueing network under θ and the segment of the input sequences recorded in $[t_k, t_k + T_c]$ which may have been not completely used in the reconstruction. At iteration $k+1$ we extend these input sequences with the values observed in $[t_{k+1}, t_{k+1} + T_c]$. For all θ we resume the extended state reached at the end of iteration k and continue the reconstruction by means of marking algorithm. Such a procedure is repeated again and again.

THEOREM 3 *In the stationary scenario, under Assumption 3, the Basic Control Algorithm asymptotically gives with probability one the optimal parameter vector value θ^*.*

Proof. Since, as assumed above, the control does not modify input sequences, the result directly follows from Assumption 3. ■

The Basic Control Algorithm cannot be successfully applied in the non stationary scenario. The following modified version, designed for non stationary problems, exploits the characteristics of ordinal comparison: if it is applied to a stationary system, the order among different parameter vector values θ based on estimates $L(\theta, \omega, t, t+T)$ approaches very fast (in some cases exponentially with T) the true order of parameter vector values, i.e. the order based on $\mathcal{I}(\theta)$, much faster than the rate the variance of $L(\theta, \omega, t, t+T)$ approaches 0, which is of order $1/T$ (Dai and Chen, 1997).

In the non-stationary case observe that $L(\theta, \omega, t, t+T)$ can be considered an estimate of $\mathcal{I}(\theta, t)$ only if the interval $[t, t+T]$ is all contained in a steady interval (i.e. there exists l: $\gamma_l \leq t < t+T < \gamma_{l+1}$).

The following algorithm performs the reconstruction only using data from the current control interval (*time local feature*). For this reason, reconstruction

for all $\theta \in \Theta$, as in the Basic Control Algorithm, is no more necessary. In the basic version the reconstruction should be performed for all $\theta \in \Theta$: stopping the reconstruction for a θ at some iteration would exclude this vector value in all successive iterations. On the contrary, using the Time Local version, it is possible to perform the reconstruction for a subset $\Theta_k \subseteq \Theta$, possibly different from a control interval to another, without losing any θ.

ALGORITHM 4 *Time Local Control Algorithm.*

- *Initialization: Choose a duration T_c for the control intervals, an initial value for the parameter vector θ_0 and set $k = 0$ and $t_0 = 0$.*
- *Iteration k:*

  1. *using the marking algorithm and only data measured in the control interval $[t_k, t_{k+1}]$, reconstruct the extended state sequence $\hat{z}_\theta(\cdot)$, for all $\theta \in \Theta_k$, $\Theta_k \subseteq \Theta$ being a subset to be defined;*
  2. *based on $\hat{z}_\theta(\cdot)$, compute $L(\theta, \omega, t_k, t_{k+1})$, for all $\theta \in \Theta_k$.*
  3. *choose as new parameter*

$$\theta_{k+1} = \bar{\theta} = arg \max_{\theta \in \Theta_k} L(\theta, \omega, t_k, t_{k+1}).$$

The implementation of Step 1 must be explained with more detail. Suppose m control actions have already been implemented and let θ_m be the current value of the parameter vector. We start a new control interval $[t_m, t_{m+1}]$ which will end with a control action (Step 3) at time t_{m+1}. State reconstruction is carried out only using data observed in the $(m+1)$-th control interval, i.e. the interval $[t_m, t_{m+1}]$. This allows to perform the reconstruction for just a set $\Theta_m \subseteq \Theta$, which implies a computation reduction and also a greater simplicity in handling input sequences. Notice that the Time Local feature is necessary in the non stationary scenario, where data from the past may lose significance.

However, the above control scheme, even if applied in the stationary scenario, does not converge to the optimal parameter vector value θ^*. In the stationary scenario we can just set T_c large enough to be confident with a high probability of being in a set of sub-optimal parameter vector values. To formalize this, we consider again the stationary scenario and introduce the sub-optimal set $\Theta(p)$ of parameter vector values $\theta \in \Theta$ with the first p largest values of $\mathcal{I}(\theta)$, i.e., if we order parameter vector values $\theta \in \Theta$ from the best to the worst one, $\Theta(p)$ contains the first p elements.

