Dynamic Control of Buffering Capacity in Discrete Event Systems[1]

Giovanni Liberatore, Salvatore Nicosia, Paolo Valigi
Dipartimento di Ingegneria Elettronica, Università di Roma "Tor Vergata"
Via della Ricerca Scientifica, 00133 Roma - Italy
Tel. +39.6.7259-4405, e-mail: valigi@eln.utovrm.it

Abstract In this paper, an application of perturbation analysis techniques to the problem of dynamic allocation of buffering capacity in a discrete event dynamic system is studied. This problem is of extreme interest e.g. in Kanban systems, where the buffer sizes are equal to the number of Kanbans of each working centre. The main contribution of the paper consists of an on-line control strategy, aimed at dynamically varying the buffering capacity of the servers comprising the system, in order to keep the performance of the system as high as possible, despite unknown disturbances entering the system. The proposed control strategy has been tested by means of simulation on a multi part-type Kanban system. In this case, the control action specializes to the dynamic determination of the number of Kanbans associated to each part-type, in order to counteract the effect of unknown changes in the product-mix on the system throughput. Several simulation tests have been performed in order to verify the behaviour of the proposed control strategy under different frequencies of the control action.

1. Introduction

Discrete Event Dynamic Systems (DEDS) have become important models of real systems, such as communication networks, manufacturing systems, computer systems or traffic networks; in these systems state transitions are triggered by the occurrance of particular events [1, 2]. Perturbation Analysis (PA) has been succesfully used to estimate the parameter sensitivities of performance measures of the systems above, calculating these sensitivities by only analyzing a single path.

Two basic PA methods have been developed: Infinitesimal Perturbation Analysis (IPA) and Finite Perturbation Analysis (FPA) [1]-[3]. In this paper it is considered the case of discrete parameter perturbations, which, leading to order changes of some events, makes the IPA estimates not applicable and requires an FPA approach.

[1]This work has been supported by Ministero dell'Università e della Ricerca Scientifica e Tecnologica 40% and 60% funds.

Perturbation analysis allows to estimate the timings of each event in the evolution of a DEDS, under different parameter values, therefore to estimate a perturbed path [1, 4, 5]. This means that it is possible to estimate the behaviour of the system under parameter perturbations, by only observing its behaviour with the nominal parameter values, without actually changing anything [1, 2, 3]. In this paper perturbation analysis has been applied to the study of the effects of buffer size perturbations on the performance measure of a special case of discrete event dynamic system, namely a queueing network, modeling a Kanban system. The results obtained bring an interesting contribution to existing works dealing with buffer size perturbations (see e.g. [6]), or queue lenth perturbations (see e.g. [7]).

The problem of inventory control and buffer capacity allocation in production facilities is long studied, and the introduction of Kanban in the Just-In-Time philosophy motivates additional interest for this subject [8, 9]: in these systems, for instance, it is really useful to know the effect of a buffer size perturbation, being the number of Kanban the maximum number of pieces that can be stocked in a working centre. Any action aimed at changing this number, can easily be taken, since a Kanban is simply a card, attached on the pieces to be processed. A general description of the various Just-In-Time systems and models can be found, e.g., in [10, 11] and in the references therein.

The proposed scheme for path estimation has been applied to an on-line control strategy. The throughput of the perturbed system, whose path is estimated using perturbation analysis, is employed to maximize the performance of the discrete event system subject to unknown disturbances. The developed procedure is divided into two stages: firstly, the behaviour of the nominal system is observed, and, meanwhile, its performance under several buffer size perturbations are estimated; secondly, the change which is expected to bring the higher performance improvement is actually implemented. The perturbation

analysis algorithm presented here allows to "foresee" the effect of several control actions without having to implement them; the one which is supposed to cause the greater improvement, is then performed.

The proposed scheme has been applied to a multi part-type Kanban system, subject to unknown changes in the product-mix; a control action consists of adjusting the number of Kanbans of the working centres. Perturbation analysis for on-line control has been applied in [6] to a flow control strategy in a queueing system; in [12] to a routing algorithm for a communication network; in [13] to design the parameter of a feedback control law; and in [14] to allocate transmission capacity in an ATM network. The finite perturbation analysis approach proposed in this paper represents a possible mechanism for the analysis of the "improvability" of a manufacturing system with respect to its buffer size [15].

In Section 2 the basic concepts of finite perturbation analysis are introduced. In Section 3 the procedure, specialized to the case of buffer size perturbation, is used to develop an on-line control strategy. Simulation results for a Kanban system are given in Section 4 to demonstrate the usefulness of the proposed approach and the rapidity of the control law.

