A State Reconstruction Algorithm for Parameter Dependent Discrete Event Dynamic Systems

Francesco Martinelli, Salvatore Nicosia, Paolo Valigi
Dipartimento di Informatica, Sistemi e Produzione
Università di Roma "Tor Vergata" – 00133 – Roma – Italy
martinelli@tovvx1.ccd.utovrm.it, {nicosia,valigi}@disp.utovrm.it

Abstract

In this paper, the problem of state reconstruction for the class of parameter dependent discrete event systems that can be modeled by means of queueing networks is considered, and a novel solution is proposed, based on the use of observed data. The algorithm allows to accurately reconstruct the state behavior of a system, for parameter values different from the nominal ones.

The proposed approach has been applied to the on-line control problem for real systems, and a complete control procedure is proposed.

1 Introduction

Discrete Event Systems (DES) are important models for real systems whose state transitions are triggered by the occurrence of discrete events [1, 2, 3], such as communication networks, manufacturing systems, computer systems or traffic networks.

Several techniques based on (observed) sample path analysis have been proposed, aimed at estimating the behavior of a DES under perturbed values of some of its parameters. In particular, perturbation analysis techniques are now well established, and can be used both for continuous and discrete parameters. Similar approaches include augmented system analysis [4], standard clock [5], and rapid learning [6]. Recently, an exact finite perturbation-like algorithm for the analysis of single class queueing network has been proposed, for the case of buffer capacity augmentation [7]. A "time warping" approach, similar to the one proposed here, has been considered in [8]. Modified finite perturbation rules have been proposed in [9, 10, 11] for the case of general queueing network, extending the original results in [12, 13] to the case of weakly adjacent events.

Although, historically, the main motivations for the introduction of these techniques was the attempt to reduce simulation computation effort, they are particularly useful also to implement on-line control schemes for physical systems. 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 asses its behavior. Such a type of on-line comparison instead, is made possible by sample path techniques.

Perturbation analysis techniques attempt to estimate the perturbation in the time of occurrence of events by comparing other time perturbations, as well as nominal timing, without taking into account the state evolution. The main difficulty in such a time "timing comparison" approach is that, in order to correctly compute the time perturbation of an event, almost all the 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.

Here, a novel algorithm is proposed, fully exploiting the benefit of reconstructing the "extended" state of a queueing network: if the network state is available, the "destiny" of a node arriving to a queue can be decided immediately, without any need for future events. This allows to derive exact algorithms, regardless of the magnitude of the parameter perturbation.

The proposed algorithm is based on the assumption that relevant information, such as the system initial state and the system input sequence, can be extracted from the observation of the system over a finite horizon, and properly stored. Based on the knowledge of these data, and assuming a "topological" model of the system is available, the state sequence can be fully reconstructed, for perturbed values of the system parameters. The availability of perturbed state sequences allows to compute performance indices, and to implement dynamic control schemes.

This paper considers the state reconstruction problem, illustrating in details the proposed methodology for on-line state reconstruction and control. In [14], the application of such a methodology to on-line control of manufacturing systems has been tested by means of simulation results. Initial results on the parallel implementation of the algorithm are presented in [15]. The reconstruction algorithm is mainly intended for on-line control of real systems, and therefore computational efficiency is not the major issue.

2 Queueing network dynamics

In this paper it will be considered the class of systems that can be modeled by means of queueing networks with: (a) general service time at each node, (b) general rout-

ing policy at each node, (c) general scheduling policy at each node, (d) finite buffer capacity, (e) multi-class, (f) non-preemptive service at each node, (g) infinite arrival rate sources, (h) infinite capacity sinks. Each node of a network comprises a server, which has as much input queues as the number of classes it can provide service to, and each class is associated a dedicated input queue.

The event of type "completion of service of a customer on a node" is sufficient to fully describe the evolution of the class of queueing networks considered here.

Notation

Let N_S denote the number of servers in the network, N_C the number of customer classes serviced by the network, N_Q the total number of queues in the network; let $\mathcal{N} := \{1, 2, \ldots, N_S\}$ denote the set of all the nodes, $\mathcal{C} := \{1, 2, \ldots, N_C\}$ the set of all the customer classes, $\mathcal{C}_i$ the set of all the classes that may be serviced at node i, and $\mathcal{Q} := \{(i, \alpha) : \alpha \in \mathcal{C}_i, i \in \mathcal{N}\}$ the set of all the network queues; hence $N_Q = |\mathcal{Q}|$. Finally, let $N_{C,i}$ denote the number of elements in $\mathcal{C}_i$ ($N_Q = \sum_{i=1}^{N_S} N_{C,i}$).