The following theorem characterizes the steady state probabilities and, consequently, the behavior of the algorithm, in the stationary scenario.

THEOREM 4 *In the stationary scenario, for a queueing network which satisfies Assumption 3, and given a sub-optimal set $\Theta(p)$, for all $\epsilon > 0$ small as desired there exists a T large enough that the steady state probability distribution of the ergodic Markov Chain generated by the Time Local Control Algorithm with*

$\Theta_m = \Theta$ for all m and $T_c \geq T$ is such that the parameter vector value θ_m selected by the algorithm at iteration m belongs to $\Theta(p)$ with probability larger than $1-\epsilon$, that is $Prob\{\theta_m \in \Theta(p)\} > 1-\epsilon$ for all $m > 0$.

Proof. We use the following indicator process, proposed in Dai (1996):

$$I_{p,N}(t,t+T) = \begin{cases} 1 & \text{if} \max_{\sigma\in\Theta(p)} L(\sigma,\omega,t,t+T) \geq \\ & \quad \max_{\theta\in\Theta\setminus\Theta(p)} L(\theta,\omega,t,t+T) \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

Observe that, if $\Theta_m = \Theta$ for all m, $t = t_k$ and $T = T_c$, $Prob\{I_{p,N}(t,t+T) = 1\}$ is the probability that at iteration k of the Time Local Control Algorithm we select a θ in the sub-optimal set $\Theta(p)$.

From Dai (1996), we know that $Prob\{I_{p,N}(t,t+T) = 1\}$ is a nondecreasing function of p and that, under Assumption 3, $\lim_{T\to\infty} Prob\{I_{p,N}(t,t+T) = 1\} = 1$ (stationary scenario). This means that for all $\epsilon > 0$ small as desired there exists a T such that $Prob\{I_{p,N}(t,t+T) = 1\} > 1-\epsilon$. In addition, observe that in the stationary scenario the Time Local Control Algorithm with $\Theta_m = \Theta\ \forall m$ behaves like a homogeneous Markov chain whose states are all the parameter vector values. Such a Markov chain reaches the steady state probability distribution in just one iteration and is characterized by a unique recurrence class. If error on estimates is small enough, bad parameter vectors values are associated with transient states. So, the Markov chain is ergodic. ■

Theorem 4 also implies that at steady state, considering a time interval long enough, the Time Local Control Algorithm spends in $\Theta(p)$ a fraction $1-\epsilon$ of such a time interval.

However, the choice of T_c is not straightforward. We have to take into account the estimate accuracy and, in time varying applications, the duration of steady intervals.

Besides the choice of T_c, to apply the marking algorithm in Step 1 of the control algorithm we need also to specify, for all $\theta \in \Theta_m$, an initial state which is admissible under the parameter vector θ for which the reconstruction is performed.

Notice that the measured initial state could be not admissible under all $\theta \in \Theta_m$. Now, the limit in Assumption 3 does not depend on the initial state even if, on the short term, the performance sample $L(\theta,\omega,t,t+T)$ could be affected by the initial state. An initial state admissible under all $\theta \in \Theta_m$ is the state of *empty network* and it has been used in our implementation. Different policies for the selection of the initial state have been tested by means of simulations without showing better results.

We specified that at iteration m, the state reconstruction could be performed just for a set $\Theta_m \subseteq \Theta$. The two following choices have been considered here: a "global" control scheme, where $\Theta_m = \Theta$ for all m, and a "local" control scheme, where Θ_m comprises all θ in a small "neighbor" of θ_m. The latter choice reduces

computation but could introduce in the Markov Chain describing the algorithm one (or more) recurrence class associated with a local optimum different from the recurrent class of the global optimum.

Different Θ_m could be designed if we have some extra information on the queueing network to restrict the search to a subset of Θ. Observe that some rules have to be defined to overcome the case when a $\theta \in \Theta_m$, which makes the current network extended state not admissible, is selected at Step 3. In the application presented in the next section we give a possible solution of this problem.