2. Perturbation analysis

Discrete event dynamic systems are usually characterized by a state-space, described by a "discrete set" and state transitions determined by the occurrence of "events". Thus, in general, the state of a DEDS is a piece-wise constant function of time with jumps at the times of event occurrence. In general each event is characterized by an event-*type* or event-*nature* and by its *time of occurrence*, briefly event-*time*; e.g. in a queueing network, the arrival of a customer or the departure of a customer are events of two different types. In the framework considered in this paper the concept of *Path* of a DEDS plays a major role: a *Path* is a sequence of pair (event-type, event-time), i.e. $(e(i,\xi),t(i,\xi))$, with $i \in \mathbb{Z}$, where $e(i,\xi)$ indicates the type of the i-th event, $t(i,\xi)$ its time of occurrance. The variable ξ is an element of the underlying probability space and indicates the random character of the particular production activity. The behaviour of a system depends, in general, on a set of parameters, indicated by the vector θ. For a given value of ξ, the sequence $(e(i,\xi),t(i,\xi))$, $i \in \mathbb{Z}$, when the system parameters assume their nominal values θ is said *nominal path*; the sequence, $(\tilde{e}(i,\xi),\tilde{t}(i,\xi))$, $i \in \mathbb{Z}$, for the same value of ξ, when θ is varied of $\Delta\theta$, is called *perturbed path.*

A syntethic indicator of some features of the system is denoted with $L(\xi,\theta)$, and called *sample performance measure* when referred to a nominal path, *sample perturbed performance measure*, $L(\xi,\theta+\Delta\theta)$, when referred to the perturbed path.

The procedure to estimate the sample perturbed performance measure, used in this paper, is based on an approximate construction of the perturbed path, obtained from the observation of a nominal path. The approach here used consist of an application of finite perturbation analysis rules to a finite length sample path of a discrete event dynamic systems.

The basic goal of the developed procedure is the estimation, for every event $e(i,\xi)$, of its *perturbed event-time*, $\tilde{t}(i,\xi)$, and thus of the perturbed path. This can be done by evaluating the *time perturbation* of every event: $\Delta t(i,\xi) = \tilde{t}(i,\xi) - t(i,\xi)$

As in the classical perturbation analysis approach, two types of time perturbations have been considered. The first one depends directly on the parameter perturbation $\Delta\theta$; it is computed by means of a function called *generation function*, of the form

$$\Delta t(i,\xi) = Gen(\Delta\theta, e(i,\xi), t(i,\xi), e(j,\xi), t(j,\xi), \Delta t(j,\xi)), \tag{1}$$

where $e(j,\xi)$ is another event influencing the time perturbation of the i-th event, and where $t(j,\xi)$ and $\Delta t(j,\xi)$ are its timing and time perturbation, respectively.

The second type of time perturbation, evaluated on the basis of a *propagation function*, depends only on the type, event-time and time perturbation of another event, the j-th one,

$$\Delta t(i,\xi) = Prop(e(i,\xi), t(i,\xi), e(j,\xi), t(j,\xi), \Delta t(j,\xi)). \tag{2}$$

For each event-type $e(i,\xi)$ either the generation function or the propagation one will be used for the computation of $\Delta t(i,\xi)$, depending mainly on the event-nature $e(i,\xi)$ itself.

The rules used in this paper are those exact under First Order Similarity, ([3, 16]). Qualitatively, two paths are first order similar if their difference is due to exchange between adiacent events: in this case to determine the time perturbation of the i-th event only the information about the two adiacent events are needed, namely the i-th and the$j$-th ones in the (1) and (2).

The event $e(j,\xi)$, appearing in the (1) and (2), can be either an event preceeding the i-th in the nominal path (i.e. $t(j,\xi) < t(i,\xi)$), or an event following it (thus, with $t(j,\xi) > t(i,\xi)$).

As an example of the first situation, consider, in a queueing network, the end of service of two customers which successively visit the same node: any perturbation in the timing of the end of service of the first customer has influence on the time of occurrence of the end of service of

the second one.