The following three types of counters will be used. The first counter is global, is used to count the service completion events occurred over the whole network, is denoted by n, and is referred to as the *global counter*. The second type of counters are local to each node; node i counter is denoted by k_i, $i \in \mathcal{N}$, counts the total number of service completion events occurred at node i, regardless of the customer class, and is referred to as a *node counter*. The third type of counters are local to each node as well; counter $k_{i,\alpha}$, $\alpha \in \mathcal{C}_i$, $i \in \mathcal{N}$, is used to count the number of class α customers serviced at node i, and is referred to as a *class counter*. Clearly, $\sum_{\alpha \in \mathcal{C}_i} k_{i,\alpha} = k_i$.

The event of service completion of a class α customer at the i-th node will be denoted by $c_{i,\alpha}$, the service completion of the $k_{i,\alpha}$-th customer of class α at node i will be denoted by $c_{i,\alpha}(k)$, the time of occurrence of the completion event $c_{i,\alpha}(k)$ will be denoted by $t^c_{i,\alpha}(k)$; the completion time of the customer currently under service on node i, or the completion time of the last customer serviced by node i, if node i is currently starved or blocked, will be simply denoted by t^c_i, and $t^c := (t^c_1, t^c_2, \ldots, t^c_{N_S})^T$ will denote the vector of all completion times.

The vector x of the *queueing state* is given by $x := (x_L^T, x_C^T, x_B^T)^T$, where x_L is the N_Q dimensional vector of the current queue lengths (without counting the customers under service), x_C is the N_S dimensional vector of the classes of customers currently under service (with $x_{C,i} = 0$, $i \in \mathcal{N}$ denoting that server i is empty/starved), and finally x_B is the N_S dimensional vector indicating whether a server is blocked or not. The above structure of the state vector implies that each node can be blocked by only one other node. Hence, splitting nodes are not allowed here, while assembling/joining nodes are.

Let $\mathcal{X}$ denote the state space of the queueing network, let $\mathcal{E} = \{c_{i,\alpha}, \alpha \in \mathcal{C}_i, i \in \mathcal{N}\}$, denote the set of all the completion events, let $\mathcal{A}(x) := \{i : (x_C(i) \neq 0) \wedge (x_B(i) = 0), i \in \mathcal{N}\}$, for all $x \in \mathcal{X}$, denote the set of servers active while the network is in state x, and let $\mathcal{R}(x'; x, s) := \{i : i \in \mathcal{A}(x') \setminus (\mathcal{A}(x) \setminus \{s\}), i \in \mathcal{N}\}$ denote the set of servers re-activated upon the transition from state x to state x' due to service completion at server s.

It is assumed that randomness in state transition can only arise from random scheduling policies, and/or random routing policies. The stochastic structure of these policies is not relevant, it will be simply assumed that each node i, $i \in \mathcal{N}$, where a random scheduling policy is used, is associated a random sequence $\{u_{S,i}\}_{k_i}$, taking values in $\mathcal{C}_i$, and indexed by the node counter k_i. Similarly, it is assumed that each node i, $i \in \mathcal{N}$, where a random routing policy is used, are associated $N_{C,i}$ random sequences $\{u_{R,i,\alpha}\}_{k_{i,\alpha}}$, taking values in $\mathcal{Q}$, indexed by the class α counter $k_{i,\alpha}$ of node i.

As for the randomness in customer service duration, for the purpose of this paper it will be assumed that each node i, $i \in \mathcal{N}$, are associated $N_{C,i}$ random sequences $\{u_{D,i,\alpha}\}_{k_{i,\alpha}}$, taking values in $\mathbb{R}^+$ (positive real numbers), indexed by the class counters, with the $\bar{k}_{i,\alpha}$-th element of the α-th sequence of node i being the random variable determining the service duration of the $\bar{k}_{i,\alpha}$-th class α customer serviced by node i.