As for the complexity of the Time Local Control Algorithm, observe that at each iteration we have to apply the Marking Algorithm for all $\theta \in \Theta_m$ on a time interval of length T_c. So, the computational complexity is proportional to the cardinality of the set Θ_m and to the control interval duration T_c. The memory required by the algorithm must be sufficient to store the input sequence corresponding to an interval during T_c. So, roughly, it is like the memory required by the Marking Algorithm.

5. Dynamic control of manufacturing systems

The Time Local Control Algorithm has been applied to the problem of dynamic allocation of buffer capacities in manufacturing systems. The problem is relevant *per se*, and also as a model of Kanban controlled manufacturing systems (Di Mascolo, Frein, Dallery and David, 1991).

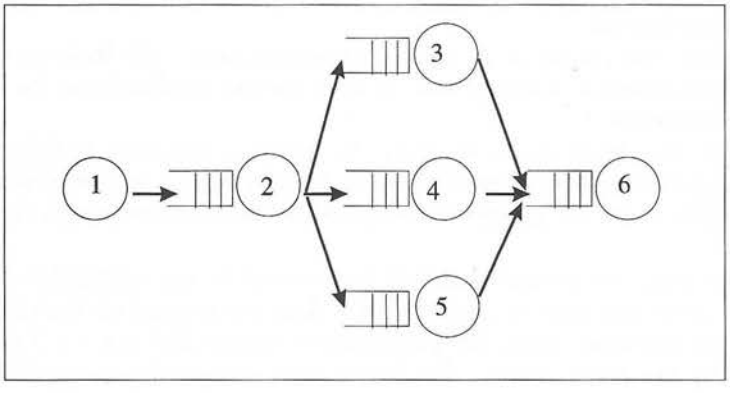

Figure 1. The manufacturing system.

The control scheme is aimed at dynamically varying the buffer capacity of some queues in the system, in reaction to unknown changes/disturbances in the production scenario, in order to optimize system performance.

The performance measure considered in this paper is the *long term system*

L1	L2	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10
169	169	168	203	168	169	168	205	169	169	168	169
168	168	287	204	169	168	169	170	170	168	204	170
288	288	167	168	288	288	288	289	4	288	169	168
204	204	288	169	204	176	204	204	289	4	288	5
170	170	169	205	176	177	4	169	205	204	170	289
176	176	175	48	170	170	5	316	168	170	49	4
289	205	203	288	5	205	176	177	6	5	205	177
205	289	3	315	177	289	167	168	5	289	48	205
167	167	176	287	4	204	3	288	177	176	176	6
5	5	4	167	289	295	49	211	49	167	316	176

Table 1. Order of the first 10 buffer allocations: comparison based on long and short run simulations.

throughput $T(H)$, computed over a large number of service customers:

$$T(H) = \frac{H}{\tau(H)}, \tag{9}$$

where $\tau(H)$ is the time at which the H-th serviced customer departs from the system.

The manufacturing system considered in our experiments is depicted in Fig. 1. It is a simple case of multi part-type manufacturing system, with the service duration of all the nodes independent of part type, nodes $3, 4$, and 5 working only one part-type each; the routing policy from node 2 to nodes $3, 4$, and 5 is random and is a model of the mix of part type arriving at the system and of a deterministic routing policy, based on part-type: each type to a different node.

It is assumed that all the queues can be controlled, i.e. their buffer capacity can be modified, satisfying the following constraint, which defines the parameter space Θ:

$$\sum_{(i,\alpha)\in\mathcal{Q}_C} B_{(i,\alpha)} = B_{tot}, \tag{10}$$

where $B_{(i,\alpha)}$ is the buffer capacity of queue (i,α). $\mathcal{Q}_C$ is the set of *controlled queues* (in this case $\mathcal{Q}_C = \{2,3,4,5,6\}$) and B_{tot} is the total buffer capacity available to the controlled queues.