The second situation occurs in a production line with finite capacity buffer in each machine. Consider a machine, e.g. M_{n-1}, that after having serviced its piece, sends it to the machine M_n when its buffer B_n is "semi-full" (i.e. with only a place available). If B_n were perturbed and its size reduced by one unit, the emission of that piece from M_{n-1} would be delayed. The duration of the delay of the event $e(i,\xi)$ of type "exit of a piece from machine M_{n-1}" can be determined at the occurrance of the event $e(j,\xi)$ of type "exit of a piece from machine M_n", which make a place available in B_n and allows machine M_{n-1} to resume its service in the perturbed path. Event $e(j,\xi)$ clearly follows $e(i,\xi)$ in the nominal path and it is necessary to determine the time perturbation of $e(i,\xi)$.

The perturbed path is estimated using all the information that has been obtained run-time from the observed nominal path; whenever the time perturbation of an event $e(i,\xi)$ depends on a future event $e(j,\xi)$ (i.e. $t(j,\xi) > t(i,\xi)$), the construction of the perturbed path is suspended until the occurrence of such event $e(j,\xi)$. In other words, if in the functions (1) and (2), $t(j,\xi) > t(i,\xi)$, a sort of "cut-and-paste" scheme is applied between the times $t(i,\xi)$ and $t(j,\xi)$.

This procedure is, in some way, similar to the "cut-and-paste" approach proposed, e.g., in the extended perturbation analysis (see e.g. [3]) and in the path construction method [1, 4, 5]. In the latter "cut-and paste" is required when there is an event that is feasible in the perturbed path but unfeasible in the nominal one.

The control scheme implemented in Section 3, is based on the estimation of the perturbed path under the effects of buffer capacity perturbations in a manufacturing system. The perturbation analysis technique, outlined in the following, will be developed for the special case of "finite"-lenght path construction. It is assumed that the system can be modeled as a queueing network with general service time and limited buffer size for each node, *first come first served* (FCFS) policy for customers in queues and random and not state dependent routing between nodes. The parameters considered are the buffer sizes of each node. In the following a node will be said to be in *full output* if its downstream buffer is full and the node cannot release the finished customer.

The analysis consists of the application to each event of either the generation function or the propagation function, depending on the event. The basic idea of the implemented generation function is the following. A perturbation can be generated in two cases. If a buffer size of a node is augmented, every event of exit of a piece blocked by that node is anticipated, and the full output interval in the nominal path is eliminated. If a buffer size of a node is diminuished, every event of exit of piece from a node preceeding the perturbed one is delayed, and a full output interval is introduced in the perturbed path. In all the other cases the propagation function has to be applied, this function can be found in [16].

3. On-line control strategy

In this section a control scheme, using the path estimation procedure briefly sketched in Section 2, is presented; it is then applied to a manufacturing system that can be modeled as a queueing network with the features that have been described above. Let $\mathcal{N} = \{1, 2, \ldots, M\}$, be the set of the nodes, where M is the number of nodes in the network; let B_n be the buffer size of node n, $n \in \mathcal{N}$; denote with t_k^n, $n \in \mathcal{N}$ and $k \in \mathbf{Z}$ the time of occurrence of the event of type "exit of the k-th customer from node n", assuming that the customers serviced by a node during a whole sample path are numbered increasingly from 1. It is assumed that it is possible to observe a sample path of the system, to which the described analysis is applied.

The proposed control procedure is varying some buffer size somewhere in the system in reaction to unknown changes in the production scenario, in order to keep as high as possible the performance of the system. The sample performance measure considered in this paper is the *sample throughput* of the system, defined as:

$$STP(\xi, k_1, k_2) = \frac{t_{k_2}^M - t_{k_1}^M}{k_2 - k_1} \tag{3}$$

with $k_1 < k_2$, and where $t_{k_2}^M$ and $t_{k_1}^M$ are respectively the time of the exit of the k_2-th customer and of k_1-th customer from node M, during a nominal path, characterized by the variable ξ. To estimate the expected value of (3), called *throughput*, the mean of p samples is evaluated. These samples are obtained from adiacent segments in the same nominal path, i.e.

$$TP(N, p) = \frac{1}{p} \cdot \sum_{m=1}^{p} STP(\xi, N_m, N_{m+1}) \tag{4}$$

where integers N_m, $m = 1, 2, \ldots, p$ are chosen according to $N_{m+1} = N_m + N$, $m = 1, 2, \ldots, p$, with N being a design parameter and $N_1 = \bar{N}$ chosen as a warm-up parameter. The path estimation scheme outlined in Section 2, allows to estimate the sample throughput of the system with some perturbed buffers by only evaluating the time perturbation of the event of "exit of customer k_2 from the M-th node". The parameter perturbation of interest in this section, are simmetrical unity changes

in the buffer sizes of two different nodes, say j and ℓ, i.e.

