Context-Based Statistical Process Control: A Monitoring Procedure for State-Dependent Processes

Irad BEN-GAL

Department of Industrial Engineering
Tel-Aviv University
Ramat-Aviv, Tel-Aviv 69978, Israel

Gail MORAG

Department of Industrial Engineering
Tel-Aviv University
Ramat-Aviv, Tel-Aviv 69978, Israel

Armin SHMILOVICI

Department of Information Systems Engineering
Ben-Gurion University
Beer-Sheva 84105, Israel

Most statistical process control (SPC) methods are not suitable for monitoring nonlinear and state-dependent processes. This article introduces the context-based SPC (CSPC) methodology for state-dependent data generated by a finite-memory source. The key idea of the CSPC is to monitor the statistical attributes of a process by comparing two context trees at any monitoring period of time. The first is a *reference tree* that represents the "in control" reference behavior of the process; the second is a *monitored tree*, generated periodically from a sample of sequenced observations, that represents the behavior of the process at that period. The Kullback–Leibler (KL) statistic is used to measure the relative "distance" between these two trees, and an analytic distribution of this statistic is derived. Monitoring the KL statistic indicates whether there has been any significant change in the process that requires intervention. An example of buffer-level monitoring in a production system demonstrates the viability of the new method with respect to conventional methods.

KEY WORDS: Context tree; Kullback–Leibler statistic; Statistical process control.

1. INTRODUCTION

1.1 Statistical Process Control Methods: Overview and Taxonomy

Since the introduction of statistical process control (SPC)—the Shewhart chart—extensive research has been performed to adapt it to various industrial settings. Early SPC methods were based on two critical assumptions: (1) there exists an a priori knowledge of the underlying distribution (often, observations are assumed to be normally distributed), and (2) the observations are independent and identically distributed (iid). In practice, these assumptions are frequently violated in various industrial processes. In this article we present a novel SPC method that is not based on those assumptions.

An extensive literature review leads us to categorize current SPC methods by two major criteria:

1. Methods for *independent* data, where observations are not interrelated versus methods for *dependent* data
2. Methods that are *model-specific*, requiring a priori assumptions on the process characteristics, usually defined by an underlying analytical distribution or a closed-form expression, such as autoregressive integrated moving average (ARIMA), versus methods that are termed *model-generic*, which try to estimate the underlying model with minimum a priori assumptions.

Figure 1 presents a taxonomy of SPC methods. In the following paragraphs, we discuss some of the SPC methods presented in the literature and explore their relationship to the method presented in this article.

The information-theoretic process control (ITPC) is an independent data–based and model-generic SPC method proposed by Alwan, Ebrahimi, and Soofi (1998). It uses information theory principles, such as maximum entropy, subject to constraints derived from the process moments. It provides a theoretical justification for the traditional Gaussian assumption and suggests a unified control chart, as opposed to traditional SPC, which requires separate charts for each moment.

Traditional SPC methods, such as Shewhart, cumulative sum (CUSUM), and exponential weighted moving average (EWMA), are for independent data and are model-specific. It is important to note that these methods are implemented extensively in industry, although the independence assumptions are frequently violated in practice: automated testing devices increase the sampling frequency and introduce autocorrelation into the data. Moreover, implementation of feedback control devices on the shop floor tends to create structured dynamics in certain system variables (see Boardman and Boardman 1990; Ben-Gal and Singer 2001). Applying traditional SPC to such interrelated processes increases the frequency of false alarms and shortens the "in-control" average run length (ARL), compared with uncorrelated observations.

Most model-specific methods for dependent data are based on time series. The underlying principle of these methods is as follows: Find a time series model that can best capture the autocorrelation process, use that model to filter the data, then apply

TECHNOMETRICS, NOVEMBER 2003, VOL. 45, NO. 4
DOI 10.1198/004017003000000122

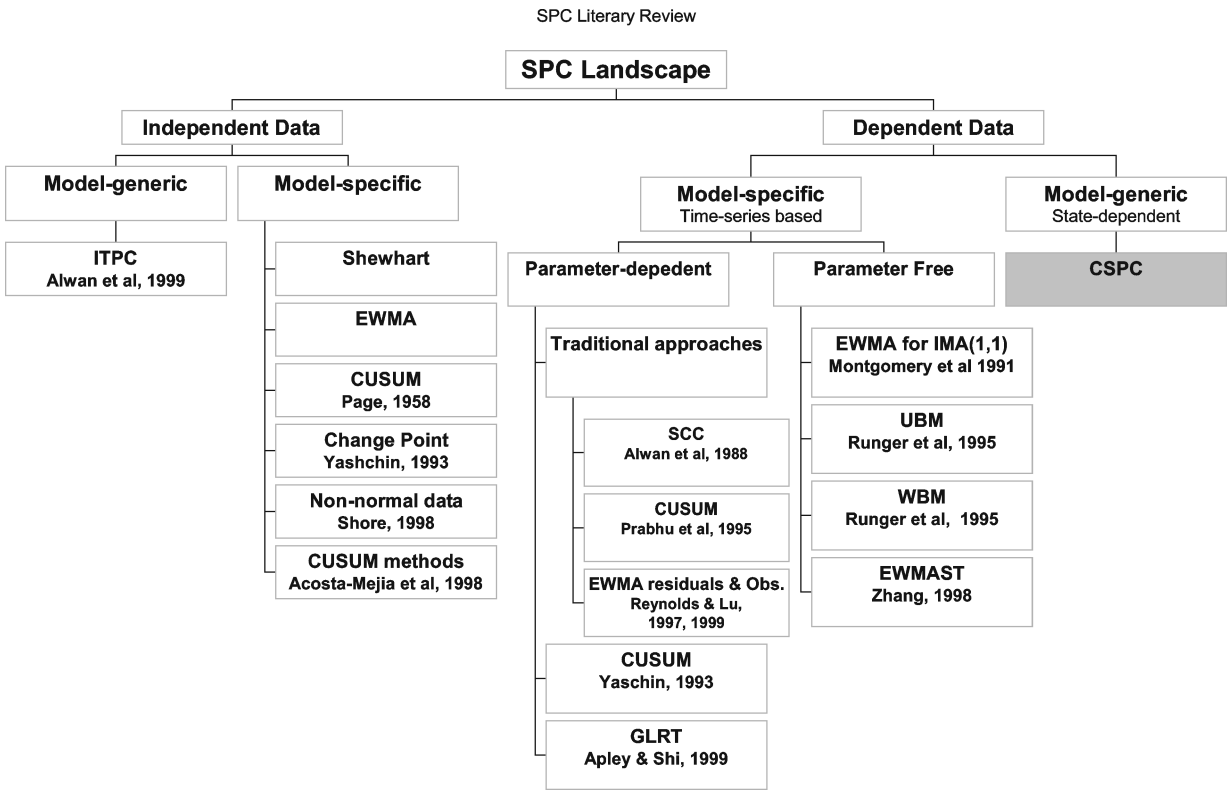


Figure 1. Taxonomy of SPC Methods.

traditional SPC schemes to the stream of residuals. In particular, the ARIMA family of models is widely applied for the estimation and filtering of process autocorrelation. Under certain assumptions, the residuals of the ARIMA model are independent and approximately normally distributed, and traditional SPC can be applied. Furthermore, it is commonly conceived that ARIMA models [mostly the simple ones, such as AR(1)] can effectively describe a wide variety of industry processes (see Box and Jenkins 1976; Apley and Shi 1999).