Let $(\Omega, \mathcal{B}, P)$ be a common probability space for the whole the random sequences $\{u_{S,i}\}_{k_i}$, $\{u_{R,i,\alpha}\}_{k_{i,\alpha}}$, and $\{u_{D,i,\alpha}\}_{k_{i,\alpha}}$, then, each element $\omega \in \Omega$ corresponds to a unique realization of these sequences.

Queueing network dynamics

Let $x(0) \in \mathcal{X}$ be the initial queueing state of the network, let $t(0) \in \mathbb{R}$ be the initial time, and assume the initial service completion times $t^c_i(0)$, $t^c_i(0) \in \mathbb{R}$, are given for all $i \in \mathcal{N}$. Set to zero all the counters, i.e., $n = 0$, $k_i = 0$, for all $i \in \mathcal{N}$, $k_{i,\alpha} = 0$, for all $\alpha \in \mathcal{C}_i$, and for all $i \in \mathcal{N}$. The determination of the next queueing state is based on the computation of the server s^* where the next service completion will occur, and the corresponding service completion time t^*:

$$t^* = \min_{i \in \mathcal{A}(x(n))} \{t^c_i(n)\}, \tag{1a}$$

$$s^* = \min\{j : t^c_j(n) = t^*, j \in \mathcal{A}(x(n))\}, \tag{1b}$$

where the *min* operation in (1b) allows to select a unique server. Given t^* and s^*, the new state and completion times can be computed as:

$$x(n+1) = \phi_x(x(n), t^c(n), s^*, \bar{u}_R, \bar{u}_S), \tag{1c}$$

$$t(n+1) = t^*, \tag{1d}$$

$$t^c(n+1) = \phi_t(x(n), t^c(n), s^*, \bar{u}_D), \tag{1e}$$

$$n = n + 1 \tag{1f}$$

where the vectors $\bar{u}_S$, $\bar{u}_R$, and $\bar{u}_D$ are realization of a suitable number of random variables from 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 the i-th component $\phi_{t,i}$ of function ϕ_t is defined as:

$$\phi_{t,i}(x(n), t^c(n), s^*, \bar{u}_D) = \begin{cases} t^* + \bar{u}_{D,i}, & \text{if } i \in \mathcal{R}(x(n+1); x(n), s^*), \\ t_i^c(n), & \text{otherwise}, \end{cases}$$

where $\bar{u}_{D,i}$ denotes the new service duration for server i. Notice that equations (2), describing the network dynamics, are completely deterministic and can be evaluated provided that vectors $\bar{u}_R$, $\bar{u}_S$, and $\bar{u}_D$ are given.

Now, let the sets of known infinite sequences $u_S(\cdot)$, $u_R(\cdot)$, and $u_D(\cdot)$, be a realization 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. Let $u(\cdot) := \{u_S(\cdot), u_R(\cdot), u_D(\cdot)\}$ denote the set of all the known sequences $u_S(\cdot)$, $u_R(\cdot)$, and $u_D(\cdot)$. Formally, $u(\cdot)$, the *input sequence*, can be indexed by the vector $([k_i]\,[k_{i,\alpha}])$ of all the node and class counters. Finally, let the set $\mathcal{U}$ of all the *admissible input sequences* be the set of all the deterministic sequences $u(\cdot)$ that are realizations of the stochastic sequences $\{u_{S,i}\}_{k_i}$, $\{u_{R,i,\alpha}\}_{k_{i,\alpha}}$, and $\{u_{D,i,\alpha}\}_{k_{i,\alpha}}$, for some value $\omega \in \Omega$.

Then, the whole state evolution can be computed via the map $\Phi = (\Phi_x^T\ \Phi_t^T)^T$, obtained by the formal composition of the equations (1a)-(1f) governing a single state transition:

$$x(\cdot) = \Phi_x(x(0), t^c(0), u(\cdot)), \tag{2a}$$
$$t^c(\cdot) = \Phi_t(x(0), t^c(0), u(\cdot)). \tag{2b}$$

For a given initial queueing state $x(0)$, and initial service completion times $t^c(0)$, a given infinite input sequence $u(\cdot)$ yields a unique infinite state sequence $x(\cdot)$ and a unique service completion time sequence $t^c(\cdot)$ provided that both the initial state $x(0)$ and the initial service completion times $t^c(0)$ are fully specified.

