Optimization Of Kanban-Based Manufacturing Systems

Christos G. Panayiotou [a] Christos G. Cassandras [b,1]

[a] *Department of Electrical and Computer Engineering, University of Massachusetts, Amherst, MA 01003.*

[b] *Department of Manufacturing Engineering, Boston University, Boston, MA 02215.*

Abstract

We develop and analyze an algorithm to maximize the throughput of a serial kanban-based manufacturing system with arbitrary arrival and service process distributions by adjusting the number of kanban allocated to each production stage while maintaining the total work-in-process inventory at any desired level. The optimality properties of the algorithm are proved under a necessary and sufficient "smoothness condition". The algorithm is driven by throughput sensitivities which, in general, can only be estimated along an observed sample path of the system. It is shown that the algorithm converges to the optimal allocation in probability and, under additional mild conditions, almost surely as well. Finally, it is shown that Finite Perturbation Analysis (FPA) techniques can be used to obtain the sensitivity estimates in order to reduce the amount of simulation required in either on-line or off-line simulation-based optimization.

Key words: Manufacturing system, kanban, discrete event system, ordinal optimization, perturbation analysis

1 Introduction

The objectives of the Just-In-Time (JIT) manufacturing approach (see (Sugimori *et al.*, 1977) and (Ashburn, 1986)) are, among others, to reduce the work-in-process inventory and its fluctuations and hence reduce production costs. The main principle of the technique is to produce material only when it is

[1] Corresponding author. Tel: (617) 353-7154, Fax: (617) 353-4830; E-mail: cgc@enga.bu.edu

needed. Its most celebrated component is the so called *kanban* method, which many researchers and analysts have adopted in order to model various types of manufacturing processes. The main idea behind the kanban method is the following. A production line is divided into several stages and at every stage there is a fixed number of tags (or tickets) called *kanban*. An arriving job receives a kanban at the entrance of the stage and maintains possession of it until it exits the stage. If an arriving job does not find an available kanban at the entrance, it is not allowed to enter that stage until a kanban is freed; in this case, the job is forced to wait in the previous stage and becomes *blocked*.

There are several variations of the basic kanban production system (see (Di Mascolo *et al.*, 1991) for an overview) and, over the past years, much work has been devoted to the analysis and performance evaluation of such schemes. One of the main issues associated with the kanban method is the determination of the number of kanban at every stage. It is obvious that in order to achieve a minimum work-in-process inventory, no more than one job should be allowed at every stage. This, however, would severely restrict the achievable throughput. Clearly, therefore, the selection of the number of kanban is closely linked to the usual tradeoff between throughput and work-in-process inventory (or, equivalently, the delay or "lead time" of jobs through the system). Other factors affecting the determination of the number of kanban (including the coefficient of variation in processing times, machine utilization, and the autocorrelation of processing times) were investigated by (Philipoom *et al.*, 1987), who also proposed an empirical methodology for determining the number of kanban. In related work, (Gupta and Gupta, 1989) investigated additional performance measures such as production idle time and shortage of final products. In studying the performance of kanban systems, both analytical models (e.g., (Kimura and Terada, 1981; Mitra and Mitrani, 1988; So and Pinault, 1988)) and simulation (e.g., (Huang *et al.*, 1983; Lulu and Black, 1987; Schroer *et al.*, 1985)) have been used. In the former case, one must resort to certain assumptions regarding the various stochastic processes involved (e.g., modeling demand and service processes at different stages through exponential distributions); however, even in the case of a simple finite Markov chain model, the large state space of such models necessitates the use of several types of approximations. In the case where simulation is used, any kind of parametric analysis of the model requires a large number of simulation runs (one for each parameter setting) to be performed. A comparative overview of many such studies is contained in (Uzsoy and Martin-Vega, 1990).

Advances in the field of Discrete Event Systems (DES) have provided opportunities for the study of kanban systems from different perspectives. For example, (Glasserman and Yao, 1994) have placed these systems in a Generalized Semi-Markov Process (GSMP) framework and have derived, under certain conditions, several structural properties such as monotonicity and concavity of throughput as a function of the number of kanban. Using Perturbation Analysis (PA), (Ho *et al.*, 1979) have also considered the problem of allocating a finite number of buffers to a set of sequential machines in order to maximize the system throughput, a problem equivalent to allocating a given number of kanban over a series of stages in a simple kanban system. Their optimization approach was based on estimating a "pseudogradient" of the throughput with respect to the vector describing a buffer allocation, treating buffer sizes as continuous variables. A similar in spirit approach was used in (Yan *et al.*, 1994), where the actual buffer contents are also treated as continuous variables and a stochastic approximation scheme is invoked to perform the optimization itself. In (Liberatore *et al.*, 1995), the problem considered in (Ho *et al.*, 1979) is revisited for more general topologies and a control scheme for dynamically allocating kanban is proposed. The advantage of PA in this context lies in its ability to estimate the effect of adding/removing one kanban on the system performance based on a single observed sample path (e.g., a single simulation run). It is therefore possible to develop schemes for adjusting the allocation of kanban *on line*, without explicit knowledge of the distributional characteristics of the stochastic processes involved.

In this paper, we consider a kanban system consisting of several stages connected in series. A job arrives at the first stage and is placed in a buffer waiting for service. Once it receives service, if there is a free kanban in the next stage it advances to that stage; otherwise it waits for the next available kanban in the current stage blocking the operation of the corresponding server. Our objective is to allocate a fixed number of kanban (equivalently, buffer slots) to a number of stages so as to maximize the system throughput. Unlike (Ho *et al.*, 1979; Yan *et al.*, 1994), we do not resort to continuous flow models of the system or "pseudogradient" estimation. Instead, we develop a *discrete* optimization scheme maintaining the naturally discrete nature of the controllable parameters, i.e., the numbers of kanban at each stage.

This paper is based on our earlier work described in (Panayiotou and Cassandras, 1996). In (Panayiotou and Cassandras, 1996) we assumed a deterministic model and identified a necessary and sufficient condition under which a specific discrete optimization scheme we derived can yield an allocation that maximizes throughput. Specifically, we assumed that a closed-form expression for the system's performance under any allocation exists, and showed that our scheme delivers the optimal allocation in a fixed number of steps K, where K is the number of *available* kanban to be allocated[2]. In this paper, we extend the approach to stochastic models. Note that in most practical systems, closed-form expressions for performance measures are rarely available, so one is usually forced to use performance estimates obtained through

[2] If K is not fixed, the algorithm can be used on line to incrementally adjust a kanban allocation until a point where throughput and work-in-process inventory can be traded off at desirable levels.

on-line system observation or simulation. Therefore, we have adapted the original optimization scheme to operate in such environments. Furthermore, by exploiting properties of ordinal comparison techniques derived in (Cassandras *et al.*, 1998; Dai, 1996), we show that it converges in probability, and, under some additional technical conditions, almost surely as well, to the optimal allocation. We point out that the results in this paper make no assumptions on the distributions of the interarrival and service times. Further, we note that since the algorithm is driven by ordinal comparisons of finite difference throughput estimates, convergence is fast, as also observed in similar optimization schemes (e.g., see (Cassandras *et al.*, 1998)). The second contribution is the use of a Finite PA (FPA) approach to estimate the finite differences required by our discrete optimization scheme from a single observed sample path. We have chosen to pursue this approach (in contrast to more general concurrent estimation techniques as in (Cassandras and Pan, 1995; Cassandras and Panayiotou, 1996)), since it exploits the structure of the system and the performance measure of interest (throughput). The results of this FPA approach are similar to those of (Ho *et al.*, 1979; Liberatore *et al.*, 1995); they are obtained, however, through the use of a simple recursive max-plus type of equation that formally describes the departure time dynamics at all stages of the kanban system and hence yields a formal characterization of the corresponding departure time perturbation dynamics as well. A byproduct of this analysis is a simple derivation of throughput monotonicity with respect to the number of kanban at any stage.

The paper is organized as follows. In Section 2 we formally define the optimization problem and introduce two conditions on the performance measure of interest. In Section 3 we present the Incremental Optimization algorithm which is designed to work for systems where a closed-form expression of performance as a function of kanban allocations is available. In Section 4 we show that a modified version of the algorithm, adapted for a stochastic environment (i.e., when only noisy performance estimates are available), converges in probability, and, under some additional conditions, almost surely as well, to the optimal allocation. In Section 5 we develop the PA approach used to derive the perturbed departure time dynamics and hence the throughput sensitivities required by our algorithm. Some simulation examples are included in Section 6 and we close with the conclusions from this work in Section 7.

2 Problem Formulation

We consider a manufacturing process modeled as a kanban system consisting of $N+1$ stages in series. The entrance to stage 0 contains an infinite-capacity buffer, i.e. stage 0 has infinite kanban. A job that completes service at any stage $i = 0, \cdots, N-1$ proceeds to the $(i+1)$th stage if there is an available kanban for that stage, otherwise it waits, hence blocking the operation of the corresponding server; stage N is assumed to be connected to an infinite sink. Jobs are processed at all stages on a First-In-First-Out (FIFO) basis and no distinction among job types is made. Let x_i denote the number of kanban allocated to stage i and define the N-dimensional vector $\mathbf{x} = [x_1, ..., x_N]$ to represent a kanban allocation. We will assume that at least one kanban is initially allocated to each of stages $1, \cdots, N$ ($x_i \geq 1$), otherwise the throughput of the system is zero. We will further assume that an upper bound on the work-in-process is given such that $\sum_{i=1}^{N} x_i = K'$. Note that since every stage must have at least one kanban, only $K = K' - N$ are *available* to be allocated to the N stages. Let $J(\mathbf{x})$ be an objective function (typically, the throughput). The problem then is to determine an allocation $\mathbf{x}$ that maximizes $J(\mathbf{x})$ subject to this constraint on the total number of kanban.