Model-specific methods for autocorrelated data can be further partitioned to *parameter-dependent* methods that require explicit estimation of the model parameters, and to *parameter-free* methods, where the model parameters are only implicitly derived, if at all. Several parameter-dependent methods were proposed over the years for autocorrelated data. Alwan and Roberts (1988) proposed the special cause chart (SCC), in which the Shewhart method is applied to the stream of residuals. They showed that the SCC has major advantages over Shewhart with respect to mean shifts. The SCC deficiency lies in the need to explicitly estimate all of the ARIMA parameters. Moreover, the method performs poorly for a large positive autocorrelation, because the mean shift tends to stabilize rather quickly to a steady-state value, and thus the shift is poorly manifested on the residuals (see Wardell, Moskowitz, and Plante 1994; Harris and Ross 1991). Runger, Willemain, and Prabhu (1995) implemented traditional SPC for autocorrelated data using CUSUM methods. Lu and Reynolds (1997, 1999) extended the method by using the EWMA method with a small difference; their model had a random error added to the ARIMA model. All of these models are based on a priori knowledge of the data source and require an explicit parameter estimation. Runger and Willemain (1995) demonstrated that for certain autocorrelated processes, the use of traditional SPC yields improved performance compared with ARIMA-based methods. The generalized likelihood ratio test (GLRT) method proposed by Apley and Shi (1999) takes advantage of the residual transient dynamics in the ARIMA model when a mean shift is introduced. The generalized likelihood ratio was applied to the filtered residuals. The method was compared to the Shewhart, CUSUM, and EWMA methods for autocorrelated data, inferring that the choice of the adequate time series–based SPC method highly depends on the specific process characteristics. Moreover, Apley and Shi (1999) and Runger and Willemain (1995) emphasized that modeling errors of ARIMA parameters have a strong impact on the performance (e.g., the ARL) of parameter-dependent SPC methods for autocorrelated data. If the process can accurately be defined by an ARIMA time series, the parameter-dependent SPC methods are superior in comparison to nonparameteric methods, because they allow efficient statistical analyses. When this is not the case, the effort of estimating the time series parameters is impractical. This conclusion, among other reasons, triggered the development of parameter-free methods.

A parameter-free model was proposed by Montgomery and Mastrangelo (1991) as an approximation procedure based on EWMA. They suggested using the EWMA statistic as a one-step-ahead prediction value for the IMA$(1, 1)$ model. Their underlying assumption was that even if the process is better described by another member of the ARIMA family, the IMA$(1, 1)$ model is a good enough approximation. Zhang

(1998), however, compared several SPC methods and demonstrated that Montgomery's approximation performed poorly. He proposed using the EWMA statistic for stationary processes, but to adjust the process variance according to the autocorrelation effects. Runger and Willemain (1995, 1996) discussed the weighted batch mean (WBM) and the unified batch mean (UBM) methods. The WBM method assigns weights for the observations mean and defines the batch size so that the autocorrelation among batches reduces to 0. In the UBM method, the batch size is defined (with unified weights) so that the autocorrelation remains under a certain level. Runger and Willemain demonstrated that weights estimated from the ARIMA model do not guarantee a performance improvement and that it is cost-effective to apply the simple UBM method. In general, parameter-free methods do not require explicit ARIMA modeling. However, they are all based on the implicit assumption that the time series model is adequate to describe the process. Although this can be true in some industrial environments, such an approach cannot capture nonlinear process dynamics that depend on the state in which the system operates, for example, processes that are described by hidden Markov models (HMMs) (Elliott, Lakhdaraggoun, and Moore 1995).

This article presents the *context-based SPC* (CSPC) methodology for state-dependent data. This is a novel SPC method characterized as model-generic and based on the *context tree* that was first proposed by Rissanen (1983) for data-compression purposes, and later for prediction and identification (Weinberger, Rissanen, and Feder 1995). The context tree provides a compact model of the sequence of observations even for complex, nonlinear processes, such as HMMs of higher order. The construction algorithm of the context tree generates the minimal tree that fits a given string of data and estimates its parameters.

The key idea of the CSPC is to monitor the statistical attributes of a process, by comparing two context trees at any monitoring period of time. The first of these trees is a *reference tree* that represents the "in control" reference behavior of the process; the second is a *monitored tree*, generated periodically from a sample of sequenced observations, that represents the behavior of the process at that period. The Kullback–Leibler (KL) statistic (Kullback 1959) is used to measure the relative "distance" between these two trees, and an analytic distribution of this statistic is derived. Monitoring the KL statistic indicates whether there has been any significant change in the process that requires intervention.

1.2 Motivation and Some Potential Applications

The proposed CSPC has several appealing characteristics. First, it "learns" the process data dependence and its underlying distribution without assuming a priori information, and hence it is model-generic (nonparametric). This is an important advantage over most traditional SPC schemes, particularly when monitoring processes whose underlying model is unknown. Second, the method extends the current limited scope of SPC applications to nonlinear state-dependent processes. Later we show that even special SPC methods that were designed to handle correlated data fail to monitor a nonlinear process. Third, the CSPC allows a convenient monitoring of discrete process variables. Finally, it uses a single control chart for monitoring the process. The single chart can be further decomposed if "in-depth" analysis is required for the source of deviation from control limits.

The CSPC method may be applied to various processes, as long as they can be approximated by different arrangements of symbols and the relationship between the symbols can be given a statistical expression. However, the method in its current form is based on two constraints: (1) it requires a relatively large amount of data, and (2) it is limited to handle discrete measures over a finite alphabet. Albeit, there are many areas in which these limitations do not apply. Three examples of such potential areas are briefly described.

Image Monitoring. Rissanen (1983), in his original article, proposed applying the context tree to model two-dimensional images, where symbol values reflect the pixels' color (or gray level). Such an idea can be used in a wide area of applications related to image monitoring. One of these is automatic screening for patterns in the textile industry or in the printing industry to identify features of interest or anomalies. The context tree can be used to continuously compare images from running fabric or new prints and detect changes that indicate an anomaly in the coloring system or wear of the print mechanism.