In the following the vector $x_e := (x^T, (t^c)^T)^T$ will be used to represent the "complete" state, and referred to as the network *extended state*, with *extended state space* $\mathcal{X}_e := \mathcal{X} \times \mathbb{R}$. Then, (2) can be rewritten as:

$$x_e(\cdot) = \Phi(x_e(0), u(\cdot)), \tag{3}$$

and function Φ will be referred to as the *input to extended state map*. It is stressed that equation (3) is completely deterministic, and its solution depends on completely deterministic initial extended state and input sequence.

Parameter dependent queueing networks

Let $\theta \in \Theta$ be the vector of all the *network parameters* that may change value during network operation, and whose impact on network performance is of interest. It is assumed that the *parameter space* Θ has a finite number of elements and does not contain structural parameters, such as, e.g., the number of nodes in the network. Hence, a third argument θ will be added to map Φ:

$$x_e(\cdot) = \Phi(x_e(0), u(\cdot), \theta). \tag{4}$$

It is important to stress that, for queueing networks, the values that the state may assume are not independent from network parameters. 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. The state space $\mathcal{X}$ is given by $\mathcal{X} = \cup_{\theta \in \Theta} \mathcal{X}(\theta)$. A given queueing state x is an *admissible queueing state under parameter* θ if $x \in \mathcal{X}(\theta)$.

For example, if θ is the N_Q dimensional vector of buffer capacities, with $\theta_{(i,\alpha)}$ denoting the buffer capacity of the queue $(i,\alpha) \in \mathcal{Q}$, then the set $\mathcal{X}(\theta)$ is given by:

$$\begin{aligned} \mathcal{X}(\theta) := \{&(x_L^T \quad x_C^T \quad x_B^T)^T \\ &: x_{L,(i,\alpha)} \in \{0, 1, \ldots, \theta_{(i,\alpha)}\},\ (i,\alpha) \in \mathcal{Q}, \\ &x_{C,j} \in \mathcal{C}_j,\ j \in \mathcal{N}, x_{B,\ell} \in \mathcal{Q}, \ell \in \mathcal{N}\} \end{aligned}$$

where $x_{L,(i,\alpha)}$, $x_{C,j}$, and $x_{B,\ell}$ denote the (i,α)-th, j-th, and ℓ-th component of the queueing state sub-vectors x_L, x_C, and x_B, respectively.

3 State reconstruction

The solution to the state reconstruction problem considered in this paper, is based on the assumption that, for a given physical plant, modeled by a queueing network, it is possible to observe its behavior over a finite horizon, and in particular it is possible to observe the input sequence and the initial extended state.

The extended state sequence, for a given observed input sequence, a given observed initial state, and some values of the network parameter, is reconstructed by means of the system dynamics, i.e., by means of the map (4).

To illustrate the basic idea of the proposed state reconstruction algorithm, the following problem is initially considered, though its solution is not physically realizable.

Problem 1 Assume the evolution of a given queueing network over an infinite horizon is observed, for a nominal network parameter vector $\theta = \theta^N$. Let $x_{e,0}^O$ be the *observed initial extended state*, $u^O(\cdot)$ the *observed input sequence*, and $x_e^O(\cdot) = \Phi(x_{e,0}^O, u^O(\cdot), \theta^N)$ the observed extended state sequence. For such a queueing network, find the extended state sequence $\hat{x}_e(\cdot)$ corresponding to a perturbed value $\theta = \hat{\theta}$ of the network parameter vector, from the same initial condition $x_{e,0}^O$, under the same input sequence $u^O(\cdot)$. □

Problem 1 is solved by the following theorem, under the only assumption that the observed initial queueing state is admissible under perturbed parameter $\hat{\theta}$. Its proof is easy, hence omitted.

Theorem 1 *If the observed initial extended state* $x_{e,0}^O = ((x_0^O)^T\ (t_0^{c\,O})^T)^T$ *is such that* $x_0^O \in \mathcal{X}(\hat{\theta})$, *then Problem 1 is solved by the following algorithm:*

$$\hat{x}_e(\cdot) = \Phi(x_{e,0}^O, u^O(\cdot), \hat{\theta}). \tag{5}$$

Theorem 1 allows to solve the state reconstruction problem provided an infinite input sequence is available. Instead, in real applications, only finite input sequences are available, and one is interested in making the best use of them, i.e., in reconstructing the longest possible extended state sequence. To formally state this requirement, the following notation and machinery will be used.