We will make use of the following definitions. First, $\mathbf{e}_i = [0, ..., 0, 1, 0, ...0]$ is an N-dimensional vector with all of its elements zero except the ith element which is equal to 1. Second,

$$\Delta J_i(\mathbf{x}) = J(\mathbf{x} + \mathbf{e}_i) - J(\mathbf{x}) \tag{1}$$

is the change in $J(\mathbf{x})$ due to the addition of a new kanban to the ith element of an allocation $\mathbf{x}$. In other words, it is the sensitivity of $J(\mathbf{x})$ with respect to x_i. Finally, let

$$\mathcal{A}_k = \left\{ \mathbf{x} \ : \ \sum_{i=1}^{N} x_i = k + N, \ x_i \geq 1 \right\}, \quad k = 0, 1, \cdots$$

be the set of all possible allocations of k *available* kanban to N stages.

Using the above definitions, the optimization problem is formally stated as:

$$(\mathbf{P1}) \qquad \max_{\mathbf{x} \in \mathcal{A}_K} J(\mathbf{x})$$

In addition, we define the following conditions on $J(\mathbf{x})$:

Smoothness Condition or Condition **(S)**:
If $J(\mathbf{x}^*) \geq J(\mathbf{x})$ for some $\mathbf{x}^* \in \mathcal{A}_k$ and any $\mathbf{x} \in \mathcal{A}_k$, $k = 1, \cdots, K$ then

$$\max_{i=1,...,N} J(\mathbf{x}^* + \mathbf{e}_i) \geq \max_{i=1,...,N} J(\mathbf{x} + \mathbf{e}_i) \tag{2}$$

Uniqueness Condition or Condition **(U)**:
Let $i^* = \arg\max_{i=1,...,N} \{\Delta J_i(\mathbf{x})\}$, then

$$\Delta J_{i^*}(\mathbf{x}) > \Delta J_j(\mathbf{x}), \quad \text{for any } \mathbf{x} \in \mathcal{A}_k, \ k = 1, \cdots, K, \text{ and any } j \neq i^*. \tag{3}$$

Condition **(S)** does not allow an allocation that is not optimal in $\mathcal{A}_k$ to become the only optimal allocation in $\mathcal{A}_{k+1}$ by the addition of one kanban to some stage. When $J(\mathbf{x})$ is the *throughput* of a serial system, this condition has been empirically observed to hold over all cases considered and it reflects the fact that the addition of a single kanban to a stage of a non-optimal allocation will not cause a "non-smooth" jump in the overall throughput of a serial system. It is possible, however, that the presence of branching and merging in more general topologies may violate **(S)**.

Condition **(U)** requires that at every allocation the maximum finite difference as defined in (1) is unique. This is a rather technical condition, as will become clear in the sequel, and it may be relaxed as shown in Section 3.1.

3 Incremental Optimization Algorithm

Problem **(P1)** falls in the class of discrete resource allocation problems. A major difficulty here is due to the fact that performance measures such as throughput are not separable over the number of stages, i.e., one cannot express $J(\mathbf{x})$ as $J(\mathbf{x}) = \sum_{i=1}^{N} J_i(x_i)$, in which case a number of algorithms exist for the solution of **(P1)** (e.g., (Ibaraki and Katoh, 1988; Cassandras and Julka, 1994)). Under conditions **(S)** and **(U)**, however, the following simple incremental allocation process similar to one found in (Ibaraki and Katoh, 1988) provides an optimal allocation of K kanban in K steps.

Define the sequence $\{\mathbf{x}_k\}$, $k = 0, \cdots, K$ such that

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{e}_{i_k^*} \tag{4}$$

where

$$i_k^* = \arg \max_{i=1,\ldots,N} \{\Delta J_i(\mathbf{x}_k)\} \tag{5}$$

and $\mathbf{x}_0 := [1, 1, ..., 1]$. After K steps, $\mathbf{x}_K$ is the optimal solution of **(P1)** as shown in the theorem that follows.

Theorem 1 *For any $k = 0, 1, \cdots$, x_k in (4) yields a solution to problem* **(P1)** *if and only if $J(\mathbf{x})$ satisfies conditions* **(S)** *and* **(U)**.

PROOF. We use induction on $k = 0, 1, \cdots$ and establish the result for any number of kanban k to be allocated over N stages. First, define the following vectors: $\mathbf{x}_k$ is the allocation reached at the kth step in (4); $\mathbf{x}_k^*$ is the solution of **(P1)** over $\mathbf{x} \in \mathcal{A}_k$; and finally $\mathbf{y}_k$ is any allocation in $\mathcal{A}_k$.

For $k = 0$, (5) gives $i_0^* := arg\ \max_{i=1,\ldots,N}\{\Delta J_i(\mathbf{x}_0)\}$. Then, from the definition of $\Delta J_i(\mathbf{x})$ and condition **(U)**, it follows that

$$J(\mathbf{x}_0 + \mathbf{e}_{i_0^*}) > J(\mathbf{x}_0 + \mathbf{e}_{i_0}) \qquad \text{for all } i_0 = 1, ..., N,\ i_0 \neq i_0^* \tag{6}$$

which implies that $\mathbf{x}_1^* = \mathbf{x}_0 + \mathbf{e}_{i_0^*}$. Note that this is true because (5) is obtained from an exhaustive search over the entire space $\mathcal{A}_1$ which includes only N allocations, $\mathbf{x}_0 + \mathbf{e}_i$ for $i = 1, \cdots, N$. Since equation (4) gives $\mathbf{x}_1 = \mathbf{x}_0 + \mathbf{e}_{i_0^*}$, it follows that $\mathbf{x}_1 = \mathbf{x}_1^*$, that is, $\mathbf{x}_1$ is the solution of **(P1)** over $\mathcal{A}_1$.

Now suppose that for some $k \geq 1$ the vector x_k obtained from (4)-(5) yields the optimal allocation, that is

$$J(\mathbf{x}_k) = J(\mathbf{x}_k^*) \geq J(\mathbf{y}_k) \qquad \text{for all } \mathbf{y}_k \in \mathcal{A}_k$$

From (5), again $i_k^* = \arg\max_{i=1,\ldots,N}\{\Delta J_i(\mathbf{x}_k)\}$ (a unique index under **(U)**). It then follows from the definition of $\Delta J_i(\mathbf{x})$ that

$$\begin{aligned} J(\mathbf{x}_k + \mathbf{e}_{i_k^*}) &= J(\mathbf{x}_k) + \Delta J_{i_k^*}(\mathbf{x}_k) \\ &\geq J(\mathbf{x}_k) + \Delta J_{i_k}(\mathbf{x}_k) \\ &= J(\mathbf{x}_k + \mathbf{e}_{i_k}), \qquad\qquad \text{for any } i_k = 1, \cdots, N \end{aligned}$$

Therefore,

$$J(\mathbf{x}_k + \mathbf{e}_{i_k^*}) = \max_{i=1,..,N}\{J(\mathbf{x}_k + \mathbf{e}_i)\} \geq \max_{i=1,..,N}\{J(\mathbf{y}_k + \mathbf{e}_i)\}$$

where the inequality is due to the smoothness condition **(S)**. Hence, $\mathbf{x}_{k+1}^* = \mathbf{x}_k + \mathbf{e}_{i_k^*}$ Finally, note that (4) also gives $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{e}_{i_k^*}$, and therefore, $\mathbf{x}_{k+1} = \mathbf{x}_{k+1}^*$, i.e. $\mathbf{x}_{k+1}$ is the solution of **(P1)** over $\mathcal{A}_{k+1}$.

Conversely, suppose that the algorithm yields the optimal solution for any $K = 1, 2, \cdots$, however it does not satisfy conditions **(S)** and **(U)** for some $k < K$. This implies that there exists an allocation $\mathbf{x}_k^* \in \mathcal{A}_k$ such that $J(\mathbf{x}_k^*) \geq J(\mathbf{y}_k)$ for all $\mathbf{y}_k \in \mathcal{A}_k$ and $\max_{i=1,..,N}\{J(\mathbf{x}_k^* + \mathbf{e}_i)\} < \max_{i=1,..,N}\{J(\mathbf{y}_k + \mathbf{e}_i)\}$. This implies that the algorithm does not yield an optimal allocation over $\mathcal{A}_{k+1}$, which is a contradiction. ■

Remark 1 *If there are K available resources to be allocated to N stages, then the process (4)-(5) requires K steps before it delivers the optimal allocation.*

In contrast, using exhaustive search requires a number of steps which is combinatorially explosive:

$$\begin{pmatrix} K+N-1 \\ K \end{pmatrix} = \frac{(K+N-1)!}{(N-1)!K!} \tag{7}$$

Note that the algorithm defined by (4)-(5) is simple to implement provided the sensitivity information in (5) is available. We shall subsequently refer to this as the *Incremental Optimization* (IO) algorithm.