$$S\tilde{T}P(\xi, k_1, k_2, j, \ell) = \frac{t^M_{k_2} + \Delta t^M_{k_2} - t^M_{k_1}}{k_2 - k_1} \qquad (5)$$

where $\Delta t^M_{k_2}$ depends on the buffer-size perturbations of nodes j, $\Delta B_j = +1$ and of node ℓ, $\Delta B_\ell = -1$, and is computed applying the proposed scheme only to the path segment between $t^M_{k_1}$ and $t^M_{k_2}$ (it has been noticed in [17] that finite perturbation analysis applied to short intervals instead of the whole path has a better accuracy). The expected value of (5) is estimated, calculating the mean over p samples, obtained from the same N_m and N_{m+1} of (4), i.e.

$$\tilde{TP}(N, p, j, \ell) = \frac{1}{p} \cdot \sum_{m=1}^{p} S\tilde{T}P(\xi, N_m, N_{m+1}, j, \ell), \qquad (6)$$

This value is called *estimated perturbed throughput*, with j denoting the node whose buffer is augmented, and ℓ the one whose buffer is diminuished. It is recalled that the estimated time perturbation are reset at zero, at the start of every path sub-interval $[t^M_{N_m}, t^M_{N_{m+1}}]$, with $m = 1, 2, \ldots, p$.

For a given manufacturing system, in general, it can be identified a subset, $\mathcal{S}$, $\mathcal{S} \subseteq \mathcal{N}$, of the *controllable buffers*, i.e. the nodes whose buffers can be varied. A control action, resulting from the proposed strategy, can be denoted with an ordered couple $(j, \ell) \in \mathcal{S} \times \mathcal{S}$, meaning that the buffer size of node j is increased by one, and that of node ℓ is decreased by the same quantity.

The basic steps of the proposed control strategy can be summarized as follow, for given values of design parameters N, $\bar{N}$ and p:

Procedure

Step 1. The throughput of the system is evaluated by observing its evolution, and averaging a finite number of sample throughput values as in (4).

Step 2. By means of the proposed path estimation approach, the sample perturbed throughput is estimated for all the ordered couples $(j, \ell) \in \mathcal{S} \times \mathcal{S}$, using (5). The value of the perturbed throughput is then evaluated averaging a finite number of samples, according to (6).

Step 3. The couple $(\bar{j}, \bar{\ell}) \in \mathcal{S} \times \mathcal{S}$, with the higher value of $\tilde{TP}(N, p, \bar{j}, \bar{\ell})$, is determined. If this value is higher than $TP(N, p)$, an improvement is expected if the control action denoted with $(\bar{j}, \bar{\ell})$ is taken: the action is, then, actually taken. □

Remark 3.1 Perturbation analysis allows to estimate the effect on the performance measure, of changes in some buffers without having to change anything. In other words, it is possible to foresee the results of a control action without actually implementing it.

It is evident, moreover, that the controller has a finite delay, equal to the length of the interval $t^M_{N_{p+1}} - t^M_{N_1}$ on the nominal path, which is necessary to calculate the p samples used to determine $TP(N, p)$ and $\tilde{TP}(N, p, j, \ell)$. This control procedure is able then to react to unknown changes in the production scenario that can be assumed almost constant during this interval. The quantity $N \times p$ can be considered a measure of the rapidity of the controller. □

Remark 3.2 The concept underlying the control rules explained, is similar to that of "improvability", introduced in [15]: a system performance is, in fact, improvable by the action $(\bar{j}, \bar{\ell})$, if an higher value of the performance measure after the application of the control action is estimated. □

Remark 3.3 In most Kanban systems the buffer size of a certain working centre is equal to the number of Kanbans related to this node [18, 19]. Since these latter are simply paper protocols, every action aimed at changing their number can easily be taken. Besides, in these systems it is important (see for instance, [8, 20]) to vary dynamically their number. For this reason, in the following, it is paid a particular attention to these systems, and the buffer size of a node is assumed to be the number of its Kanbans. □

Remark 3.4 In many manufacturing systems, the space dedicated to the storage of the pieces is limited, both for physical limits to that space and for economic reasons (being the capacity of a buffer linked to the *work-in-process* of the system, and therefore to the cost of the production cycle); see, for instance, [19]. For this reason, only simmetrical buffer changes are considered the above procedure. □

4. Simulation results

The proposed control strategy has been applied to the network illustrated in Figure 1, which can be seen as a model of a three part-type manufacturing cell. Nodes 3, 4, 5 and 6 are used to model a working station whose behaviour is dependent on the part-type. The buffer of node 6 can only accumulate the piece currently under service. Nodes 3, 4 and 5 may be seen as preprocessing nodes, each one servicing only one part-type. They have a common storage space, physically limited, and organized into three logical buffers by assigning each node (hence each part-type) a different number of Kanbans. The product mix entering the cell is modeled as the routing probability from node 2 to the nodes 3, 4 and 5. Situation of simultaneous blockings due to full outputs of the nodes 3, 4 and 5 are handled by a *first blocked first released* policy.