Let $u(\cdot)$ be an admissible infinite input sequence, and let $[k_{i,\alpha}]$ denote the vector containing all the N_Q class counters. Then, $u_{[k_{i,\alpha}]}(\cdot)$ denotes the unique $[k_{i,\alpha}]$-*length finite initial subsequence* (briefly, $[k_{i,\alpha}]$-*subsequence*) of the infinite sequence $u(\cdot)$, i.e., the finite subsequence containing, for all $\alpha \in \mathcal{C}_i$ and for all $i \in \mathcal{N}$, all the elements of the corresponding sequences $u_R(\cdot)$ and $u_D(\cdot)$ in $u(\cdot)$ with index less than or equal to $k_{i,\alpha}$, and all the elements of the corresponding sequence $u_S(\cdot)$ with index less than or equal to $k_i = \sum_{\alpha \in \mathcal{C}_i} k_{i,\alpha}$. In addition, given a $[k_{i,\alpha}]$-subsequence $\bar{u}_{[k_{i,\alpha}]}(\cdot)$, let $\mathcal{U}(\bar{u}_{[k_{i,\alpha}]}(\cdot))$ denote the set of all the admissible infinite sequences having $\bar{u}_{[k_{i,\alpha}]}(\cdot)$ as the common $[k_{i,\alpha}]$-length finite initial subsequence:

$$\mathcal{U}(\bar{u}_{[k_{i,\alpha}]}(\cdot)) := \{u(\cdot) \in \mathcal{U} : u_{[k_{i,\alpha}]}(\cdot) = \bar{u}_{[k_{i,\alpha}]}(\cdot)\}. \tag{6}$$

Now, assume a $[k_{i,\alpha}]$-subsequence $\bar{u}_{[k_{i,\alpha}]}(\cdot)$, an initial extended state $x_{e,0}$, and a network parameter $\bar{\theta}$ are given. Then, for each infinite sequence $u(\cdot)$ in $\mathcal{U}(\bar{u}_{[k_{i,\alpha}]}(\cdot))$, the corresponding extended state sequence $x_e(\cdot)$ from the initial extended state $x_{e,0}$ can be computed, and the set $\mathcal{S}(x_{e,0}, \bar{u}_{[k_{i,\alpha}]}(\cdot), \bar{\theta})$ can be constituted as:

$$\mathcal{S}(x_{e,0}, \bar{u}_{[k_{i,\alpha}]}(\cdot), \bar{\theta}) :=$$
$$\{x_e(\cdot) : x_e(\cdot) = \Phi(x_{e,0}, u(\cdot), \bar{\theta}),\ u(\cdot) \in \mathcal{U}(\bar{u}_{[k_{i,\alpha}]}(\cdot))\}.$$

The set $\mathcal{S}(x_{e,0}, \bar{u}_{[k_{i,\alpha}]}(\cdot), \bar{\theta})$ comprises all the infinite extended state sequences that can be generated from all the admissible input sequences having $\bar{u}_{[k_{i,\alpha}]}(\cdot)$ as the common $[k_{i,\alpha}]$-subsequence.

Given an extended state sequence $x_e(\cdot)$, the sub-sequence obtained by taking the first $\bar{n}$ terms (the n-th term is the value that the extended state assumes when the global counter is equal to n) will be denoted by $[x_e(\cdot)]_{\bar{n}}$ and will be referred to as the $\bar{n}$-subsequence of the infinite sequence $x_e(\cdot)$.

Given the set $\mathcal{S}(x_{e,0},\ \bar{u}_{[k_{i,\alpha}]}(\cdot),\ \bar{\theta})$, a k-subsequence $[x_e^*(\cdot)]_k$ is a common k-subsequence of $\mathcal{S}(x_{e,0},\ \bar{u}_{[k_{i,\alpha}]}(\cdot), \bar{\theta})$ if $[x_e(\cdot)]_k = [x_e^*(\cdot)]_k$ for all $x_e(\cdot) \in \mathcal{S}(x_{e,0}, \bar{u}_{[k_{i,\alpha}]}(\cdot), \bar{\theta})$.

The state reconstruction problem for finite input sequences can be formally stated as follows.