It is possible to relax the Uniqueness condition **(U)** through a straightforward extension of the IO algorithm as described in the next section.

3.1 Extension of the Incremental Optimization Algorithm

Suppose that the sequence $\{\mathbf{x}_k\}$ in (4) yields $\bar{\mathbf{x}}_k$ after k steps and assume that (5) gives $i_k^* = i = j$, for two indices $i, j \in 1, \cdots, N$. In this case, it is clear that $J(\bar{\mathbf{x}}_k + \mathbf{e}_i) = J(\bar{\mathbf{x}}_k + \mathbf{e}_j)$, but the process has no way of distinguishing between i and j in order to define a unique new state $\mathbf{x}_{k+1}$ given $\mathbf{x}_k = \bar{\mathbf{x}}_k$. Note also that random selection cannot guarantee convergence since it is possible that at the next iteration *only one* of the two allocations (either $\bar{\mathbf{x}}_k + \mathbf{e}_i$ or $\bar{\mathbf{x}}_k + \mathbf{e}_j$) can yield the optimum. Since there is inadequate information to choose between i and j, it is natural to postpone the decision until more information is available. To achieve this we modify the process as described next, by using a recursion on a set of allocations $U_k \in \mathcal{A}_k$. In particular, we define a sequence of sets $\{U_k\}$, $k = 0, \cdots, K$ such that

$$U_{k+1} = \{\mathbf{x}_k + \mathbf{e}_i \mid \Delta J_i(\mathbf{x}_k) = \Delta J_{i_k^*}(\mathbf{x}_k), i = 1, \cdots, N, \mathbf{x}_k \in U_k\} \tag{8}$$

where

$$i_k^* = \arg \max_{\substack{i=1,\ldots,N \\ \mathbf{x}_k \in U_k}} \{\Delta J_i(\mathbf{x}_k)\}. \tag{9}$$

and $U_0 = \{\mathbf{x}_0\}$, $\mathbf{x}_0 = [1, 1, ..., 1]$. After K steps, it is easy to see that any allocation in U_K is the optimal solution to (**P1**). The extra cost incurred by this scheme compared to (4)-(5) involves storing additional information.

As already mentioned, the IO algorithm and its extension above are easy to implement if a closed-form expression for $J(\mathbf{x})$ (and hence $\Delta J_i(\mathbf{x})$) is available for any possible allocation vector $\mathbf{x}$. However, this is seldom the case, since the manufacturing processes we are interested in are generally stochastic DES with arbitrary distributions characterizing the job arrival and the $N+1$ service processes. One is therefore forced to resort to estimates $\hat{J}(\mathbf{x})$ of $J(\mathbf{x})$, obtained either from simulation or on-line observations. This raises an obvious question as to whether the IO algorithm can be appropriately adapted to operate with these estimates and whether it will still converge to the optimal allocation. This issue is investigated in the following section.

4 Stochastic Incremental Optimization Algorithm

In this section we focus our attention to the kanban allocation problem in a stochastic environment. In this case, we assume that the performance measure is in the form of an expectation, $J(\mathbf{x}) = E[\mathcal{L}(\mathbf{x})]$, where $\mathcal{L}(\mathbf{x})$ is a sample function used as the noisy performance estimate. The problem then, is to determine the optimal kanban allocation based on a scheme similar to the IO algorithm defined by (4)-(5) now driven by estimates of $J(\mathbf{x}_k)$. In particular, let $\hat{J}^t(\mathbf{x}_k)$ denote a noisy estimate of $J(\mathbf{x}_k)$ obtained through simulation or on-line observation of the system over an "estimation period" t. Clearly, the IO algorithm (as well as its extension in Section 3.1) can no longer guarantee convergence in such a stochastic setting. For instance, suppose that at the kth step of the allocation process $i_k^* = j$, however, due to noise, we obtain an estimate $\hat{i}_k^* = m \neq j$. In this case, the mth stage will get an additional kanban, whereas it is possible that at the optimal allocation the mth stage has only as many kanban as it had prior to the kth iteration. Since there is no way of reallocating kanban to another stage, the optimal allocation will never be reached.

With this observation in mind, we introduce next a number of modifications to the IO algorithm. First, there should be a mechanism through which kanban erroneously allocated to some stage are reallocated to other stages. Second, it must be possible to progressively improve the performance estimates so as to eliminate the effects of estimation noise. Toward this goal, let $f(l)$ denote the length of the sample path on the lth iteration and let it be such that $\lim_{l\to\infty} f(l) = \infty$. We then define a stochastic process $\{\hat{\mathbf{x}}_{k,l}\}$, $k = 0, \cdots, K$, $l = 1, 2, \cdots$, as follows:

$$\hat{\mathbf{x}}_{k+1,l} = \hat{\mathbf{x}}_{k,l} + \mathbf{e}_{\hat{i}_{k,l}^*} \quad k = 0, \cdots, K-1 \tag{10}$$

for all $l = 1, 2, \cdots$, and every K iterations, i.e. after allocation $\hat{\mathbf{x}}_{K,l}$, the process is reset to

$$\hat{\mathbf{x}}_{0,l+1} = [1, \cdots, 1] \quad l = 1, 2, \cdots \tag{11}$$

where

$$\hat{i}^*_{k,l} = \arg \max_{i=1,\ldots,N} \left\{ \Delta \hat{J}_i^{f(l)}(\hat{\mathbf{x}}_{k,l}) \right\}. \tag{12}$$

We will subsequently refer to the above allocation scheme as the *Stochastic Incremental Optimization* (SIO) algorithm. Regarding the performance estimates $\hat{J}^t(\mathbf{x})$ obtained over an estimation period of length t, we will make the following assumption:

*A1***:** For every $\mathbf{x}$ the estimate $\hat{J}^t(\mathbf{x})$ is ergodic as the sample path length increases in the sense that

$$\lim_{t\to\infty} \hat{J}^t(\mathbf{x}) = J(\mathbf{x}), \ \ a.s. \quad \text{for all } \mathbf{x} \in \mathcal{A}_k, k = 0, \cdots, K$$

We can then establish the following result, the proof of which is very

similar to that of Lemma 4.1 in (Cassandras *et al.*, 1998):

Lemma 1 *Suppose that assumption A1 is satisfied and that* $\lim_{l\to\infty} f(l) = \infty$. *For any allocation* $\mathbf{x} \in \mathcal{A}_k$, $k = 0, \cdots, K$, *if* $\Delta J_i(\mathbf{x}) < \Delta J_j(\mathbf{x})$ *for some* $i, j \in 1, \cdots, N$ *then*

$$\lim_{l\to\infty} \Pr\left[\Delta \hat{J}_i^{f(l)}(\mathbf{x}) \geq \Delta \hat{J}_j^{f(l)}(\mathbf{x})\right] = 0$$

and

$$\lim_{l\to\infty} \Pr\left[\Delta \hat{J}_i^{f(l)}(\mathbf{x}) < \Delta \hat{J}_j^{f(l)}(\mathbf{x})\right] = 1$$

PROOF. Let

$$\hat{y}_l = \Delta \hat{J}_i^{f(l)}(\mathbf{x}) - \Delta \hat{J}_j^{f(l)}(\mathbf{x}), \qquad y = \Delta J_i(\mathbf{x}) - \Delta J_j(\mathbf{x}) < 0.$$

Then, Assumption *A1* and $\lim_{l\to\infty} f(l) = \infty$ guarantee that

$$\lim_{l\to\infty} \hat{y}_l = y, \ \ a.s.$$

Since a.s. convergence implies convergence in probability, we know that, for every $\epsilon > 0$,

$$\lim_{l\to\infty} \Pr[|\hat{y}_l - y| \geq \epsilon] = 0.$$

Setting $\epsilon = -y > 0$, we obtain

$$\lim_{l\to\infty} \Pr[|\hat{y}_l - y| \geq -y] = 0. \tag{13}$$

Finally, since $\Pr[\hat{y}_l \geq 0] \leq \Pr[|\hat{y}_l - y| \geq -y]$, it immediately follows from (13) that

$$\lim_{l\to\infty} \Pr[\hat{y}_l \geq 0] = 0$$

which is the statement of the lemma. ■

Theorem 2 *For any performance measure $J(\mathbf{x})$ that satisfies conditions* **(S)** *and* **(U)**, *$\{\hat{\mathbf{x}}_{K,l}\}$ converges in probability to the global optimal allocation, as $l \to \infty$.*