Biological Screening. DNA sequences consist of four bases (nucleotides)—adenine, cytosine, guanine, and thiamine—forming a discrete and final alphabet. Provided that groups of DNA sequences sharing a common functionality or structure can be identified, they may be used as a training database to construct context tree models. The context tree–based models can be applied, using a SPC approach, to identify whether a new sequence belongs to that group or has a similar feature or structure that might apply a certain relation to that group. Examples for DNA sequences that are known to share common statistical properties are acting regulatory sequences, encoding sequences for amino acids that construct proteins, intron sequences that are transcribed but not yet translated, and promoters, which are defined in general as regions proximal to the transcription-start site of genes transcribed by RNA polymerase (Ohler and Niemann 2001). For example, Figure 2 presents the score statistics of two *Escterichia coli* sequences of DNA-spaced reading frames. The upper series represent promoter sequences, and the lower series represent nonpromoter sequences. Values of the score statistics are based on the context tree model. It is evident that the two populations can be well distinguished by using the context tree. A straightforward application is thus a promoter identifier along the genome that is principally similar to SPC schemes. (Further details on this promising direction of research are beyond the scope of this article and can be found in Ben-Gal, Arviv, Shmilovici, and Grosse 2002.)

Production Monitoring via Buffers. A common practice in the analysis of production systems is to use queueing networks and Markov chains to model production lines, where the machines' processing times follow certain probability distributions. Extensive literature exists on the applicability of these models to the design and the analysis of production systems, whose states are defined by the level of the buffers in the line (see, e.g., Buzacott and Yao 1986a, 1986b; Bitran and Dasu 1992; Gershwin 1994). Nonetheless, a common practice in productivity-related SPC, is to monitor the machine processing times rather than the buffer levels themselves. One reason

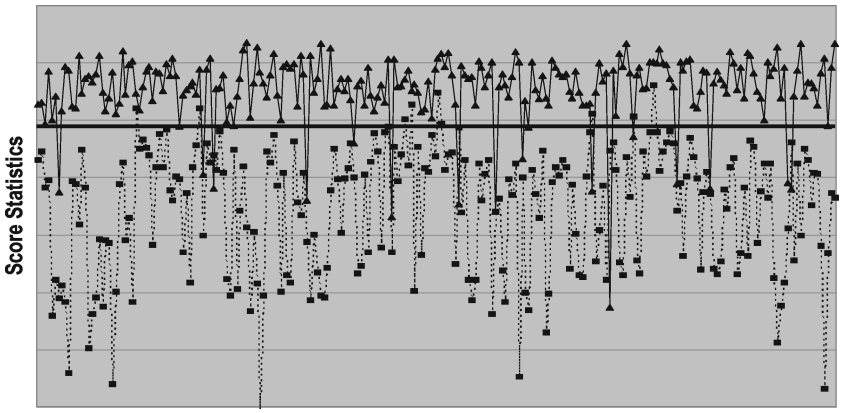


Figure 2. Score Statistics of Two E. coli Sequences of DNA-Spaced Reading Frames. The upper series represent promoter sequences (— ▲ —), and the lower series represent nonpromoter sequences (- - - ■ - - -). Values of the score statistics are computed by the context tree model.

to do this is that the statistical behavior of buffer levels is complex and highly nonlinear, and often cannot be described by a known stochastic process, and thus is inadequate for traditional SPC methods, as discussed in Section 4. On the other hand, there are several reasons to monitor the buffer levels instead of monitoring the machine processing-times. First, the buffer levels are direct measures for the productivity, as opposed to the processing times that are closely related, yet indirect measures of productivity. Second, because defective parts are screened out and do not enter the buffers, the buffer levels reflect not only the machine processing times, but also some quality features of produced parts, as well as the interactions among machines. These interactions are of particular importance in assembly lines. Finally, the buffer levels are affected by a number of machines that are located upstream and downstream of that buffer: a low productivity of upstream machines will cause the buffers to empty, whereas a low productivity of downstream machine will cause the buffers to fill. Thus, instead of monitoring every machine in the line, often it is sufficient to monitor only a few buffers.

In Section 4 we apply the CSPC procedure to buffer monitoring of a production line. We show that the CSPC succeeds in indicating inconsistencies of the production system, whereas traditional SPC methods fail to do so.

The rest of the article is organized as follows. Section 2 introduces the theoretical background for the context tree model and the principles of its construction (with a detailed construction algorithm and a walk-through example deferred to App. B). Section 3 develops the control limits for the CSPC methodology based on the KL statistic and presents the CSPC methodology. Section 4 illustrates the CSPC through a detailed numerical example and compares it with conventional SPC methods, and Section 5 concludes with some discussion.

2. MODELING PROCESS DEPENDENCE WITH CONTEXT TREES

In this section we introduce the context tree model for state-dependent data and the concepts of its construction algorithm following the definitions and notations of Rissanen (1983) and Weinberger et al. (1995). A detailed walk-through example presenting the context tree construction is given in Appendix B. (App. A includes a glossary of terms used in this article.)

Consider a sequence (string) of observations $x^N = x_1, \ldots, x_N$, with elements x_t, $t = 1, \ldots, N$, defined over a finite symbol set, X, of size d. In practice, this string can represent a realization sequence of a discrete variable drawn from a finite set. Particularly, the discrete variable can be a queue length in a queuing system, such as the number of parts in a buffer in a production line. For a finite buffer capacity c, the "finite symbol set" (of possible buffer levels) is $X = \{0, 1, 2, \ldots, c\}$ and d, the symbol-set size, is thus equal to $d = c + 1$. For instance, the string $x^6 = 1, 0, 1, 2, 3, 3$ repesents a sequence of six consecutive observations of the buffer level (number of parts) in a production line with buffer capacity of $c = 3$.

A family of probability measures $P_N(x^N)$, $N = 0, 1, \ldots,$ is defined over the set $\{X^N\}$ of all stationary sequences of length N, such that the marginality condition

$$\sum_{x \in X} P_{N+1}(x^N x) = P_N(x^N) \tag{1}$$

holds for all N, $x^N x = x_1, \ldots, x_N, x$, and $P_0(x^0) = 1$, where x^0 is the empty string. To simplify notation, the subindex is omitted, so that $P_N(x^N) \equiv P(x^N)$.