The same system was considered in Liberatore, Nicosia and Valigi (1995b, 1997) but the set of controlled queues was $\mathcal{Q}_C = \{3,4,5\}$ due to the minor accuracy of the perturbation analysis scheme used there, which allowed to consider only smaller perturbations.

Table 1 reports the ordered sequence of parameter values θ based on short run and on long run performance estimates (in particular on a time interval

corresponding to the service of $M = 500$ customers for the short run and $M = 1000000$ for the long run). $L1$ and $L2$ are two independent long run simulations while Sk, $k = 1, 2, \ldots, 10$, denote short run simulations. In the Table the number is an identifier for buffer allocations. For example, 169 is the buffer allocation $\theta_{opt} = (1\,0\,0\,4\,3)$ which is the long run optimal allocation for the considered system. Different experiments, with different realizations of the stochastic process, are reported in the Table. Observe that choosing, after a short run simulation, the best estimated buffer allocation provides very often one of the two best long run solutions while just twice a worse solution was achieved.

In the implementation of the Time Local Control Algorithm, we took as *control period* (or *observation period*) duration T_c, the time needed by the system to service a given constant number M of customers.

In the following we have considered a global control scheme, i.e. $\Theta_m \equiv \Theta$ $\forall m$, and a local control scheme, with Θ_m comprising buffer allocations which differ from the current one by moving just one buffer slot from one queue to another.

If the Θ_m considered contains parameter values under which the current extended state (i.e. the state at time t_{m+1}) is not admissible, it could happen that the current extended state is not admissible under the parameter vector value θ^* selected by the algorithm. This is because there can be some buffers whose current content exceeds the new capacity assigned through θ^*. In these cases, some temporary buffer slots have been added in order to store the exceeding customers. These buffer slots disappear as soon as the exceeding customers are serviced.

In the simulation experiments, all the service times, whose probability distribution is unknown to the control algorithm, are exponentially distributed, with mean service time equal to 0.5, 1, 3, 2, 1, and 1, for nodes from 1 to 6, respectively. The exponential probability distribution is not needed, however, for the application of the algorithm which works under very general conditions. In particular, we do not make any assumption on the stochastic process characterizing the system. The sum B_{tot} of the buffer capacity of the controlled queues is chosen equal to 8 (without taking into account customers under service) while the initial allocation is $\theta = (1\,2\,2\,2\,1)$.

It is assumed that the routing probabilities are unknown, and describe a disturbance acting on the system, that has to be rejected by the control scheme. Both the cases of stationary and time-varying routing probabilities have been considered. In the stationary case, the routing probabilities from node 2 to nodes 3, 4, and 5 have been chosen equal to 0.05, 0.05, and 0.9, respectively. In the time-varying case, the routing probabilities are subject to step-wise changes, with initial values as in the previous case, then the values are set to 0.9, 0.05 and 0.05, respectively, and finally they become 0.05, 0.9 and 0.05, respectively. The control period M has been chosen as $M = 500$. The long term throughput in the stationary scenario of the system controlled by the Time Local Control

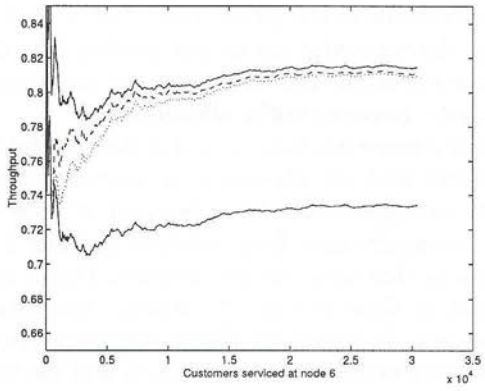

Figure 2. Global and local control scheme: throughput behavior (solid above: optimal; dashed: globally controlled; dotted: locally controlled; solid below: uncontrolled).

Buffer allocation	Buffer allocation identifier	Percentage
$\theta_1 = (10043)$	169	51.67
$\theta_2 = (10034)$	168	31.67
$\theta_3 = (11033)$	204	10
$\theta_4 = (10133)$	176	3.3
$\theta_5 = (10052)$	170	1.67
$\theta_6 = (12221)$	242	1.67

Table 2. Frequency selection of buffer allocations.