The system is subject to unknown changes in its product-mix, modeled by means of unknown changes in the routing probability from node 2 to the three central nodes. The main goal of the controller is to keep as high as possible the system throughput, despite unknown changes in the product-mix.

The considered queueing network is an approximate model of a single-card Kanban cell, with fixed order point of one [21], working different part-types with a first come first served sequencing rule [8]. It is then possible to control the in-process inventory by changing the number of Kanbans of each part-type using the control actions defined in the above procedure. Such an action, in view of the simmetrical buffer size variation, leaves unchanged the total buffering capacity and hence the total storage capacity.

Two interesting results have been found: firstly, the control scheme reacts periodically to periodic changes in the product-mix; secondly, this strategy improves the mean value of the throughput, during a whole production cycle, with respect to its value for the uncontrolled system.

The behaviour of the proposed scheme has been evaluated by means of a simulation model of the queueing network in Figure 1.

A first experiment (Experiment 1) is run with these specifications: node 1 has always a piece in its buffer, and the pieces finished by node 6 are immediately taken away; all the nodes have service time exponentially distributed, with mean 1; buffers 1, 2, 6 have size 1, while the sum of Kanbans of nodes 3, 4 and 5 is 6. The control action can vary the number of Kanbans of these three nodes (i.e. the set of controllable buffers is $\mathcal{S} = [3, 4, 5]$), from a minimum of 1 to a maximum of 4.

The throughput and the estimated perturbed throughput are evaluated on the basis of (4) and (6) by considering 15 samples (i.e. p=15), each one evaluated on 50 pieces (i.e. N=50). The maximum of $\tilde{TP}(50, 15, j, \ell)$ for all $(j, \ell) \in \mathcal{S} \times \mathcal{S}$, is compared with TP(50,15). A control action can be taken every 750=50 $\times$ 15 pieces, i.e. every *control interval* equal to $CI = N \times p$. The control procedure is not applied during the transient period, chosen to last for the first 100 pieces (i.e. $N_1 = \bar{N} = 100$ in the (4) and (6)).

The routing probabilities to nodes 3, 4 and 5, are periodic functions of the number of pieces. Figure 2 illustrates the behaviour of the routing probability from node 2 to node 3 over a period (called *routing period* or simply *period*). During each routing period 30 control actions can be taken: then, in this experiment, every subperiod lasts 22500 pieces. The routing probabilities to nodes 4 and 5 are traslated of one third (called the *routing subperiod* or simply *subperiod*) and two thirds of the routing period, respectively. In every subperiod 10 control actions can be taken.

Every simulation experiment lasts five routing periods, corresponding to 150 control intervals, thus 112700 pieces. The results, averaged over 18 statistically independent simulation runs, are illustrated in Figure 3 and Figure 4. Notice that all the diagrams show the relevant metrics as function of the control intervals, corresponding to $N \times p$ pieces.

Figure 3 shows the average values of the number of Kanbans of node 3, i.e. the size of node 3 buffer, observed at the end of every control interval and taking into account the last control action. The same diagrams for nodes 4 and 5 can be obtained by translating that of node 3 of one third and two thirds of the routing period, respectively. It is evident that the control action is periodic as well: during the first routing subperiod, i.e. during the first 10 control actions (when the routing probability to node 3 assumes its maximum value of 0.9), the controller increases the value of the buffer size of the node 3 up to its maximum value of 4. During the other two routing subperiods the size of buffer 3 is reduced to its minimum value of 1. So, by the proposed strategy, as much stocking space as possible for the part-type mostly worked in that subperiod is provided.

Figure 4 shows the behaviour of the average values of throughput (over 18 independent simulation experiments) observed after every control interval. The thinner grey line represents the value of throughput of the network without control, with buffer sizes of nodes 3, 4 and 5 equal to 2; the thicker one represents the performance of the controlled system (these values are those of TP(50,15), used to determine the control action). It should be emphasized that the *controlled throughput*, i.e. the throughput of the system under the action of the proposed control scheme, is almost periodic with period equal to the routing subperiod.