PROOF. Let $\mathbf{y}_k,\ k = 1, \cdots, K$ denote the allocations that the IO process in (4) would visit if $J(\mathbf{x})$ were known exactly. Clearly, $\mathbf{y}_K$ is the optimal allocation due to Theorem 1. We proceed by determining the probability that (10)-(12) will yield $\mathbf{y}_K$ for some l:

$$\Pr[\hat{\mathbf{x}}_{K,l} = \mathbf{y}_K] = \Pr[\hat{i}^*_{K-1,l} = i^*_{K-1}|\hat{\mathbf{x}}_{K-1,l} = \mathbf{y}_{K-1}]\Pr[\hat{\mathbf{x}}_{K-1,l} = \mathbf{y}_{K-1,l}]$$

where $\hat{i}^*_{K-1,l}$ and i^*_{K-1} are defined in (12) and (5) respectively. Further conditioning, we get

$$\Pr[\hat{\mathbf{x}}_{K,l} = \mathbf{y}_K] = \left\{\prod_{k=1}^{K-1} \Pr[\hat{i}^*_{k,l} = i^*_k|\hat{\mathbf{x}}_{k,l} = \mathbf{y}_k]\right\}\Pr[\hat{\mathbf{x}}_{0,l} = \mathbf{y}_0] \tag{14}$$

Next, take any term of the product

$$\begin{aligned}
\Pr\left[\hat{i}^*_{k,l} = i^*_k|\hat{\mathbf{x}}_{k,l} = \mathbf{y}_k\right] &= \Pr\left[\Delta\hat{J}^{f(l)}_{i^*_{k,l}}(\mathbf{x}) > \max_{\substack{j=1,\cdots,N\\ j\neq i^*_{k,l}}}\left\{\Delta\hat{J}^{f(l)}_j(\mathbf{x})\right\}|\hat{\mathbf{x}}_{k,l} = \mathbf{y}_k\right] \\
&= 1 - \Pr\left[\Delta\hat{J}^{f(l)}_{i^*_{k,l}}(\mathbf{x}) \leq \max_{\substack{j=1,\cdots,N\\ j\neq i^*_{k,l}}}\left\{\Delta\hat{J}^{f(l)}_j(\mathbf{x})\right\}|\hat{\mathbf{x}}_{k,l} = \mathbf{y}_k\right] \\
&= 1 - \Pr\left[\bigcup_{\substack{j=1\\ j\neq i^*_{k,l}}}^{N} \Delta\hat{J}^{f(l)}_{i^*_{k,l}}(\mathbf{x}) \leq \Delta\hat{J}^{f(l)}_j(\mathbf{x})|\hat{\mathbf{x}}_{k,l} = \mathbf{y}_k\right]
\end{aligned}$$

$$\geq 1 - \sum_{\substack{j=1 \\ j \neq i^*_{k,l}}}^{N} \Pr\left[\Delta \hat{J}^{f(l)}_{i^*_{k,l}}(\mathbf{x}) \leq \Delta \hat{J}^{f(l)}_{j}(\mathbf{x}) | \hat{\mathbf{x}}_{k,l} = \mathbf{y}_k\right] \quad (15)$$

Since $\lim_{l\to\infty} f(l) = \infty$, and since $\Delta J_{i^*_{k,l}}(\mathbf{x}) > \Delta J_j(\mathbf{x}), j \neq i^*_{k,l}$ all terms in the summation go to 0 due to Lemma 1 and therefore, all terms $\Pr[\hat{i}^*_{k,l} = i^*_k | \hat{\mathbf{x}}_{k,l} = \mathbf{y}_k]$ approach 1 as $l \to \infty$. Moreover, by (11) we have $\Pr[\hat{\mathbf{x}}_{0,l} = \mathbf{y}_0] = 1$, where $\mathbf{y}_0 = [1, \cdots, 1]$. It follows that $\lim_{l\to\infty} \Pr[\hat{\mathbf{x}}_{K,l} = \mathbf{y}_K] = 1$ and the theorem is proved. ■

Remark 2 *The convergence rate of the algorithm depends on the difference $\Delta_j(\mathbf{x}) - \Delta_i(\mathbf{x})$. When this difference is large, then the limit of the probabilities described in Lemma 1 converges faster. Therefore, the algorithm quickly converges to a set of allocations that exhibit near optimal performance. Subsequently, it may oscillate between allocations in this set and eventually settles down to the optimal one.*

4.1 Stronger Convergence Results

Under some additional mild conditions and by properly selecting the "observation interval" $t = f(l)$, it is possible to show that the process (10)-(12) converges to the optimal allocation almost surely. To show this, first recall a result from (Dai, 1996).

Lemma 2 *Suppose that $\{\hat{x}_t, t \geq 0\}$ is a stochastic process satisfying (a) $\lim_{t\to\infty} \hat{x}_t = x$, a.s.; (b) $\lim_{t\to\infty} E[\hat{x}_t] = x$; (c) $Var[\hat{x}_t] = O(\frac{1}{t})$. If $x > 0$, then*

$$\Pr[\hat{x}_t \leq 0] = O(\frac{1}{t}).$$

The assumptions in Lemma 2 are very mild and almost always satisfied in the simulation or direct sample path observation of DES. Lemma 2 establishes the rate of convergence for comparing $\hat{x}_t$ against 0. Using this result, we can prove the following lemma which will be needed for our main result.

Lemma 3 *Assume that, for every i, the estimate $\hat{J}^t_i(\mathbf{x})$ satisfies the assumptions of Lemma 2. Then, for any $i, j, i \neq j$,*

$$\Pr[\Delta \hat{J}^t_i(\mathbf{x}) \geq \Delta \hat{J}^t_j(\mathbf{x})] = O(\frac{1}{t})$$

and

$$\Pr[\Delta\hat{J}_i^t(\mathbf{x}) < \Delta\hat{J}_j^t(\mathbf{x})] = 1 - O(\frac{1}{t})$$

provided that $\Delta J_i(\mathbf{x}) < \Delta J_j(\mathbf{x})$.

PROOF. Let $\hat{x}_k = \Delta\hat{J}_j^t(\mathbf{x}) - \Delta\hat{J}_i^t(\mathbf{x})$ and $x = \Delta J_j(\mathbf{x}) - \Delta J_i(\mathbf{x})$. Then, $\{\hat{x}_k\}$ satisfies the assumptions of Lemma 2 and $x > 0$. Thus the conclusion holds. ■

Theorem 3 *For any performance measure* $J(\mathbf{x})$ *that satisfies conditions* **(S)** *and* **(U)** *and the conditions of Lemma 3, if the observation interval* $f(l) \geq l^{1+c}$ *for some constant* $c > 0$*, the process* $\{\hat{\mathbf{x}}_{k,l}\}$ *converges to the global optimal allocation almost surely.*

PROOF. From Lemma 3 we get that

$$\Pr\left[\Delta\hat{J}_{i_{k,l}^*}^{f(l)}(\mathbf{x}) \leq \Delta\hat{J}_j^{f(l)}(\mathbf{x}) | \hat{\mathbf{x}}_{k,l} = \mathbf{y}_k\right] = O(\frac{1}{f(l)}) = O(\frac{1}{l^{1+c}})$$

Also from (15) we get that the

$$\Pr[\hat{x}_{k,l} \notin \mathcal{X}_k^* | \hat{\mathbf{x}}_{k,l} = \mathbf{y}_k] = O(\frac{1}{l^{1+c}}) \tag{16}$$

where $\mathcal{X}_k^*$ is the set of all allocations that exhibit optimal performance from the set $\mathcal{A}_k$. Clearly,

$$\sum_{l=1}^{\infty} \frac{1}{l^{1+c}} < \infty \tag{17}$$

Hence, using the Borel-Cantelli Lemma (see pp. 255-256 of (Shiryayev, 1979)) we conclude that $\hat{x}_{k,l}$ converges to the optimal allocation almost surely. ■

If the Uniqueness condition **(U)** is not satisfied, then there is no guarantee that the SIO algorithm will converge to the optimal in any sense. In this case, one can proceed in a way similar to the set-iterative process in Section 3.1. For example, one can include in the set U_k all the allocations whose estimated performance lies within some distance δ from the observed maximum. However, we will not elaborate on this issue any further and concentrate instead on the crucial problem of obtaining estimates of the finite differences $\Delta J_i(\mathbf{x})$, $i = 1, \cdots, N$.

Note that the algorithm SIO cannot be directly used on-line because allowing the system allocation to reset according to (11) will significantly degrade the system's performance, and it will take K steps until the performance reaches the optimal level again. This is clearly inefficient. Nevertheless, it is possible to use SIO on-line if one adopts ideas from concurrent estimation (Cassandras and Panayiotou, 1996). Specifically, one can implement a simulator of the system under any allocation, but rather than using a random number generator to obtain the various event lifetimes, observe them directly from the actual system. We can then use SIO with the simulated system to allocate all K resources and then update the allocation of the actual system. Note that if one resorts to simulation, for either an on-line or off-line implementation, in order to obtain the estimates $\Delta \hat{J}_i^{f(l)}(\hat{\mathbf{x}}_{k,l})$ in (12) for *all* $i = 1, \cdots, N$, at least $N+1$ simulations are required before a single step of the SIO algorithm can take place. This motivates the use of PA techniques for obtaining all these from a *single* simulation (or observation of a sample path of the actual system). Finally, it is worth pointing out that if this approach is used for on-line control, the system of interest may contain features such as machine failures and repairs, preventive maintenance, and setup times; this is because the estimates obtained are dependent only on observable events and event times and not on any specific distribution characteristics for the stochastic processes involved.