Algorithm (using a global and a local scheme) is reported in Fig. 2 and compared with the performance of the system without control scheme, and with two different choices for the buffer capacity. The first choice, $\theta = (1\,2\,2\,2\,1)$, is a symmetric buffer allocation: the capacity is 1 for queues 1 and 6 and 2 for queues $3, 4$ and 5. The second choice is the optimal buffer allocation $\theta_{opt} = (1\,0\,0\,4\,3)$. The diagram clearly indicates the strong improvement on system performance achieved by the proposed control scheme: as a matter of fact, the performances of the controlled system are almost equal to the optimal one.

Observe also that the performance in the global case (dashed) is slightly better in the short term, and becomes identical in the long term to the performance of the local control scheme (dotted).

The computation time required to run the local control scheme here has been about 3% of the time required by the global control scheme.

Table 2 illustrates the frequency with which each buffer allocation has been

chosen by the control scheme in the global case. Notice that only 6 allocations have been chosen by the controller out of the total of 495. Observe the accordance between data reported in this table and the ones corresponding to the long run estimated order between buffer allocations.

Finally, the control scheme has been tested under the effect of the time varying routing probabilities with the changes cited earlier, where the first change happens after 10000 customers have been serviced at node 6 and the second change after 20000 customers have been serviced at node 6. To make apparent the control action in this time-varying scenario, Fig. 3 does not report the long term throughput as done in Fig. 2. Rather, the total simulation time has been partitioned into 30 intervals, where each interval corresponds to the time required to provide service to 1000 customers, and the throughput of every interval has been computed and reported in Fig. 3. Notice that the time of the change is not known by the controller. The effectiveness of the Time Local Control Algorithm is apparent: as a matter of fact the algorithm selects very often in each steady interval the first two better allocations in that interval (e.g. allocations 169 and 168 in the first steady interval). In particular, in each steady interval, the percentage of times the best two allocations in that interval have been selected are 100%, 70%, and 90% respectively.

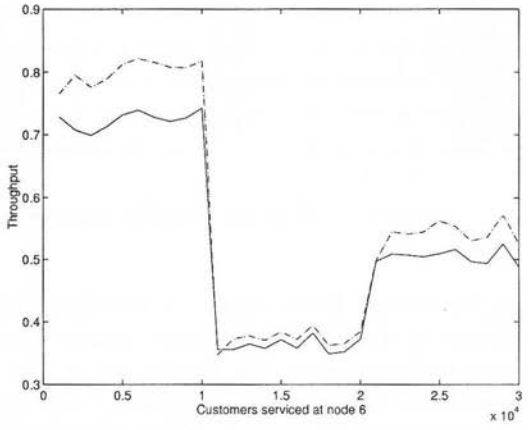

Figure 3. Throughput behavior (time-varying routing; dotted: controlled, solid: uncontrolled).

As a concluding remark, observe that the basic idea of the control scheme considered in this paper is to use the Marking Algorithm to simultaneously estimate system performance for the set of parameter vectors Θ_m. The process of simultaneous estimation is inherently parallel: it amounts to executing several times the Marking Algorithm, with the same observed input sequence, the same observed initial extended state, and different parameter vectors. Once the input

sequence has been determined, the state reconstruction for all the parameters in Θ_m can be carried out independently, and in a concurrent manner. A concurrent state reconstruction algorithm has been presented in Martinelli, Nicosia and Valigi (1997a).

6. The extensions to general DEDS

The control algorithm described in this paper can be easily extended to general DEDS. In particular, equation (1) can be seen as the dynamic equation of any DEDS described through the GSMP formalism. Service times of the machines must be considered, in the general case, as event lifetimes; random scheduling and routing as non-deterministic state transitions in the graph associated with the considered DEDS. The notions of parameter allocations and admissible states depend on the particular DEDS considered.