Some interesting features of the implemented strategy are, moreover, evident. (a) A control action is implemented when the estimated perturbed throughput is higher than the nominal throughput; (b) Figure 3 shows that the control actions are taken only at the end of the first three control intervals in every subperiod; (c) this implies that an improvement is only expected in these three intervals. In addition, Figure 4 shows that the value of the controlled throughput increases during the first three control intervals and remains quite constant during the remaining part of every subperiod. (c') Therefore, the controlled throughput improves only when expected, thus when a control action is taken.

Figure 4 shows that after the first control interval from the beginning of each routing subperiod, the controlled throughput has a value lower than the value of the *uncontrolled throughput* (i.e. the throughput of the network with number of Kanbas not controlled and equal to 2 for nodes 3, 4 and 5), and after two control actions it becomes higher. This is due to the following reason: when the routing probability of one node switches from its lower level to its upper one, the network is unbalanced, i.e. the higher number of Kanbans is assigned to a "low-probability" node. This situation makes the controlled throughput of the network drop. Three control actions are then necessary to re-balance the network (thus to assign 4 Kanbans to the likeliest part-type). In other words, for every subperiod, the first value of the controlled throughput is lower than that of the uncontrolled one because the controlled system is unbalanced, whereas the uncontrolled network has 2 Kanbans for every part-type, independently on the product mix.

Comparing the overall mean of the throughput, it turns out that, after the five routing periods, the controlled network has a throughput mean value of 6,04, against a value of 5,77 for the uncontrolled network. This shows the effectiveness of the control strategy.

Two other experiments have been performed on the same network, subject to the same variation of the product mix as in Figure 2, but with the lenght of the control interval, $CI = N \times p$, reduced.

One experiment (Experiment 2) is performed with a control inteval of CI=150 pieces, evaluating the throughput by averaging p=15 samples, each sample evaluated on N=10 pieces. During each routing subperiod 10 control actions can be taken as in the previous experiment. The overall simulation lenght, of 150 control intervals, is now of 22540 pieces, with $\bar{N}$=20 in the (4) and (6). Figures 5 and 6 show the behaviour of the controlled capacity and of the system throughput averaged over 18 simulation runs.

The other experiment (Experiment 3) is performed with a control interval of CI=70 pieces, obtained averaging p=7 samples of throughput, each one evaluated every N=10 pieces. As in the previous experiment the routing subperiod lasts 10 control intervals as each sample path considered contains 150 control actions, with a total lenght of 10540 pieces worked by the network. The warm-up parameter $\bar{N}$ has been chosen equal to 20. Figures 7 and 8 show the behaviour of the controlled capacity and of the throughput over 18 simulation runs. Notice that in all experiments the simulation lenght is the same in term of number of control actions. Nevertheless, in view of the choice of parameters N and p, the control actions in Experiment 1 are much slower (in time) than the control actions in the other experiments, hence the time duration of a routing period is longer in Experiment 1.

All the features underlined in Experiment 1, are again recognizable in these two other experiments: (a) one of the controlled variable, e.g. the number of Kanbans of node 3, has an almost periodic behaviour; (b) the controlled throughput is increasing during the first control intervals of each subperiod, and has an average value, over the entire path, higher than that of the uncontrolled one (6.30 against 6.02 when $CI = 150$ and 6.26 against 5.99 when $CI = 70$).

The main drawback of a shorter control interval is a larger ripple both in the controlled capacity and in the controlled throughput. On the other hand the controller can react to faster unknown changes in the production scenario.

The three experiments indicate that the proposed control scheme is effective in reacting to unknown changes in the production-mix. The three experiments show that a reduction in the number of samples used to decide on the control action maintains a significant improvement of the network performance, while allowing the system to reacts to unknown, rapid changing, disturbances.

5. Conclusion

In this paper, finite perturbation analysis has used to derive a path estimation procedure for discrete events dynamic systems, applied to the dynamic allocation of buffering capacity in a manufacturing system, problem of great practical interest.

The main contribution consists of an on-line control strategy, applicable to Kanban systems, which reacts to unknown changes in the production scenario (for example, the product mix of a production cell), by varying the number of Kanbans of the different part-type worked by the system.

This strategy is an interesting application of finite perturbation analysis to an on-line control procedure and to the problem of dynamically controlling the number of Kanbans in a manufacturing system. The results presented indicates that other interesting contributions can be brought specializing the guidelines of the proposed procedure to real Kanban systems, and encourage further developments of the proposed approach.