5 Perturbation Analysis for Throughput Sensitivity Estimation

We will henceforth concentrate on throughput as the objective function $J(\mathbf{x})$ to be maximized. In this section, we will employ PA in order to estimate the finite differences $\Delta J_i(\mathbf{x})$ defined in the previous section and required by the SIO algorithm as seen in (12). Our goal, therefore, is to estimate the effect of adding a kanban to any one of stages $1, \cdots, N$ in the system operating under an allocation $\mathbf{x} \in \mathcal{A}_k$ while observing a sample path of that system and without actually making any such change to it.

5.1 Notation and Definitions

We begin by establishing some basic notation and defining quantities we will use in our analysis. First, for any x, let $[x]^+ \equiv \max\{0, x\}$. The pair (k, n) will be used to denote the kth job in the nth stage. Associated with such a job are

Z_k^n : the service time of (k, n).

C_k^n : the service completion time of (k,n) at stage n.
D_k^n : the departure time of (k,n) from stage n; if no blocking occurs, then $D_k^n = C_k^n$.

We also define

$$I_k^n \equiv D_k^{n-1} - D_{k-1}^n \equiv -W_k^n \tag{18}$$

Observe that when $I_k^n > 0$, this quantity is the length of an idle period at stage n that starts with the departure of $(k-1,n)$ and ends with the arrival of (k,n) at time D_k^{n-1}. Conversely, if $W_k^n = -I_k^n > 0$, this is the waiting time of (k,n) who can only begin processing at time $D_{k-1}^n > D_k^{n-1}$. Similarly, we define

$$B_k^n \equiv D_{k-x_{n+1}}^{n+1} - C_k^n \tag{19}$$

which, if $B_k^n > 0$, provides the length of a blocking period for the job (k,n) completing service at time C_k^n. Finally,

$$Q_k^n \equiv D_k^n - D_{k-1}^n \ = \ Z_k^n + [I_k^n]^+ + [B_k^n]^+ \tag{20}$$

so that Q_k^n represents the interdeparture time of $(k-1,n)$ and (k,n) at stage n.

For our purposes, a *perturbed* sample path is one that would have resulted if the exact same nominal sample path had been reproduced under an allocation with one kanban added at some stage. To distinguish between quantities pertaining to the nominal path and their counterparts on a perturbed path we will use a superscript p as follows: if the number of kanban allocated to stage n is x_n in the nominal path, then ${}^p x_n$ denotes the number of kanban in the perturbed path. Similar notation applies to other quantities such as D_k^n, etc. With this in mind, we define the indicator function

$$\mathbf{1}[n+1] \ = \ \mathbf{1}[\,{}^p x_{n+1} = x_{n+1} + 1] \ = \ \begin{cases} 1 \text{ if } {}^p x_{n+1} = x_{n+1} + 1 \\ 0 \text{ if } {}^p x_{n+1} = x_{n+1} \end{cases}$$

to identify the downstream stage to any stage n where an additional kanban would have been added in a perturbed path. We also define

$$\Delta D_k^n \equiv D_k^n - {}^p D_k^n \tag{21}$$

to be the departure time perturbation for (k,n) due to the addition of a kanban to the nominal allocation.

Finally, we will find useful the following quantity, defined as the *relative perturbation* in departure times for two jobs (k_1, n_1) and (k_2, n_2):

$$\Delta^{(k_1,n_1)}_{(k_2,n_2)} \equiv \Delta D^{n_1}_{k_1} - \Delta D^{n_2}_{k_2} \tag{22}$$

5.2 Derivation of Departure Time Perturbation Dynamics

We begin with the simple observation that the departure time D^n_k satisfies the following Lindley-type recursive equation:

$$D^n_k \;=\; max\left\{D^{n-1}_k + Z^n_k,\; D^n_{k-1} + Z^n_k,\; D^{n+1}_{k-x_{n+1}}\right\} \tag{23}$$

There are three cases captured in this equation:

(1) The departure of (k, n) was activated by the departure of $(k, n-1)$. This corresponds to the case where (k, n) starts a new busy period at stage n and, upon completion of service, it is not blocked by the downstream stage $n+1$. Thus, $D^n_k = D^{n-1}_k + Z^n_k$ and from the definitions (18) and (19) it is easy to see that

$$D^n_k = D^{n-1}_k + Z^n_k \Longleftrightarrow W^n_k \le 0,\; B^n_k \le 0 \tag{24}$$

(2) The departure of (k, n) was activated by the departure of $(k-1, n)$. This corresponds to the case where (k, n) belongs to an ongoing busy period (hence, experiencing some waiting in queue before receiving service) and it is not blocked by the downstream server $n+1$. Thus, $D^n_k = D^n_{k-1} + Z^n_k$ and from (18) and (19) it is once again easy to check that

$$D^n_k = D^n_{k-1} + Z^n_k \Longleftrightarrow W^n_k \ge 0,\; B^n_k \le 0 \tag{25}$$

(3) The departure of (k, n) was activated by the departure of $(k-x_{n+1}, n+1)$. This corresponds to the case where (k, n) is blocked and must remain at the nth stage after service completion[3] . In this case, $D^n_k = D^{n+1}_{k-x_{n+1}}$ and from (19) it is easy to check that

$$D^n_k = D^{n+1}_{k-x_{n+1}} \Longleftrightarrow B^n_k \ge 0 \tag{26}$$

Next consider the perturbation ΔD^n_k defined by (21). By applying (23) to both the nominal and perturbed paths, there are 9 distinct cases possible.

Case 1. *Nominal Sample Path:* (k, n) starts a new busy period and is not blocked, i.e. $W^n_k \le 0$, $B^n_k \le 0$.

[3] Actually, this case combines two subcases, one where (k, n) starts a new busy period and leaves stage n after being blocked for some time, and another where (k, n) belongs to an ongoing busy period and leaves stage n after being blocked.

Perturbed Sample Path: (k,n) starts a new busy period and is not blocked, i.e. ${}^pW_k^n \leq 0$ and ${}^pB_k^n \leq 0$.
Applying (24) for both nominal and perturbed sample paths, we get:

$$\Delta D_k^n = D_k^{n-1} + Z_k^n - ({}^pD_k^{n-1} + Z_k^n) \;=\; D_k^{n-1} - {}^pD_k^{n-1} \;=\; \Delta D_k^{n-1} \qquad (27)$$

Case 2. *Nominal Sample Path:* (k,n) starts a new busy period and is not blocked, i.e. $W_k^n \leq 0$ and $B_k^n \leq 0$.
Perturbed Sample Path: (k,n) waits and is not blocked, i.e. ${}^pW_k^n > 0$ and ${}^pB_k^n \leq 0$.
Applying (24) for the nominal sample path and (25) for the perturbed sample path, we get:

$$\begin{aligned}\Delta D_k^n &= D_k^{n-1} + Z_k^n - ({}^pD_{k-1}^n + Z_k^n)\\ &= D_k^{n-1} + D_{k-1}^n - {}^pD_{k-1}^n - D_{k-1}^n\\ &= \Delta D_{k-1}^n + D_k^{n-1} - D_{k-1}^n\\ &= \Delta D_{k-1}^n + I_k^n\end{aligned}$$

where (18) was used in the last step. Note that $I_k^n \geq 0$ since $W_k^n \leq 0$ by assumption. Adding and subtracting ΔD_k^{n-1} and using (22) allows us to rewrite this equation in the following form (which will prove more convenient later on):

$$\Delta D_k^n \;=\; \Delta D_k^{n-1} - \left[\Delta_{(k-1,n)}^{(k,n-1)} - I_k^n\right] \qquad (28)$$

Case 3. *Nominal Sample Path:* (k,n) starts a new busy period and is not blocked, i.e. $W_k^n \leq 0$ and $B_k^n \leq 0$.
Perturbed Sample Path: (k,n) is blocked, i.e. ${}^pB_k^n > 0$.
Using (26) for the perturbed path and the definition of ΔD_k^n,

$$\begin{aligned}\Delta D_k^n &= D_k^n - {}^pD_{k-x_{n+1}-\mathbf{1}[n+1]}^{n+1}\\ &= D_k^n + D_{k-x_{n+1}-\mathbf{1}[n+1]}^{n+1} - {}^pD_{k-x_{n+1}-\mathbf{1}[n+1]}^{n+1} - D_{k-x_{n+1}-\mathbf{1}[n+1]}^{n+1}\\ &= \Delta D_{k-x_{n+1}-\mathbf{1}[n+1]}^{n+1} + D_k^n - D_{k-x_{n+1}-\mathbf{1}[n+1]}^{n+1}\end{aligned}$$

Using (19),(20) and the fact that $D_k^n = C_k^n$ we get, if $\mathbf{1}[n+1] = 0$,

$$\Delta D_k^n = \Delta D_{k-x_{n+1}}^{n+1} - B_k^n$$

and, if $\mathbf{1}[n+1] = 1$,

$$\begin{aligned}\Delta D_k^n &= \Delta D_{k-x_{n+1}-1}^{n+1} + D_k^n - D_{k-x_{n+1}}^{n+1} + D_{k-x_{n+1}}^{n+1} - D_{k-x_{n+1}-1}^{n+1}\\ &= \Delta D_{k-x_{n+1}-1}^{n+1} - B_k^n + Q_{k-x_{n+1}}^{n+1}\end{aligned}$$