One could opt to find a model that assigns the probability measure (1). A possible finite-memory source model of the sequences just defined is the finite-state machine (FSM), which assigns a probability to an observation in the string based on a finite set of states. Hence the FSM is characterized by the transition function, which defines the state for the next symbol,

$$s(x^{N+1}) = f\big(s(x^N), x_{N+1}\big), \tag{2}$$

where $s(x^N) \in \Gamma$ are the states with a finite state space $|\Gamma| = S$, $s(x^0) = s_0$ is the initial state, and $f : \Gamma \times X \to \Gamma$ is the state transition map of the machine. The FSM is then defined by $S \cdot (d - 1)$ conditional probabilities, the initial state s_0, and the transition function. The set of states of an FSM should satisfy the requirement that the conditional probability to obtain a symbol given the whole sequence be equal to the conditional probability to obtain the symbol given the past state, implying that

$$P(x|x^N) = P\big(x|s(x^N)\big). \tag{3}$$

A special case of FSM is the Markov process, which satisfies (2) and is distinguished by the property that for a kth-order Markov process, $s(x^N) = x_N, \ldots, x_{N-k+1}$. Thus, reversed strings of a

fixed length k act as source states. This means that the conditional probabilities of a symbol given all past observations (3) depend only on a fixed number of observations k, which defines the order of the process. However, even when k is small, the requirement for a fixed order can result in an inefficient estimation of the probability parameters, because some of the states often depend on shorter strings than the process order. On the other hand, increasing the Markov order to find a best fit results in an exponential (noncontinuous) growth of the number of parameters, $S = (d-1)d^k$, and, consequently, of the number of conditional probabilities to be estimated.

An alternative model to the Markovian is the context tree model suggested by Rissanen (1983) for data-compression purposes and modified later by Weinberger et al. (1995). The tree presentation of a finite-memory source is advantageous, because states are defined as contexts—graphically represented by branches in the context tree with *variable length*—hence provide more flexability in defining the number of parameters and requires less estimation effort than those required for a Markov chain presentation. The context tree is an *irreducible* set of conditional probabilities of output symbols given their contexts. The tree is conveniently estimated by the *context* algorithm. The algorithm generates an asymptotically minimal tree fitting the data (Weinberger et al. 1995). The attributes of the context tree along with the ease of its estimation make it suitable for model-generic SPC applications, as discussed later.

A context, $s(x^t)$, in which the "next" symbol in the string x_{t+1} occurs is defined as the *reversed string* (we use the same notation for contexts as for the FSM states, because here they follow similar properties),

$$s(x^t) = x_t, \ldots, x_{\max\{0,t-k+1\}}, \tag{4}$$

for some $k \geq 0$, which is itself a function of the context, and not necessarily identical for all strings (the case $k = 0$ is interpreted as the empty string s_0). The string is truncated because the symbols observed before x_{t-k+1} do not affect the occurrence probability of x_{t+1}. For the set of *optimal contexts*, $\Gamma = \{s : \text{shortest contexts satisfying (3)}\}$, k is selected to attain the shortest contexts for which the conditional probability of a symbol given the context is practically equal to the conditional probability of that symbol given the whole data, that is, nearly satisfying (3). Thus an optimal context, $s \in \Gamma$, acts as a state of the context tree, and is similar to a state in a regular Markov model of order k. However, unlike the Markov model, the lengths of various contexts do not have to be equal, and k does not need to be fixed such that it accounts for the maximum context length. The variable context lengths in the context tree model provide more flexibility and result in fewer parameters that have to be estimated and, consequently, require less data to identify the source.

Using the foregoing definitions, a description of the context tree follows. A context tree is an irreducible set of probabilities that fits the symbol sequence x^N generated by a finite-memory source. The tree assigns a distinguished optimal context for each element in the string, and defines the probability of the element, x_t, given its optimal context. These probabilities are used later for SPC—comparing sequences of observations and identifying whether they are generated from the same source.

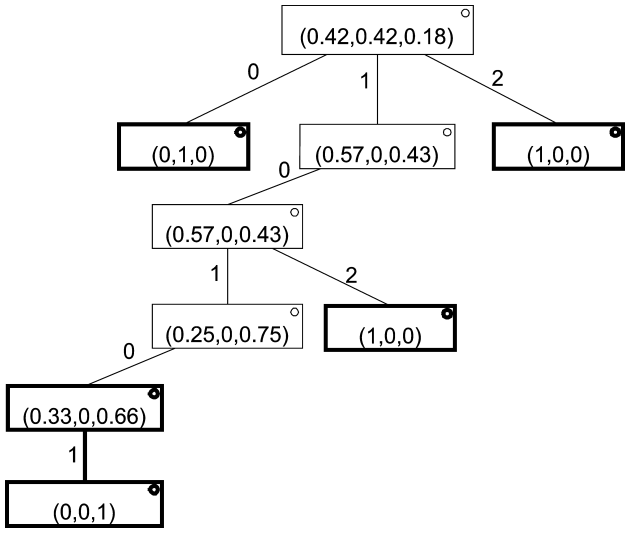


Figure 3. Illustration of a Context Tree With $S = 5$ Optimal Contexts (bolded frame).

Graphically, the context tree is a d-ary tree that is not necessarily complete and balanced. Its branches (arcs) are labeled by the different symbol types. Each node contains a vector of d conditional probabilities of all symbols, $x \in X$, given the respective context (not necessarily optimal), which is represented by the path from the root to that specific node. An optimal context, $s \in \Gamma$, of an observation x_t is represented by the path starting at the root, with branch x_t followed by branch x_{t-1}, and so on, until it reaches a leaf or a partial leaf (see Appendix B for a proper definition of a partial leaf).

Figure 3 exemplifies a context tree constructed from a sequence of observed buffer levels in a production line. Because in this case the buffer has a finite capacity of $c = 2$, there are $d = 3$ symbol types, where observation $x_t \in \{0, 1, 2\}$ refers to the number of parts in the buffer at time t. Following the context algorithm (which is detailed in App. B), $S = 5$ optimal contexts are found (marked by a bold frame); thus the set of optimal contexts is a collection of reversed strings, $\Gamma = \{0, 2, 102, 1010, 10101\}$ (read from left to right). The context 1010 is a partial leaf.

Consider the string $x^6 = 1, 2, 0, 1, 0, 1$, which is generated from the tree source in Figure 3. Using the foregoing definitions, the optimal context of the next element, $x_7 = 0$, is $s(x^6) = 1, 0, 1, 0$, that is, following the reverse string from the root until reaching an optimal context. Accordingly, the probability of x_7 given the context is $P(x_7 = 0|s(x^6)) = .33$. For a detailed example, see Appendix B.

Note that had we used a Markov chain model with maximal dependency order, which is $k = 5$ (the longest branch in the tree), then we would need to estimate the parameters of $3^5 = 243$ states (instead of the five optimal contexts in the context tree of Fig. 3), although most of them are redundant.

In practical SPC applications, one usually does not have an a priori knowledge of the dependencies that need to be estimated. The conditional probabilities of symbols given the optimal contexts, $P(x|s)$, $x \in X$, $s \in \Gamma$, and the marginal probabilities of optimal contexts, $P(s)$, $s \in \Gamma$, are estimated by the context algorithm. The joint probabilities of symbols and optimal contexts, $P(x, s)$, $x \in X$, $s \in \Gamma$, are used to derive the

CSPC control bounds and represent the context tree model. This model might be only an approximated description of the real generating source, but it is often appropriate for practical purposes.