References

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

1993.

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

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

[4] C. Cassandras and S. Strickland, "On-line sensitivity analysis of markov chains," *IEEE Trans. on Automatic Control*, vol. 34, no. 1, pp. 76–86, 1989.

[5] 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.

[6] C. Cassandras, "On-line optimization for a flow control strategy," *IEEE Trans. on Automatic Control*, vol. AC-32, no. 11, pp. 1014–1017, 1987.

[7] Z. Q. Wang, W. Song, and C. Feng, "Full state perturbation analysis of discrete event dynamic systems," *Discrete Event Dynamic Systems: Theory and Applications*, vol. 1, pp. 249– 270, 1992.

[8] B. Berkley, "Effect of buffer capacity and sequencing rules on a single-card Kanban system performance," *Int. J. Prod. Res.*, vol. 31, no. 12, pp. 2875– 2893, 1993.

[9] D. Wang, "Optimization of control strategies and buffer sizes in discrete manufacturing systems," in *12-th IFAC World Congress*, (Sydney, Australia), pp. 58–67, July 1993.

[10] D. Golhar and C. Stamm, "The Just-In-Time philosophy: a literary review," *Int. J. Prod. Res.*, vol. 29, no. 4, pp. 657– 676, 1991.

[11] C. Chu and W. Shih, "Simulation studies," *Int. J. Prod. Res.*, vol. 30, no. 11, pp. 2573– 2586, 1992.

[12] C. Cassandras, M. Abidi, and D. Towsley, "Distributed routing with on-line marginal delay estimation," *IEEE Transaction on Communications*, vol. 38, no. 3, pp. 348– 359, 1990.

[13] M. Caramanis and G. Liberopulos, "Perturbation analysis for the design of flexible manufacturing flow controllers," *Operation Research*, vol. 40, no. 6, pp. 1107–1125, 1992.

[14] N. Xi, F. F. Wu, and S.-M. Lun, "Dynamic bandwidth allocation using infinitesimal perturbation analysis," in *Proc. InfoCom'94*, pp. 383–389, July 1994.

[15] D. Jacobs and S. Meerkov, "System theoretic properties of process of continous improvement in production systems," in *IEEE Int. Conf. on Decision and Control*, (Baltimore, MA), pp. 3323–3327, June 1994.

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

[17] X. Cao, "First order perturbation analysis of a simple multiclass finite source queue," *Performance Evaluation*, vol. 7, pp. 31–41, 1987.

[18] O. Mejabi and G. Wasserman, "Basic concepts of JIT modelling," *Int. J. Prod. Res.*, vol. 30, no. 1, pp. 141–149, 1992.

[19] Y. Gupta and M. Gupta, "A system dynamics model for a multi-stage multi-line JIT-Kanban system," *Int. J. Prod. Res.*, vol. 27, no. 2, pp. 309– 352, 1989.

[20] J. Bard and B. Golany, "Determining the number of Kanbans in a multiproduct, multistage production system," *Int. J. Prod. Res.*, vol. 29, no. 5, pp. 881– 895, 1991.

[21] B. Berkley, "Tandem queues and Kanban controlled lines," *Int. J. Prod. Res.*, vol. 29, no. 10, pp. 2057–2081, 1991.

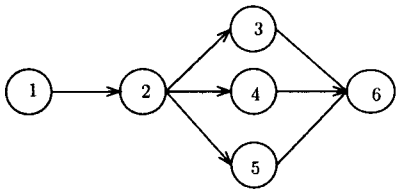

Figure 1: Manufacturing system model.

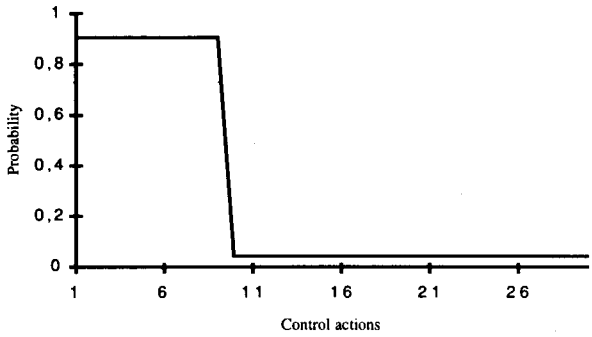

Figure 2: Routing probability from node 2 to node 3.