Again add and subtract ΔD_k^{n-1} to obtain

$$\Delta D_k^n \;=\; \Delta D_k^{n-1} - \left[\Delta_{(k-x_{n+1}-1[n+1],n+1)}^{(k,n-1)} + B_k^n - Q_{k-x_{n+1}}^{n+1} \cdot \mathbf{1}[n+1]\right] (29)$$

For the other six remaining cases, expressions for ΔD_k^n can be derived in a similar way. We omit the details and provide only the final equations:

Case 4. *Nominal Sample Path:* (k,n) waits and is not blocked, i.e. $W_k^n > 0$ and $B_k^n \leq 0$.
Perturbed Sample Path: (k,n) starts a new busy period and is not blocked, i.e. ${}^pW_k^n \leq 0$ and ${}^pB_k^n \leq 0$.

$$\Delta D_k^n \;=\; \Delta D_{k-1}^n - \left[\Delta_{(k,n-1)}^{(k-1,n)} - W_k^n\right] \tag{30}$$

Case 5. *Nominal Sample Path:* (k,n) waits and is not blocked, i.e. $W_k^n > 0$ and $B_k^n \leq 0$.
Perturbed Sample Path: (k,n) waits and is not blocked, i.e. ${}^pW_k^n > 0$ and ${}^pB_k^n \leq 0$.

$$\Delta D_k^n \;=\; \Delta D_{k-1}^n \tag{31}$$

Case 6. *Nominal Sample Path:* (k,n) waits and is not blocked, i.e. $W_k^n > 0$ and $B_k^n \leq 0$.
Perturbed Sample Path: (k,n) is blocked, i.e. ${}^pB_k^n > 0$.

$$\Delta D_k^n \;=\; \Delta D_{k-1}^n - \left[\Delta_{(k-x_{n+1}-1[n+1],n+1)}^{(k-1,n)} + B_k^n - Q_{k-x_{n+1}}^{n+1} \cdot \mathbf{1}[n+1]\right] \tag{32}$$

Case 7. *Nominal Sample Path:* (k,n) is blocked, i.e. $B_k^n > 0$.
Perturbed Sample Path: (k,n) starts a new busy period and is not blocked, i.e. ${}^pW_k^n \leq 0$ and ${}^pB_k^n \leq 0$.

$$\Delta D_k^n \;=\; \Delta D_{k-x_{n+1}}^{n+1} - \left[\Delta_{(k,n-1)}^{(k-x_{n+1},n+1)} - B_k^n - [W_k^n]^+\right] \tag{33}$$

Case 8. *Nominal Sample Path:* (k,n) is blocked, i.e. $B_k^n > 0$.
Perturbed Sample Path: (k,n) waits and is not blocked, i.e. ${}^pW_k^n > 0$ and ${}^pB_k^n \leq 0$.

$$\Delta D_k^n \;=\; \Delta D_{k-x_{n+1}}^{n+1} - \left[\Delta_{(k-1,n)}^{(k-x_{n+1},n+1)} - B_k^n - [I_k^n]^+\right] \tag{34}$$

Case 9. *Nominal Sample Path:* (k,n) is blocked, i.e. $B_k^n > 0$.
Perturbed Sample Path: (k,n) is blocked, i.e. ${}^pB_k^n > 0$.

$$\Delta D_k^n \;=\; \Delta D_{k-x_{n+1}}^{n+1} - \left[\Delta_{(k-x_{n+1}-1,n+1)}^{(k-x_{n+1},n+1)} - Q_{k-x_{n+1}}^{n+1}\right] \cdot \mathbf{1}[n+1] \tag{35}$$

Remark 3 *Notice that if we substitute equation (22) in all equations (27) to (35) and take into consideration the conditions on B_k^n and W_k^n, it is easily seen that $\Delta D_k^n \geq 0$ (provided initial conditions are such that $\Delta D_k^n = 0$ for any $k, n \leq 0$). This provides an immediate proof of the monotonicity of the throughput with respect to the number of kanban at each stage.*

Equations (27) to (35), provide the departure time perturbation dynamics, assuming the values of ${}^pW_k^n$ and ${}^pB_k^n$ are known. Our goal, however, is to derive a recursive equation for ΔD_k^n completely dependent on information observable along the nominal sample path. The following three theorems provide such recursions.

Theorem 4 *If $W_k^n \le 0$ and $B_k^n \le 0$, then:*

$$\Delta D_k^n = \Delta D_k^{n-1} - \max\Big\{0,\ \Delta_{(k-1,n)}^{(k,n-1)} - I_k^n,\ \Delta_{(k-x_{n+1}-\mathbf{1}[n+1],n+1)}^{(k,n-1)} + B_k^n - Q_{k-x_{n+1}}^{n+1}\cdot \mathbf{1}[n+1]\Big\} \tag{36}$$

PROOF. First, we show that the last two terms in the max bracket above can be expressed in terms of ${}^pW_k^n$ and ${}^pB_k^n$ alone:

$$\begin{aligned}\Delta_{(k-1,n)}^{(k,n-1)} - I_k^n &= \Delta D_k^{n-1} - \Delta D_{k-1}^n - I_k^n\\ &= D_k^{n-1} - {}^pD_k^{n-1} - D_{k-1}^n + {}^pD_{k-1}^n - D_k^{n-1} + D_{k-1}^n\\ &= {}^pD_{k-1}^n - {}^pD_k^{n-1}\\ &= {}^pW_k^n\end{aligned} \tag{37}$$

$$\begin{aligned}\Delta_{(k-x_{n+1}-\mathbf{1}[n+1],n+1)}^{(k,n-1)} &+ B_k^n - Q_{k-x_{n+1}}^{n+1}\cdot\mathbf{1}[n+1] =\\ &= \Delta D_k^{n-1} - \Delta D_{k-x_{n+1}-\mathbf{1}[n+1]}^{n+1} + B_k^n - Q_{k-x_{n+1}}^{n+1}\cdot\mathbf{1}[n+1]\\ &= D_k^{n-1} - {}^pD_k^{n-1} - D_{k-x_{n+1}-\mathbf{1}[n+1]}^{n+1} + {}^pD_{k-x_{n+1}-\mathbf{1}[n+1]}^{n+1} +\\ &\qquad + D_{k-x_{n+1}}^{n+1} - D_k^n - (D_{k-x_{n+1}}^{n+1} - D_{k-x_{n+1}-1}^{n+1})\cdot\mathbf{1}[n+1]\\ &= {}^pD_{k-x_{n+1}-\mathbf{1}[n+1]}^{n+1} - {}^pD_k^{n-1} - Z_k^n\\ &= {}^pD_{k-x_{n+1}-\mathbf{1}[n+1]}^{n+1} - ({}^pC_k^n - [{}^pW_k^n]^+)\\ &= {}^pB_k^n + [{}^pW_k^n]^+\end{aligned} \tag{38}$$

Therefore, equation (36) is equivalent to:

$$\Delta D_k^n = \Delta D_k^{n-1} - \max\{0, {}^pW_k^n, {}^pB_k^n + [{}^pW_k^n]^+\} \tag{39}$$

We can then consider the following three possible cases:

(1) If ${}^pW_k^n \le 0$ and ${}^pB_k^n \le 0$, then **Case 1** examined earlier applies and equation (27) gives $\Delta D_k^n = \Delta D_k^{n-1}$, which is precisely (39) since $\max\{0, {}^pW_k^n, {}^pB_k^n + [{}^pW_k^n]^+\} = 0$.

(2) If ${}^pW^n_k > 0$ and ${}^pB^n_k \leq 0$, then **Case 2** applies and equation (28) holds, which is again (39) since

$$\max\{0, {}^pW^n_k, {}^pB^n_k + [{}^pW^n_k]^+\} = {}^pW^n_k = \Delta^{(k,n-1)}_{(k-1,n)} - I^n_k$$

(3) If ${}^pB^n_k > 0$, then **Case 3** applies and (29) holds, which is the same as (39) since

$$\begin{aligned}\max\{0, {}^pW^n_k, {}^pB^n_k + [{}^pW^n_k]^+\} &= {}^pB^n_k + [{}^pW^n_k]^+ \\ &= \Delta^{(k,n-1)}_{(k-x_{n+1}-\mathbf{1}[n+1],n+1)} \\ &\qquad + B^n_k - Q^{n+1}_{k-x_{n+1}} \cdot \mathbf{1}[n+1]\end{aligned}$$

■

The proofs of the next two theorems are similar to that of Theorem 4 and they are omitted.