2.1 The Context Algorithm

In this section we briefly discuss the construction algorithm of the context tree. The algorithm constructs a context tree from a string of N symbols and estimates the marginal probabilities of contexts and the conditional probabilities of symbols given contexts. It contains four stages: (1) tree growing and counter updating, (2) tree pruning, (3) optimal contexts selection, and (4) estimation of the context tree probability parameters.

In the first stage, a *counter context tree*, $\mathbf{T}_t$, $0 \le t \le N$, is grown up to a maximum depth m. (We distinguish between the counter context tree, which results from the first two stages in the algorithm, and the context tree, which contains the final set of optimal contexts.) Each node in $\mathbf{T}_t$ contains d counters—one per each symbol type. The counters, $n(x|s)$, denote the conditional frequencies of the symbols $x \in X$ in the string x^t given the context s. Along with the tree growth, the counter values $n(x|s)$ are updated according to symbol occurrences. In the second stage, the counter tree is pruned to acquire the shortest reversed strings to satisfy (3). In the third stage, the set of optimal contexts Γ is obtained, based on the pruned counter tree. In the last stage, the estimated conditional probabilities of symbols given optimal contexts, $\widehat{P}(x|s)$, $x \in X$, $s \in \Gamma$, and the estimated marginal probabilities of optimal contexts, $\widehat{P}(s)$, $s \in \Gamma$, are derived. As noted in Appendix B, both $\widehat{P}(x|s)$ and $\widehat{P}(s)$ are asymptotically multinomial distributed and used to obtain the CSPC control limits. The estimated joint probabilities of symbols and optimal contexts, $\widehat{P}(x, s) = \widehat{P}(x|s) \cdot \widehat{P}(s)$, $x \in X$, $s \in \Gamma$, are then derived and represent the context tree in its final form.

A convergence theorem for the context algorithm (Weinberger et al. 1995) guarantees a rapid convergence (of order $\log t/t$) of the estimated context tree to the "true" data-generating tree source. The complexity of the context algorithm is $O(N \log N)$ for an input string of length N (Rissanen 1999). An extended version of the algorithm and a running example for the context tree construction are presented in Appendix B.

3. CONTEXT-BASED STATISTICAL PROCESS CONTROL

3.1 The Kullback–Leibler "Distance" Measure Between Context Trees

Kullback (1959) proposed a measure for the relative "distance" or the discrimination between two probability mass functions $Q(x)$ and $Q_0(x)$,

$$K\big(Q(x), Q_0(x)\big) = \sum_{x \in X} Q(x) \log \frac{Q(x)}{Q_0(x)} \ge 0. \tag{5}$$

This measure, which became known as the KL measure, is positive for all nonidentical pairs of distributions and equals 0 iff $Q(x) = Q_0(x)$ for every x. The KL measure is a convex function in the pair $(Q(x), Q_0(x))$, and invariant under all one-to-one transformations of the data. Although it is used as a distance measure, it is not symmetric and does not satisfy the triangular inequality. Kullback (1959) has shown that the KL distance (multiplied by a constant) between a d-category multinomial distribution $Q(x)$ and its estimated distribution $\widehat{Q}(x)$ is asymptotically chi-squared distributed with $d - 1$ degrees of freedom (df):

$$2N \cdot K\big(\widehat{Q}(x), Q(x)\big) \to \sum_{x \in X} \frac{(n(x) - NQ(x))^2}{NQ(x)} \sim \chi^2_{d-1}, \tag{6}$$

where N is the size of a sample taken from the population specified by $Q(x)$, $n(x)$ is the frequency of category (symbol type) x in the sample, $\sum_{x \in X} n(x) = N$, and $\widehat{Q}(x) = n(x)/N$ is the estimated probability of category x.

The KL measure for the relative "distance" between two joint probability mass functions, $Q(x, y)$ and $Q_0(x, y)$, can be partitioned into two terms, one term representing the distance between the conditioning random variable and the other representing the distance between the conditioned random variable (Kullback 1959),

$$K\big(Q(x, y), Q_0(x, y)\big) = \sum_{y \in S} Q(y) \log \frac{Q(y)}{Q_0(y)} + \sum_{y \in S} Q(y) \sum_{x \in X} Q(x|y) \log \frac{Q(x|y)}{Q_0(x|y)}. \tag{7}$$

In this article we implement the KL measure to detect the relative distance between two context trees. Another distance measure could be used, such as Jensen–Shannon (e.g., Ben-Gal et al. 2002). The first tree, denoted by $\widehat{P}_i(x, s)$, represents the monitored distribution of symbols and contexts, as estimated from a string of length N at the monitoring time $i = 1, 2, \ldots$. The second tree, denoted by $P_0(x, s)$, represents the "in control" reference distribution of symbols and contexts. The reference distribution is either known a priori or can be well estimated by the context algorithm from a long string of observed symbols.

Following the minimum discrimination information (MDI) principle (Alwan et al. 1998), the context algorithm generates a monitored tree with a structure similar to that of the reference tree. Maintaining the same structure for the monitored tree and the reference tree enables direct use of the KL measure. New observations are being collected and used for updating the monitored tree counters and its statistics [(B.3) and (B.4) in App. B]. A significant change in the monitored process is manifested in the tree counters and its resulting probabilities.

Using (7), the KL measure is decomposed for the monitored context tree and the reference context tree (both represented by the joint distributions of symbols and contexts) into two terms,

$$K\big(\widehat{P}_i(x, s), P_0(x, s)\big) = \sum_{s \in \Gamma} \widehat{P}_i(s) \log \frac{\widehat{P}_i(s)}{P_0(s)} + \sum_{s \in \Gamma} \widehat{P}_i(s) \sum_{x \in X} \widehat{P}_i(x|s) \log \frac{\widehat{P}_i(x|s)}{P_0(x|s)}, \tag{8}$$

one term measuring the KL distance between the trees' context probabilities and the other measuring the KL distance between the trees' conditional probabilities of symbols given contexts.