Problem 2 Assume the evolution of a given queueing network over a finite horizon is observed, for a nominal network parameter vector $\theta = \theta^N$. Let $x_{e,0}^O$ be the observed initial extended state, and $u_{[k_{i,\alpha}^O]}^O(\cdot)$ the observed $[k_{i,\alpha}^O]$-length input sequence. For such a queueing network, with a perturbed value $\theta = \hat{\theta}$ of the network parameter vector, find the longest common $\hat{k}$-subsequence of $\mathcal{S}(x_{e,0}^O, u_{[k_{i,\alpha}^O]}^O(\cdot), \hat{\theta})$.

Problem 2 is initially solved, under the following simplifying assumption, by means of the subsequent algorithm.

Assumption 1 *The queueing network only comprises servers with deterministic scheduling policy.*

Algorithm 1 *Deterministic Marking Algorithm*

- Given the finite observed input sequence $u_{[k_{i,\alpha}^O]}^O(\cdot)$, let $u_+^O(\cdot)$ be the infinite sequence obtained from $u_{[k_{i,\alpha}^O]}^O(\cdot)$ by adding an infinite number of mark symbols after the last terms of the sequences in $u_{[k_{i,\alpha}^O]}^O(\cdot)$.
- Apply the reconstruction algorithm given by $\hat{x}_e(\cdot) = \Phi(x_{e,0}^O, u_+^O(\cdot), \hat{\theta})$ and terminate the extended state reconstruction when a server which needs a new service duration value extracts the first mark symbol. ∇

Theorem 2 *If Assumption 1 holds, then, if the observed initial extended state* $x_{e,0}^O = ((x_0^O)^T\ (t_0^{c\,O})^T)^T$ *is such that* $x_0^O \in \mathcal{X}(\hat{\theta})$, *and the probability distributions of all the random variables determining the customer service duration* τ, *for each server, are so that, for all* $\epsilon > 0$, *Prob*$(0 < \tau \leq \epsilon) > 0$, *then Problem 2 is solved by the Deterministic Marking Algorithm.*

The theorem proof, omitted for brevity, can be found in [16]. The extension to the general case of servers with random scheduling is not difficult, and is given in the following algorithm. To simplify notation, assembly nodes are not allowed.

Algorithm 2 *Marking Algorithm*

- Given $u_{[k_{i,\alpha}^O]}^O(\cdot)$, let $u_+^O(\cdot)$ the infinite sequence obtained from $u_{[k_{i,\alpha}^O]}^O(\cdot)$ by adding infinite mark symbols after the last terms of all the sequences in $u_{[k_{i,\alpha}^O]}^O(\cdot)$.
- Associate each server i using a random scheduling policy a set F_i, initially empty, and $N_{C,i}$ flags $f(i,\alpha)$, initially set to zero.
- Apply the algorithm $\hat{x}_e(\cdot) = \Phi(x_{e,0}^O, u_+^O(\cdot), \hat{\theta})$, and, 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:

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

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

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

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

• The algorithm 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 3 summarizes the results of the solution to Problem 2, in the general case in which both deterministic and random scheduling policies are allowed. Its proof is omitted [16].

Theorem 3 *Assume the queueing network does not contain assembly nodes, then, if the observed initial extended state* $x^O_{e,0} = ((x^O_0)^T\ (t^{c,O}_0)^T)^T$ *is such that* $x^O_0 \in \mathcal{X}(\hat{\theta})$, *and the probability distributions of all the random variables determining the customer service duration* τ, *for each server, are so that, for all* $\epsilon > 0$, $Prob(0 < \tau \leq \epsilon) > 0$, *then Problem 2 is solved by the Marking Algorithm.*

4 Dynamic control

The behavior of a queueing network can be assessed by means of a performance index $\mathcal{J}(\theta)$, depending on the network parameter vector θ.

The queueing network control proposed in this paper is implemented on line, by periodically modifying the value of the network parameter θ, selecting the new value in an admissible subset of Θ (to be defined), with the objective of maximizing $\mathcal{J}(\theta)$ with respect to θ. Hence, a *control action* consists in a variation of the network parameter.

The basic idea of the proposed control scheme is as follows. Suppose m control actions have been already implemented, $m \geq 0$, and assume the value of the network parameter, after the m-th control action, is $\theta = \theta_m$.

Control actions are taken with a *control period* (or *observation period*) equal to the time required to the system to complete service on M customers. Parameter M is a design parameter.