Theorem 5 *If $W^n_k > 0$ and $B^n_k \leq 0$, then:*

$$\begin{aligned}\Delta D^n_k = \Delta D^n_{k-1} - \max\Big\{&0,\ \Delta^{(k-1,n)}_{(k,n-1)} - W^n_k, \\ &\Delta^{(k-1,n)}_{(k-x_{n+1}-\mathbf{1}[n+1],n+1)} + B^n_k - Q^{n+1}_{k-x_{n+1}} \cdot \mathbf{1}[n+1]\Big\}\end{aligned} \tag{40}$$

Theorem 6 *If $B^n_k > 0$, then:*

$$\begin{aligned}\Delta D^n_k = \Delta D^{n+1}_{k-x_{n+1}} - \max\Big\{&\Delta^{(k-x_{n+1},n+1)}_{(k,n-1)} - B^n_k - [W^n_k]^+, \\ &\Delta^{(k-x_{n+1},n+1)}_{(k-1,n)} - B^n_k - [I^n_k]^+, \Big(\Delta^{(k-x_{n+1},n+1)}_{(k-x_{n+1}-1,n+1)} - Q^{n+1}_{k-x_{n+1}}\Big) \cdot \mathbf{1}[n+1]\Big\}\end{aligned} \tag{41}$$

In summary, a single recursive expression for ΔD^n_k is obtained by combining the three theorems above. The following lemma asserts that all information involved in the evaluation of ΔD^n_k at time D^n_k is contained in the information available from the nominal sample path by that time (formally, it is part of the σ-field generated by the history of the underlying stochastic process up to time D^n_k).

Lemma 4 *All information required to evaluate the perturbation ΔD^n_k at time D^n_k is contained in the history of the underlying process up to that time.*

PROOF. From Theorems 4-6 and recalling definition (22), the following quantities are involved in the evaluation of ΔD^n_k: ΔD^{n-1}_k, ΔD^n_{k-1}, $\Delta D^{n+1}_{k-x_{n+1}}$,

$\Delta D_{k-x_{n+1}-1}^{n+1}$, I_k^n, W_k^n, B_k^n, and $Q_{k-x_{n+1}}^{n+1}$. The first four quantities are evaluated at times D_k^{n-1}, D_{k-1}^n, $D_{k-x_{n+1}}^{n+1}$, and $D_{k-x_{n+1}-1}^{n+1}$ respectively. Because of the serial topology and FIFO nature of the model, it is obvious that $D_k^n > D_k^{n-1}$, $D_k^n > D_{k-1}^n$. In addition, when the departure of (k,n) occurs at time D_k^n, the departure of $(k-x_{n+1}, n+1)$ must have already occurred, otherwise the former event would not be feasible: (k,n) would have been blocked at time C_k^n and remained blocked until $D_{k-x_{n+1}}^{n+1}$. Therefore, $D_k^n \geq D_{k-x_{n+1}}^{n+1}$. Clearly, then $D_k^n > D_{k-x_{n+1}-1}^{n+1}$ also holds. Finally, the definitions of I_k^n, W_k^n, B_k^n, and $Q_{k-x_{n+1}}^{n+1}$ involve the same departure times above. ■

5.3 Perturbation Analysis Algorithm

Theorems 4-6 directly lead to the following algorithm for evaluating all departure time perturbations.

Algorithm (PA1):

(1) **Initialize**: $\Delta D_k^n = 0$ for all $k, n \leq 0$.

(2) **At the departure of** (k,n):

(a) If (k,n) did *NOT* wait and was *NOT* blocked ($W_k^n \leq 0$, $B_k^n \leq 0$), then

$$\Delta D_k^n = \Delta D_k^{n-1} - \max\Big\{0,\ \Delta_{(k-1,n)}^{(k,n-1)} - I_k^n,\ \Delta_{(k-x_{n+1}-\mathbf{1}[n+1],n+1)}^{(k,n-1)} + B_k^n - Q_{k-x_{n+1}}^{n+1} \cdot \mathbf{1}[n+1]\Big\}$$

(b) If (k,n) waited but was *NOT* blocked ($W_k^n > 0$, $B_k^n \leq 0$), then

$$\Delta D_k^n = \Delta D_{k-1}^n - \max\Big\{0,\ \Delta_{(k,n-1)}^{(k-1,n)} - W_k^n,\ \Delta_{(k-x_{n+1}-\mathbf{1}[n+1],n+1)}^{(k-1,n)} + B_k^n - Q_{k-x_{n+1}}^{n+1} \cdot \mathbf{1}[n+1]\Big\}$$

(c) If (k,n) was blocked ($B_k^n > 0$), then

$$\Delta D_k^n = \Delta D_{k-x_{n+1}}^{n+1} - \max\Big\{\Delta_{(k,n-1)}^{(k-x_{n+1},n+1)} - B_k^n - [W_k^n]^+,\ \Delta_{(k-1,n)}^{(k-x_{n+1},n+1)} - B_k^n - [I_k^n]^+,\ \Big(\Delta_{(k-x_{n+1}-1,n+1)}^{(k-x_{n+1},n+1)} - Q_{k-x_{n+1}}^{n+1}\Big) \cdot \mathbf{1}[n+1]\Big\}$$

This algorithm subsumes the FPA results presented in (Ho *et al.*, 1979), where perturbations were associated with stages rather than individual jobs. The basic perturbation generation and propagation rules also discussed in (Ho *et al.*, 1979; Liberatore *et al.*, 1995) are evident in our algorithm. Perturbation generation, for example, can arise only in **Cases 7-8** examined earlier. One can check that the first possible perturbation is generated by either one of these cases and results in $\Delta D_k^n = B_k^n > 0$.

Note that at the last stage, blocking is not possible, therefore all terms that involve blocking above are ignored at stage N. Throughput is defined as the job departure rate from stage N, and can be estimated by $\hat{T}_k = k/D_k^N$. Therefore, ΔD_k^N immediately yields an estimate of the throughput sensitivity $\Delta J(\mathbf{x})$, $\mathbf{x} \in \mathcal{A}_K$, over k jobs, where $\mathbf{x}$ is the kanban allocation of the observed nominal sample path. Note that as long as all processes are stationary, then $\hat{T}_k$ is an unbiased estimate of the throughput which implies that the assumptions of Section 4 hold. We should point out that the same finite difference estimates may be obtained through concurrent simulation techniques (e.g., see (Cassandras and Panayiotou, 1996)) which are well-suited for constructing sample paths of a DES over a range of discrete parameter values (in this case, kanban allocations). We have selected the FPA approach above, however, because it takes advantage of the special structure of the system of interest.

6 Simulation Examples

In this section, we present application of **(PA1)** and the SIO algorithm to some simulated kanban-based production systems. A system as described in the opening paragraph of Section 2 was simulated for various values of K, N and several initial allocations $\mathbf{x}$.

6.1 Application of (PA1)

From a single simulation, **(PA1)** was used to evaluate the performance (throughput) of systems with allocations $\mathbf{x} + \mathbf{e}_i$, for any $i = 1, \cdots, N$ under the same realization of the arrival and all $N + 1$ service processes. In order to verify the correctness of the results, brute-force simulation was used to actually obtain the throughput under all $\mathbf{x} + \mathbf{e}_i$, $i = 1, \cdots, N$ allocations for values of $N = 4, 5, 6$, which correspond to systems with $5, 6$ and 7 queues in series respectively. This was also used to evaluate the efficiency of the FPA algorithm compared to brute-force simulation. Letting T denote the CPU time of a typical simulation run, then $2T$ is the CPU time required to evaluate one sensitivity by brute-force simulation. Letting $T + \tau$ denote the CPU time of the same nominal run when the FPA algorithm is incorporated to evaluate the same one sensitivity, a measure of the efficiency of **(PA1)** is given by τ/T, the *overhead* imposed by FPA on the nominal run. For a model with a Poisson arrival process and exponential service times, typical overhead values obtained were around 40%; not surprisingly, these are much larger than those of common infinitesimal PA algorithms known to have overheads often less than 1%. When the model is modified to mixed hyperexponential/Erlang service times, however, the overhead is reduced to less than 10%, depending on the order of the model used. The reason for this improvement is that the CPU time required to generate a hyperexponential or Erlang random variate is substantially higher than that of an exponential; since the FPA algorithm uses the same variates for all the sensitivities estimated from one sample path, the benefit of eliminating random variate generation increases with the complexity of the distributional models involved.

6.2 Application of the SIO algorithm.

In order to verify that the SIO algorithm yields an optimal allocation, since no closed form solution exists for the models considered, we simulated exhaustively all possible allocations to obtain the optimal one, and compared the results to the allocation that was delivered by the algorithm.