Under the null hypothesis that the monitored tree $\widehat{P}_i(x, s)$ is generated from the same source that generated $P_0(x, s)$, and using the multinomial approximation (6) together with (8), we

derive the asymptotic probability density function of the KL measure between $\widehat{P}_i(x, s)$ and $P_0(x, s)$; that is, for a long string,

$$\begin{aligned} K\big(\widehat{P}_i(x,s), P_0(x,s)\big) \to & \\ \frac{1}{2N}\chi^2_{S-1} + \sum_{s\in\Gamma} \widehat{P}_i(s) \cdot \frac{1}{2n(s)}\chi^2_{d-1} & \\ = \frac{1}{2N}\chi^2_{S-1} + \sum_{s\in\Gamma} \frac{n(s)}{N} \cdot \frac{1}{2n(s)}\chi^2_{d-1} & \\ = \frac{1}{2N}\chi^2_{S-1} + \frac{1}{2N}\sum_{s\in\Gamma} \chi^2_{d-1} & \\ = \frac{1}{2N}(\chi^2_{S-1} + \chi^2_{S(d-1)}) = \frac{1}{2N}\chi^2_{Sd-1}, & \qquad (9) \end{aligned}$$

where $n(s)$ is the frequency of an optimal context $s \in \Gamma$ in the string, S is the number of optimal contexts, d is the size of the symbol set, and N is the size of the monitored string, which can be determined either numerically or iteratively as exemplified in Section 4.1. Thus the KL statistic for the joint distribution of symbols and optimal contexts is asymptotically chi-squared distributed with degrees of freedom depending on the number of symbol types and the number of optimal contexts. (The degrees of freedom are doubled when using an estimated reference distribution.) This result is of utmost significance for the development of control charts for state-dependent discrete data streams based on the context tree model; Given a type I error probability α, the control region for the KL statistic is given by

$$0 \le 2N \cdot K\big(\widehat{P}_i(x,s), P_0(x,s)\big) \le \chi^2_{Sd-1,1-\alpha}. \qquad (10)$$

Thus the upper control limit (UCL) is the $100(1-\alpha)$ percentile of the chi-squared distribution with $(Sd-1)$ degrees of freedom.

The control limit (10) has some appealing characteristics, as follows:

1. It is a one-sided bound; if the KL value is larger than the UCL, then the process is assumed to be "out of control" for a given level of significance.
2. It lumps together all the parameters of the context tree, in contrast with traditional SPC, in which each process parameter is controlled separately. Nevertheless, the KL statistic of the tree can be easily decomposed to monitor each node in the context tree separately. This can be beneficial when seeking the cause of an "out of control" signal.
3. If Sd is large enough, then the KL statistic is approximately normally distributed. Hence conventional SPC charts can be directly applied to monitor the proposed statistic.

A basic condition for applying the KL statistic to sample data requires that $P_0(x|s) > 0, \forall x \in X, \forall s \in \Gamma$. This constraint is satisfied with the predictive approach [see (B.4) in App. B], where all probability values assigned to any of the symbol types are strictly positive, or with the nonpredictive approach [see (B.4)] by defining $0/0 \equiv 0$.

3.2 The Context-Based Statistical Process Control Monitoring Procedure

The following steps briefly outline the CSPC monitoring procedure:

1. Obtain the reference context tree $P_0(x, s)$, either analytically or by using the context algorithm to a long string of data.
2. For any monitoring time point, take a sample of sequenced observations of size N and generate the monitored tree, $\widehat{P}_i(x, s)$. Each sequence is called a "run" and contributes a monitoring point in the CSPC chart. Adjust the structure of the monitored tree such that it fits the structure of the reference context tree. Update the counters of the monitored context tree by the values of the string. Estimate the probability measures of the monitored context tree using (B.3) and (B.4).
3. Compute the KL estimates, measuring the relative "distance" between the estimated monitored distributions $\widehat{P}_i(x, s)$ and the reference distributions $P_0(x, s)$.
4. Plot the KL statistic value in the control chart against the UCL found using (10). If the KL value is larger than the UCL, this indicates that a significant change has likely occurred in the process. Search for special-cause variability and eliminate it.
5. For the next monitoring point, collect a new string of data, increment $i = i + 1$, and go to step 2. If no data are available, then stop the monitoring stage.

4. NUMERIC EXAMPLE: BUFFER MONITORING IN A PRODUCTION LINE

Consider a production line with K machines modeled as a network of reliable service stations $(M_1, M_2, \ldots, M_K)$ and separated by buffer storages $(B_1, B_2, \ldots, B_K)$. Buffers carry parts between two consecutive operations and have finite capacities. Figure 4 presents a two-machine subsystem of a larger production line, which can be decomposed through methods shown by Gershwin (1994). The monitored subsystem consists of two machines, M_k and M_{k+1}, and a buffer, B_k, with a finite capacity, c.

We denote the probability that machine M_k has finished processing a part during the inspection time interval by p_k and call it the *production probability*. Accordingly, $q_k = 1 - p_k$ is the probability that the machine has not finished its process during the inspection time interval. We denote the buffer levels by b, $b = 0, \ldots, c$, and define them as the system states (a conventional approach in production system analysis). Such a definition of states is beneficial for several reasons: (1) The state space is finite and well defined; (2) as seen later, a rigorous monitoring of buffer levels can indicate whether there has been

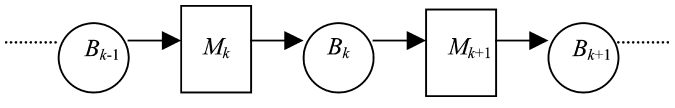


Figure 4. A Subsystem of a Production Line of K Machines.

a productivity change in the system—including changes in machines and buffers that are not part of the considered subsystem; and (3) the transition probabilities between states can be computed using known models, such as Markov chains. For example, the first-order Markov model assumes that transition probabilities depend only on the current buffer levels and is given by the following equations (assuming that an empty/full buffer will trigger an automatic filling/emptying process in the next time interval):

$$\begin{aligned} &P(x_{t+1}=b|x_t=b-1)=P(0|c)=p_k q_{k+1}, && b=1,\ldots,c;\\ &P(x_{t+1}=b|x_t=b+1)=P(c|0)=p_{k+1}q_k, && b=0,\ldots,c-1;\\ &P(x_{t+1}=b|x_t=b)=1-p_k q_{k+1}-p_{k+1}q_k, && b=0,\ldots,c, \end{aligned} \tag{11}$$

where x_t is the observed buffer level at time t defining the system state at time t.

In the remainder of the section, we use a Markov model of the monitored subsystem. This example is chosen because it allows a straightforward comparison between the known Markov model and its equivalent context tree. Note that the context tree in general is more appropriate than the Markovian to model the considered production process, because various states (i.e., various buffer levels) might have a different dependency order. For example, even when the production process is "in control," the middle (centered) buffer levels might follow a first-order Markov chain, whereas higher and lower buffer levels might follow high-order dependencies, which result from trends in machine productivity.

The example contains three parts: (1) derivation of the "in control" model for the production process by using the context algorithm, (2) application of CSPC to the monitored variables during "in control" and "out of control" phases of the production system, and (3) comparison of the CSPC to Shewhart and time series–based SPC methods.