7. Conclusions

In this paper an exact state reconstruction algorithm for DEDS has been proposed, based on data observed from the system we are controlling, and on a suitable model of the system itself. The algorithm allows to exactly reconstruct the state of the system, under perturbation on the parameters characterizing it. The algorithm has then been used to implement dynamic control schemes for DEDS. In particular, in this paper a global and a local control scheme for dynamic allocation of buffer capacities is proposed, and its application to a manufacturing system is described in detail. A parallel implementation of the whole control scheme is also mentioned. The proposed control scheme can be easily extended to general DEDS and can be successfully applied to control non-stationary DEDS, where the optimal allocation is not always the same in time.

References

CASSANDRAS, C. (1993A) *Discrete Event Systems: Modeling and Performance Analysis.* Irwin & Aksen, Boston, MA.

CASSANDRAS, C. (1993B) Rapid learning techniques for discrete event systems: Some recent results and applications to traffic smoothing. In: *12th IFAC World Congress*, 3, 323–326, Sydney, Australia.

CASSANDRAS, C. AND PANAYIOTOU, C. (1996) Concurrent sample path analysis of discrete event systems. In: *Proc. of the 35th Conference on Decision and Control*, 3332–3337, Kobe, Japan.

CASSANDRAS, C. AND STRICKLAND, S. (1989) Observable augmented systems for sensitivity analysis of Markov and semi-Markov processes. *IEEE Trans. on Automatic Control*, **34**, 10, 1026–1037.

DAI, L. AND CHEN, C. H. (1997) Rates of convergence of ordinal comparison for dependent discrete event dynamic systems. *Journal of Optimization Theory and Applications*, **94**, 1, 29–54.

DAI, L. Y. (1996) Convergence properties of ordinal comparison in the simulation of discrete event dynamic systems. *Journal of Optimization Theory and Applications*, **91**, 363–388.

DI MASCOLO, M., FREIN, Y., DALLERY, Y., AND DAVID, R. (1991) A unified modeling of Kanban systems using Petri nets. *The Int. Journal of Flexible Manufacturing Systems*, **3**, 275–307.

GLASSERMANN, P. (1990) *Gradient Estimation Via Perturbation Analysis.* Kluwer.

HO, Y. AND CAO, X. (1991) *Perturbation Analysis of Discrete Event Dynamic Systems.* Kluwer.

LIBERATORE, G., NICOSIA, S., AND VALIGI, P. (1995A) Dynamic allocation of kanbans in a manufacturing system using perturbation analysis. In: *Proc. of the 1995 INRIA/IEEE Conference on Emerging Technologies and Factory Automation*, Vol. 3, 595–600, Paris, France.

LIBERATORE, G., NICOSIA, S., AND VALIGI, P. (1995B) Path construction techniques for dynamic control of kanbans systems. In: *Proc. of the 34th Conference on Decision and Control*, 2610–2611, New Orleans, LO.

LIBERATORE, G., NICOSIA, S., AND VALIGI, P. (1997) Dynamic allocation of buffer capacity in discrete event systems. *Intelligent Automation and Soft Computing: An International Journal*, **2**, 4, 407–422.

MARTINELLI, F., NICOSIA, S., AND VALIGI, P. (1997A) Dynamic control schemes for manufacturing systems. In: *2nd IFAC Workshop on New Trends in Design of Control Systems*, 393–398, Smolenice, Slovakia.

MARTINELLI, F., NICOSIA, S., AND VALIGI, P. (1997B) A state reconstruction algorithm for parameter dependent discrete event dynamic systems. In: *Proc. of the 5th IEEE Mediterranean Conference on Control and Systems*, Cyprus.

PARK, Y. AND CHONG, E. (1995) Distributed inversion in timed discrete event systems. *Discrete Event Dynamic Systems: Theory and Applications*, **5**, 219–241.

PEPYNE, D. AND CASSANDRAS, C. (1997)Adaptive dispatching control for elevator systems during up-peak traffic. In: *36th Conference on Decision and Control*, San Diego, USA.

VAKILI, P. (1991) A standard clock technique for efficient simulation. *Operation Research Letters*, **10**, 455–452.