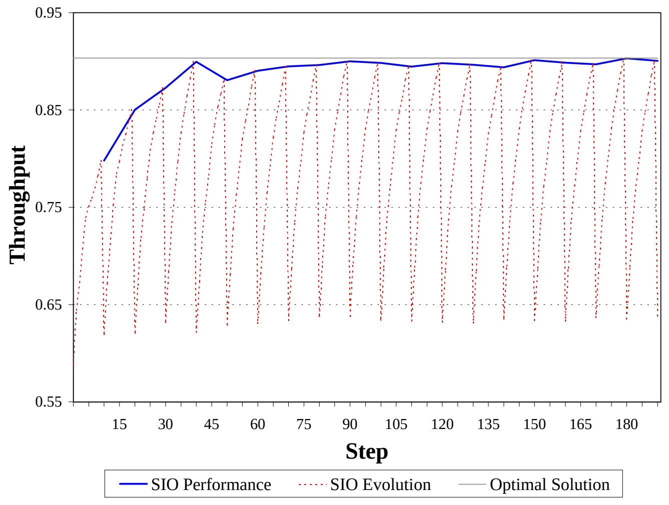

Fig. 1. Evolution of the SIO algorithm

Figure 1 shows the evolution of the algorithm for a system with five stages in series ($N = 4$). In this figure, the horizontal axis is given in term of steps, where a "step" represents the interval between the allocation of an additional resource through (10) and (11). The arrival process is Poisson with rate $\lambda = 1.0$ and the service processes are all exponential with rates $\mu_1 = 2.0$, $\mu_2 = 1.5$, $\mu_3 = 1.3$, $\mu_4 = 1.2$, and $\mu_5 = 1.1$. Initially, we obtain performance estimates every 100 departures ($f(l) = 100$ departures) and every time we reset $\hat{\mathbf{x}}_{k,l}$ we increase the observation interval by another 100 departures. Through exhaustive search, it was found that the optimal allocation is $[1, 3, 4, 5]$ with throughput 0.9033. As seen in Figure 1, the SIO algorithm yields near-optimal allocations within the first four or five iterations[4] (SIO performance curve). It is also worth reporting some additional results not evident from Figure 1. Specifically, the algorithm delivered allocations which were among the top 10% designs even at the very first iteration when the observation interval was limited to only 100 departures. After the first 10 iterations, (observation intervals greater than 1000 departures) the allocations obtained were among the top 1% designs, and after the 20th iteration the SIO algorithm consistently picked the top design $[1, 3, 4, 5]$. Finally, notice the saw-tooth shape of the evolution of the SIO algorithm curve which reflects the incremental nature of the algorithm and the resetting of (11).

In addition, the results of the exhaustive search process can be used to produce a histogram of the throughput over all possible allocations (designs). For the parameters of the example considered above, the histogram is shown in Figure 2. Our experience is that, depending on the number of stages and the service time distributions involved, it is sometimes the case that many allocations provide optimal performance, and sometimes the case that very few provide optimal or near-optimal performance with poor performance for a large number of allocations. For the example under consideration, note that only 3 out of the possible 220 allocations exhibit performance very close to the optimal. Furthermore, this figure indicates that the majority of allocations exhibit performance that is not more than 20% worse than the optimal. Since a priori knowledge of the form of such response surfaces is generally unavailable, the use of the SIO algorithm provides an attractive approach to kanban allocation.

7 Conclusions

We have developed an algorithm for adjusting the number of kanban in a serial production system in order to maximize throughput while maintaining a desirable low Work-In-Process inventory level. The algorithm yields an optimal kanban allocation under the "smoothness condition" introduced in Section 2. Since the algorithm is driven by finite differences capturing how throughput changes as an additional kanban is allocated to a stage of the system, perturbation analysis is well-suited to estimate all these quantities from a *single sample path*. In Section 4, we provided a formal derivation of the event time perturbation dynamics for this system, which extends the perturbation propagation rules first proposed in (Ho *et al.*, 1979) and readily establishes simple structural properties such as throughput monotonicity with respect to the number of kanban. Since the FPA algorithm is based exclusively on observed

[4] An iteration corresponds to K steps

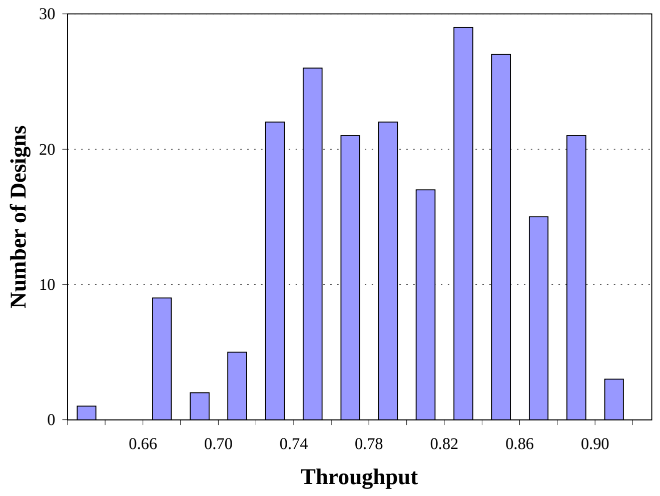

Fig. 2. Performance of all possible allocations when $N = 4$, $\lambda = 1.0$, $\mu_1 = 2.0$, $\mu_2 = 1.5$, $\mu_3 = 1.3$, $\mu_4 = 1.2$, and $\mu_5 = 1.1$

sample path data, it applies to any system with arbitrary arrival and service process distributions.

Extensions to systems of more general topologies may violate the "smoothness condition", but may still yield near-optimal allocations with little computational effort. Finally, alternative techniques, such as concurrent estimation (see (Cassandras and Panayiotou, 1996)), for estimating the finite differences above, and generalizations to systems with multiple products, each with its own kanban allocation, are the subject of ongoing research.

Acknowledgements

This work was supported in part by NSF under Grants EEC-9527422 and ACI-9873339, by AFOSR under contracts F49620-95-1-0131 and F49620-98-1-03871 and by the Air Force Rome Laboratory under contract F30602-97-C-0125

References

Ashburn, A. (1986). Toyota famous OhNo system, American machinist. In Y. Monden (Ed.). 'Applying Just in Time: The American/Japanese Experience'. IIE Press.

Cassandras, C. G. & C. G. Panayiotou (1996). Concurrent sample path analysis of discrete event systems. In 'Proceedings of the 35th IEEE Conference on Decision and Control'. pp. 3332–3337.

Cassandras, C. G. & J. Pan (1995). Some new approaches to discrete stochastic optimization problems. In 'Proceedings of the 34th IEEE Conference on Decision and Control'. pp. 573–578.

Cassandras, C. G. & V. Julka (1994). Descent algorithms for discrete resource allocation problems. In 'Proceedings of the 33rd IEEE Conference on Decision and Control'. pp. 2639–2644.

Cassandras, C. G., L. Dai & C. G. Panayiotou (1998). 'Ordinal optimization for a class of deterministic and stochastic discrete resource allocation problems'. *IEEE Transactions on Automatic Control* **43**(7), 881–900.

Dai, L. (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., Y. Frein, Y. Dallery & R. David (1991). 'A unified modeling of kanban systems using petri nets'. *Intl. Journal of Flexible Manufacturing Systems* **3**, 275–307.

Glasserman, P. & David Yao (1994). A GSMP framework for the analysis of production lines. In D. Yao (Ed.). 'Stochastic Modeling and Analysis of Manufacturing Systems'. Springer-Verlag. chapter 4, pp. 133–188.

Gupta, Y. & M.C. Gupta (1989). 'A system dynamics model for a multistage multiline dual-card JIT-kanban system'. *Intl. Journal of Production Research* **27**(2), 309–352.

Ho, Y., M.A. Eyler & T.T. Chien (1979). 'A gradient technique for general buffer storage design in production line'. *Intl. Journal of Production Research* **17**(6), 557–580.

Huang, P., L.P. Rees & Taylor B.W. (1983). 'A simulation analysis of the japanese Just-In-Time technique (with kanbans) for a multiline multistage production system'. *Decision Sciences* **14**, 326–344.

Ibaraki, T. & N. Katoh (1988). *Resource Allocation Problems, Algorithmic Approaches*. MIT Press.

Kimura, O. & H. Terada (1981). 'Design and analysis of pull system, a method of multi-stage production control'. *Intl. Journal of Production Research* **19**(3), 241–253.

Liberatore, G., S. Nicosia & P. Valigi (1995). Path construction techniques for dynamic control of kanban systems. In 'Proceedings of 34th IEEE Conference on Decision and Control'. pp. 2610–2611.

Lulu, M. & J.T. Black (1987). 'Effect of process unreliability on integrated manufacturing/production systems'. *Journal of Manufacturing Systems* **6**(1), 15–22.

Mitra, D. & I. Mitrani (1988). Analysis of a novel discipline for cell coordination in production lines. Technical report. AT&T Laboratories.

Panayiotou, C. G. & C. G. Cassandras (1996). Optimization of kanban-based production systems. In 'Proceedings of *WODES '96*'. pp. 39–44.

Philipoom, P., L.P. Rees, B.W. Taylor & P.Y. Huang (1987). 'An investigation of the factors influencing the number of kanbans required in the implementation of the JIT technique with kanbans'. *Intl. Journal of Production Research* **25**(3), 457–472.

Schroer, B., J.T. Black & S.X. Zhang (1985). 'Just-In-Time (JIT), with kanban, manufacturing system simulation on a microcomputer'. *Simulation* **45**(2), 62–70.

Shiryayev, A. (1979). *Probability*. Springer-Verlag. New York.

So, K. C. & S. C. Pinault (1988). 'Allocating buffer storage in a pull system'. *Intl. Journal of Production Research* **26**(12), 1959–1980.

Sugimori, Y., K. Kusunoki, F. Cho & S. Uchikawa (1977). 'Toyota production system materialization of Just-In-Time and research-for-human systems'. *Intl. Journal of Production Research* **15**(6), 553–564.

Uzsoy, R. & L. A. Martin-Vega (1990). 'Modeling kanban-based demand-pull systems: a survey and critique'. *Manufacturing Review*.

Yan, H., X.Y. Zhou & G. Yin (1994). Finding optimal number of kanbans in a manufacturing system via stochastic approximation and perturbation analysis. In 'Proceedings of 11th Intl. Conference on Analysis and Optimization of Systems'. pp. 572–578.