4.1 Process Description and Derivation of the "In Control" Model

Figure 5 presents the first-order Markov diagram of the buffer levels obtained by specifying a buffer capacity, $c = 4$, and production probabilities, $p_k = p_{k+1} = .8$. This production probability value (which can be estimated in practice by the machine production rate) has been selected for two reasons. First, it represents a relatively high production rate, which is not too high for monitoring purposes, because anomalies at higher production rates are easier to detect. Second, it guarantees a steady-state buffer level equal to 2 (the steady-state probabilities are all equal to .2, as shown later in Fig. 6)—exactly half of the buffer capacity. Substituting these production probabilities in (11) yields a state transition probability where $P(x_{t+1}=b|x_t=b-1)=P(x_{t+1}=b|x_t=b+1)=.16$; $P(x_{t+1}=b|x_t=b)=.68$, $b=0,\ldots,4$.

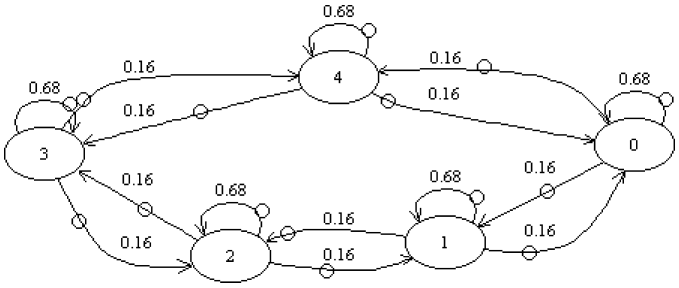


Figure 5. State Transition Diagram for the Process.

To obtain a direct comparison with conventional SPC methods based on the normal distribution, the Markov model is also defined as a restricted random walk, which is generated as follows:

1. A sequence of N values is generated from an iid normal process, with mean $\mu_0 = 0$ and standard deviation $\sigma_0 = 1$.
2. The string values are quantized by selecting two thresholds (approximately -1 and 1) to obtain a sequence of discrete random steps, $z_i \in \{-1, 0, 1\}$, $i = 1, 2, \ldots, N$, which represent the change in the buffer level, where $P(z_i = -1) = P(z_i = 1) = .16$ and $P(z_i = 0) = .68$.
3. The cumulated sum of the independent random steps defines an unconstrained random-walk process, which is equivalent to a buffer level with infinite capacity.
4. Because the buffer capacity is finite, the absolute values of the unconstrained random walk are restricted to a finite integer range: the modulo(5) function is applied to the data to obtain a symbol set of constant size $d = 5$ of the symbol set $X = \{0, 1, 2, 3, 4\}$.

The underlying normal distribution permit a straightforward comparison to conventional SPC methods in later steps of the example. Table 1 exemplifies a short realization of the generating process for $N = 5$.

An analytical context tree (Fig. 6) can be directly derived from the Markov diagram in Figure 5. It is a single-level tree with $S = 5$ contexts and a symbol set of $d = 5$. The root node displays the steady-state probabilities of the Markov process, and the contexts (the leafs) display the transition probabilities given the context. This context tree represents the "in control" reference distribution $P_0(x, s)$ of the process.

Because the analytical model is often unknown in practice, let us illustrate the convergence of the estimated context tree, $\widehat{P}_0(x, s)$, to the analytical context tree, $P_0(x, s)$ of Figure 6. The context algorithm is applied to an increasing-length data string, which is generated from the restricted random walk. As the string grows, the constructed tree converges to the analytical model and the KL distance measure between the trees approaches 0. Figure 7 presents the asymptotic convergence of

Table 1. Feedback-Controlled Process Generation Example

	x_1	x_2	x_3	x_4	x_5
Step 1: iid normal values string	−.4326	−1.6656	.1253	.2877	−1.1465
Step 2: Quantized string	0	−1	0	0	−1
Step 3: Cumulated sum string	0	−1	−1	−1	−2
Step 4: Restricted absolute string [modulo(5)]	0	4	4	4	3

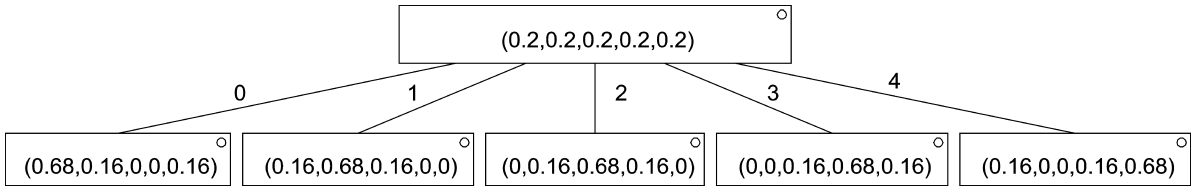


Figure 6. The Analytically Derived Single-Level Context Tree.

the KL "distance" between $\widehat{P}_0(x, s)$ and $P_0(x, s)$ as a function of N, the string length. The bold line in Figure 7 indicates that a longer input string results in an improved estimation of the analytical distributions $P_0(x, s)$. It also shows the rapid convergence of context algorithm to the "true" model. The dotted line indicates the weighted UCL, $\chi^2_{(24,.9975)}/(2 \cdot N)$, as derived from (10). Notice that for $N > 325$, the KL value is constantly below the weighted UCL. Figure 7 can help an experimenter determine the string length, N, required for a satisfactory estimation of the reference "in control" context tree. In particular, note that for approximately $N < 300$, the KL measure is in its transient mode, whereas for $300 < N < 700$, the KL measure stabilizes and then attends a steady-state mode.

When the "in control" model is unknown, we suggest two alternative procedures for determining the initial string length required to estimate it. The first of these is an empirical approach: computing the convergence rate of estimated context trees, which are constructed from an ever-growing input string, until the convergence rate reaches its steady state-value. Recall that context algorithm converges to the true tree model in a rate of $\log N/N$. The second approach is based on the multinomial distribution characteristics. Bromaghin (1993) and May and Johnson (1997) summarized several techniques to determine the sample size needed for estimation of the multinomial parameters given a significance level or a specified interval width. Their results might be used to determine analytically the initial string length. As an example, Bromaghin (1993) suggested the following upper bound for the sample size (based on a worst-case scenario when the probability of a category equals .5):

$$N = 1 + \text{int}\left(\max_{x \in X}\left[\frac{.25 z^2_{(1-\alpha_x/2)}}{\Delta_x^2} - z^2_{(1-\alpha_x/2)}\right]\right), \qquad (12)$$

where N is the sample size, $z^2_{(1-\alpha)}$ is the normal distribution percentile, α_x is the significance level required for $x \in X$, Δ_x is the interval width for $x \in X$, $\sum_{x \in X} \alpha_x \leq \alpha$, and int$(\cdot)$ represents the integer function. Bromaghin provided a table of recommended sample sizes based on (12) for the case of equal interval width and equal significance levels, that is, $\Delta_x = \Delta$ and $\alpha_x = \alpha/d$, $\forall x \in X$. For example, for $\alpha = .05$, $\Delta = .05$ and $d = 15$ (the number of categories), one obtains $N_0 = 853$. This worst-case approach may provide a simple rule of thumb for sample size determination.