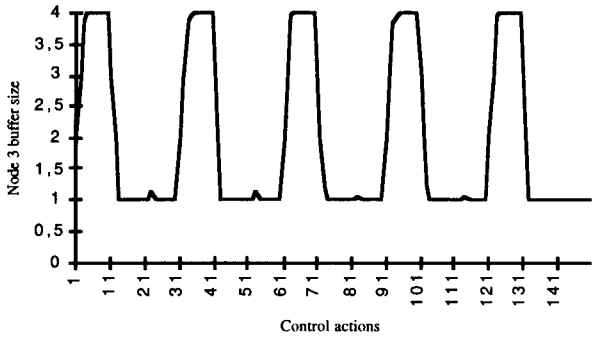

Figure 3: Buffer size of node 3 (Experiment 1).

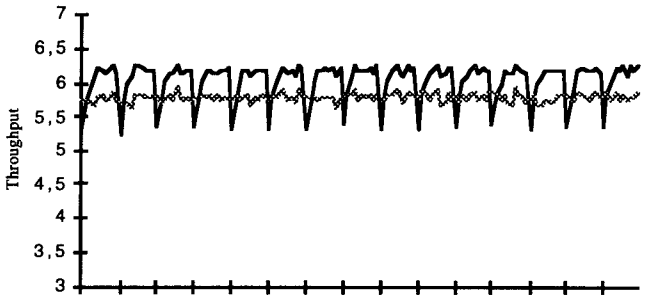

Figure 4: System throughput (Experiment 1).

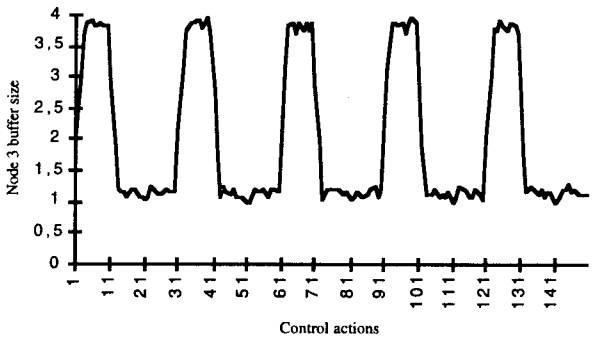

Figure 5: Buffer size of node 3 (Experiment 2).

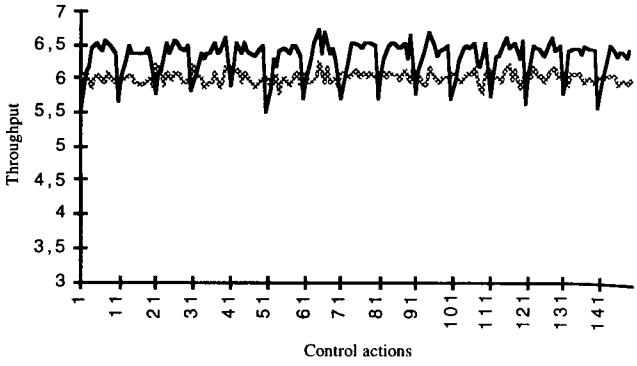

Figure 6: System throughput (Experiment 2).

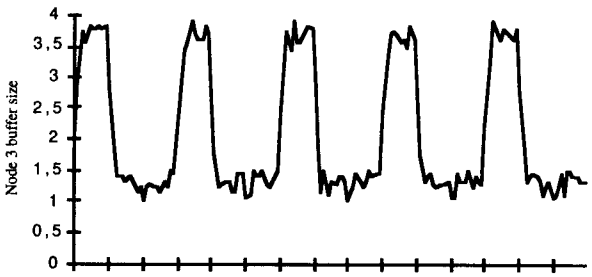

Figure 7: Buffer size of node 3 (Experiment 3).

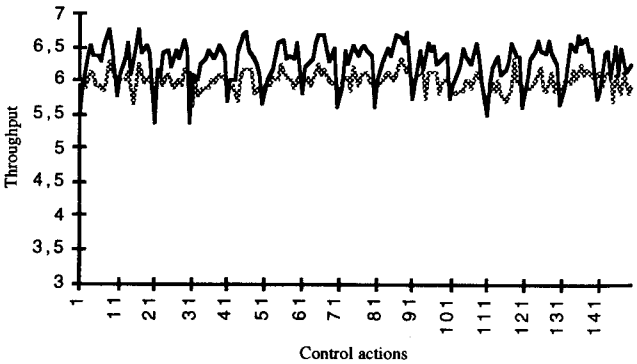

Figure 8: System throughput (Experiment 3).