Within each control period, the network behavior is observed, and relevant data, e.g. the observed input sequence, are stored for later processing. At the beginning of each control interval, all the counters are reset to zero. During the m-th observation period, the performance index $\mathcal{J}(\theta_m)$ is computed, and the input $[k^m_{i,\alpha}]$-subsequence $u^m_{[k^m_{i,\alpha}]}(\cdot)$, the initial and final (i.e., at the end of the period) extended state $x^m_{e,0}$ and $x^m_{e,f}$ respectively, are recorded.

The evaluation of $\mathcal{J}(\theta)$, for all $\theta \in \Theta_m$, with Θ_m a suitable set to be defined, can be carried out by reconstructing the extended state subsequence for every $\theta \in \Theta_m$, by means of the marking algorithm described above, with the input $[k^m_{i,\alpha}]$-subsequence $u^m_{[k^m_{i,\alpha}]}(\cdot)$ recorded during the observation period, and a suitable initial extended state $\bar{x}_{e,0}$, to be computed taking into account the observed initial state $x^m_{e,0}$ and the network parameter for which the extended state has to be reconstructed.

Given the reconstructed extended state subsequence, it is possible to compute the performance index $\mathcal{J}(\theta)$, for all $\theta \in \Theta_m$, and to choose, as the current control action, the network parameter $\theta^* \in \Theta_m$ such that:

$$\mathcal{J}(\theta^*) = \max_{\theta \in \Theta_m} \mathcal{J}(\theta). \tag{7}$$

Given the final extended state $x^m_{e,f}$, the set Θ_m can be defined as:

$$\Theta_m = \{\theta \in \Theta : x^m_f \in \mathcal{X}(\theta)\}, \tag{8}$$

where x^m_f is the queueing state sub-vector corresponding the the extended state $x^m_{e,f}$. Set Θ_m can be reduced, by considering suitable constraints, e.g., in order to reduce computational complexity, or to take into account constraints in the real system.

The initial extended state $x^a_{e,0}$ to be used by the reconstruction algorithm can be selected by one of the two following rules, where the distance funcion $d(\cdot,\cdot)$ is defined as $d(x^1,x^2) := \sum_{i=1}^{N_Q} \left|x^1_{L,i} - x^2_{L,i}\right|$, whit $x^j_{L,i}$ denoting the i-th entry of the $x_L$component of the queueing state vector x^j.

Rule a). Select the initial extended state $x^a_{e,0}$=$((x^a_0)^T\ (t^{c,m}_0)^T)^T$ such that:

$$x^a_0 = \min_{x_0 \in \mathcal{X}(\theta), \forall \theta \in \Theta_m} \{d(x_0, x^m_0)\}. \tag{9}$$

Rule b). For all $\theta \in \Theta_m$, select the initial extended state $x^{a,\theta}_{e,0} = ((x^{a,\theta}_0)^T (t^{c,m}_0)^T)^T$ such that

$$x^{a,\theta}_0 = \min_{x_0 \in \mathcal{X}(\theta)} \{d(x_0, x^m_0)\}. \tag{10}$$

5 Simulation results

Several simulation tests have been performed, aimed at verifying the performance of the proposed real-time control scheme, and of the underlying state reconstruction algorithm. Here only a brief summary is given, a more detailed description can be found in [14, 15].

The results refer to the dynamic control of a simple multi-class manufacturing system, depicted in figure 1,

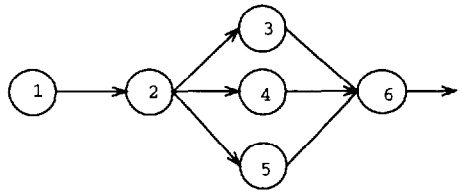

Figure 1: The manufacturing system

and in which control actions amount to allocation of buffer capacities of the nodes $2, 3, 4, 5, 6$, referred to as the set of *controlled nodes*, with the constraint that the sum of all the capacities remains constant, and equal to 8. It is assumed that the service duration of all the nodes is independent of part type, nodes $3, 4$, and 5 works only one part-type each. The routing probability from node 2 to nodes $3, 4$, and 5 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.

All the simulation experiments have been carried out with the following specifications for the network: exponentially distributed service times, with mean equal to 0.5, 1, 3, 2, 1, and 1, for nodes from 1 to 6, respectively. Routing probabilities from node 2 to nodes $3, 4$, and 5 equal to 0.05, 0.05, and 0.9, respectively.

The performance index to be maximized is the system throughput, computed over 30000 part serviced by node 6. The control period M has been chosen as $M = 500$.

The optimal buffer capacity allocation for the system without any control scheme active has been determined by means of independent simulations. Then, the simulation of the controlled system indicates that almost 80% of the control actions coincide with the buffer allocation, which is the optimal one for the system without control.

From the point of view of performance, simulation results indicate that the controlled system achieves a system throughput of about 0.81 (over a throughput of 0.815 for the optimal system, i.e., of the system without control scheme, and with fixed optimal buffer capacity allocation), with a 14% improvement over the throughput of the uncontrolled system (i.e., the system without control scheme, and with an fixed buffer capacity allocation of $(1, 2, 2, 2, 1)$), which turns out to be equal to 0.72.

Acknowledgment

This work has been supported by MURST (ex) 40 % and 60 % funds.

References

[1] P. Glassermann, *Gradient Estimation Via Perturbation Analysis*. Kluwer Academic, 1990.

[2] Y. Ho and X. Cao, *Perturbation Analysis of Discrete Event Dynamic Systems*. Kluwer Academic Publishers, 1991.

[3] C. Cassandras, *Discrete Event Systems: Modeling and Performance Analysis*. Boston, MA: Irwin & Aksen, 1993.

[4] C. Cassandras and S. Strickland, "Observable augmented systems for sensitivity analysis of Markov and semi-Markov processes," *IEEE Trans. on Automatic Control*, vol. 34, no. 10, pp. 1026–1037, 1989.

[5] P. Vakili, "A standard clock technique for efficient simulation," *Operation Reseacrh Letters*, vol. 10, pp. 455–452, 1991.

[6] C. Cassandras, "Rapid learning techniques for discrete event systems: Some recent results and applications to traffic smoothing," in *12th IFAC World Congress*, vol. 3, (Sydney, Australia), pp. 323–326, July 1993.

[7] C. G. Panayiotou and C. Cassandras, "Optimization of kanban-based production systems," in *proc. of the WODES 96, International Workshop on Discrete Event Systems*, (Edinburgh (Scothland): IEE Computing Control Division), 1996.

[8] C.G. Cassandras and C. G. Panayiotou, "Concurrent sample path analysis of discrete event systems," in *Proc. of the 35th Conf. on Decision and Control*, Kobe (Japan), December, 1996.

[9] G. Liberatore, S. Nicosia, and P. Valigi, "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, (Paris, France), pp. 595–600, October, 10-13 1995.

[10] G. Liberatore, S. Nicosia, and P. Valigi, "Path contruction techniques for dynamic control of kanbans systems," in *Proc. of the 34th Conference on Decision and Control*, (New Orleans, LO), pp. 2610–2611, December 1995.

[11] G. Liberatore, S. Nicosia, and P. Valigi, "Dynamic allocation of buffer capacity in discrete event systems," *Journal of Intelligent Automation and Soft Computing*, 1997.

[12] Y. Ho, M. Eyler, and T. T. Chien, "A gradient technique for general buffer storage design in a production line," *Int. Journ. of Prod. Res.*, vol. 17, no. 6, pp. 557– 580, 1979.

[13] Y. Ho, X. Cao, and C. Cassandras, "Infinitesimal and finite perturbation analysis for queueing networks," *Automatica*, vol. 19, pp. 439–445, 1983.

[14] F. Martinelli, S. Nicosia, and P. Valigi, "Dynamic control of manufacturing systems based on a novel state reconstruction algorithm," in *1997 IEEE Int. Conference on Robotics and Automation*, (Albuquerque, NM), April 20-25 1997.

[15] A. Loretucci, F. Martinelli, S. Nicosia, and P. Valigi, "A parallel algorithm for on-line simulation and control of discrete event systems," in *Annual Conference of the Italian Society for Computer Simulation*, (Rome), December 18-19 1996.

[16] F. Martinelli, S. Nicosia, and P. Valigi, "State reconstruction techniques for discrete event dynamic systems: Theory and application," Tech. Rep. R.01-97, Dipartimento di Informatica, Sistemi e Produzione, Università di Roma "Tor Vergata", Roma, 1997.