The estimated context tree model in Figure 8 was derived by applying the context algorithm to a string of $N = 1{,}000$ observations determined by the results in Figure 7. The predictive approach [see (B.4) in App. B] was used to compute the estimated conditional probabilities $\widehat{P}_0(x|s)$, $\forall x \in X$, $s \in T_N$.

To conclude, the context algorithm performed well with respect to both the monitored tree structure and its estimated probability measures. The algorithm accurately identified $\widehat{P}_0(x, s)$ as a single-level tree with $S = 5$ optimal contexts (corresponding to the equivalent Markov states in Fig. 5). The estimated conditional probabilities of symbols given contexts were "close" to the analytical probabilities in terms of the KL measure.

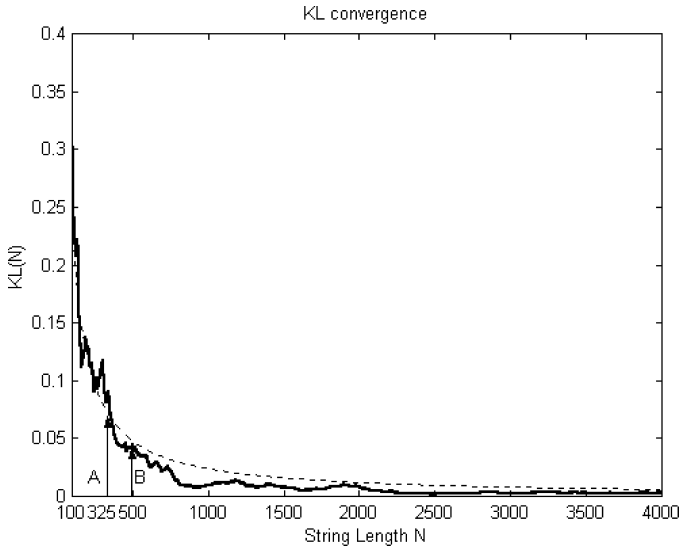


*Figure 7. The KL Value (——) Between the Analytical Tree and the Estimated Tree as a Function of the Input String Length N (- - - UCL/(2*N) limit value).*

4.2 The Monitoring Stage

Based on the structure of the reference context tree, the UCL was calibrated to obtain a type I error probability of $\alpha = .0025$, which corresponds to the typical choice of "3 sigma" in traditional SPC. Following (10), the UCL for the KL statistic was determined to $\chi^2_{(S \cdot d-1,1-\alpha)} = \chi^2_{(24,.9975)} = 48$.

Two shift scenarios of the underlying normal distribution were selected to illustrate the performance of CSPC monitoring procedure:

1. Shifts in the standard deviation of the underlying normal process, denoted by $\sigma' = \lambda \cdot \sigma_0$, where $\sigma_0 = 1$, and λ taking the values of 1 ("in control"), .5, 1.5, and 2.
2. Shifts in the mean of the underlying normal process, denoted by $\mu' = \mu_0 + \delta \cdot \sigma_0$, where $\mu_0 = 0$, $\sigma_0 = 1$, and δ varyies between 0 and 3.

During the monitoring stage of both scenarios, consecutive strings of length of $N = 125$ data points were used to generate 50 monitored context trees, $\widehat{P}_i(x, s)$, $i = 1, 2, \ldots, 50$, for

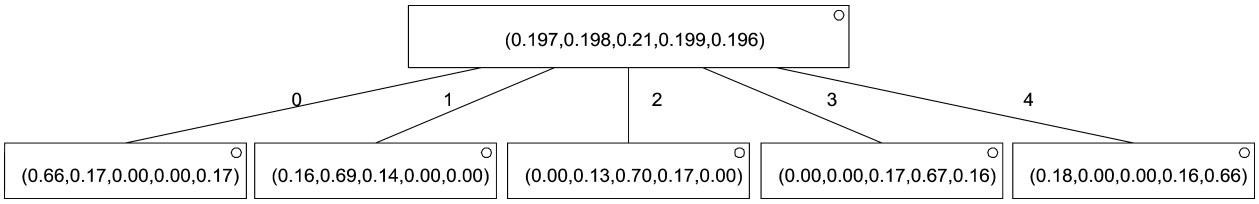


Figure 8. Estimated Reference Context Tree as a Result of the Implementation of Context Algorithm to N = 1,000 Observations.

each shifted process. A segment of 50 runs was used for a clear graphical presentation, allowing us to sketch both the "in control" and "out of control" charts on the same graph. String lengths of $N = 125$ adhere to the chi-squared sampling principle suggested by Cochran (1952), requiring that at least 80% of the sampling bins, in this case corresponding to the nonzero conditional probabilities of symbols given optimal contexts, $P_i(x|s)$, contain at least four data points. Note that this string length is much larger than sample sizes used in conventional SPC methods (e.g., $N = 5$ in Shewhart charts). The estimation of parameters of a model-generic method requires a larger sample size as the order of dependency increases. Larger sample sizes can be excused in processes, such as the buffer-level monitoring example considered here, where the sampling frequency is high and relatively cheap. The proposed CSPC procedure should be implemented primarily on such environments. New statistics other than the KL might decrease the required string length and will be investigated in future research.

Scenario 1: Shifts in the Standard Deviation of the Underlying Normal Distribution. The CSPC monitoring procedure (outlined in Sec. 3.2) was applied to shifts in the standard deviation of the underlying normal distribution. A total of 50 runs with respective λ values of 1, 1.5, 2, and .5 were generated from each shifted distribution. Monitored trees were constructed for each run, and the KL estimates between each monitored tree and the reference tree were computed and plotted on the control chart. Figures 9 and 10 present the control charts for all the shifted processes. Based on the simulated results in these figures, Table 2 presents the probabilities of the random-walk steps due to the shift in the underlying standard deviation, the corresponding type II errors, and the estimated ARL.

As can be seen, for the "in control" process ($\lambda = 1$), 100% of the runs are below the UCL, which implies that the type I error probability of the CSPC procedure in this example equals 0. For shifted processes, the rate of successful shift detection by the CSPC is relative to the transition probability change. A standard deviation shift of $\lambda = .5$ decreased the transition probability significantly, from .16 to .02, and resulted in 100% of the runs being out of the control limits (Fig. 10). However, a standard deviation shift of $\lambda = 1.5$ increased the transition probability from .16 to .25 and resulted in 20% of the runs being above the UCL (Fig. 9, dotted line). The corresponding ARL = 5 is based on approximately 625 observations and emphasizes the large sample size that might be required by the CSPC in some cases. Some ideas on how to shorten the CSPC's sample size are given in the conclusions section.

Time series–based SPC methods are designed to detect shifts in either process means or the process standard deviation. As

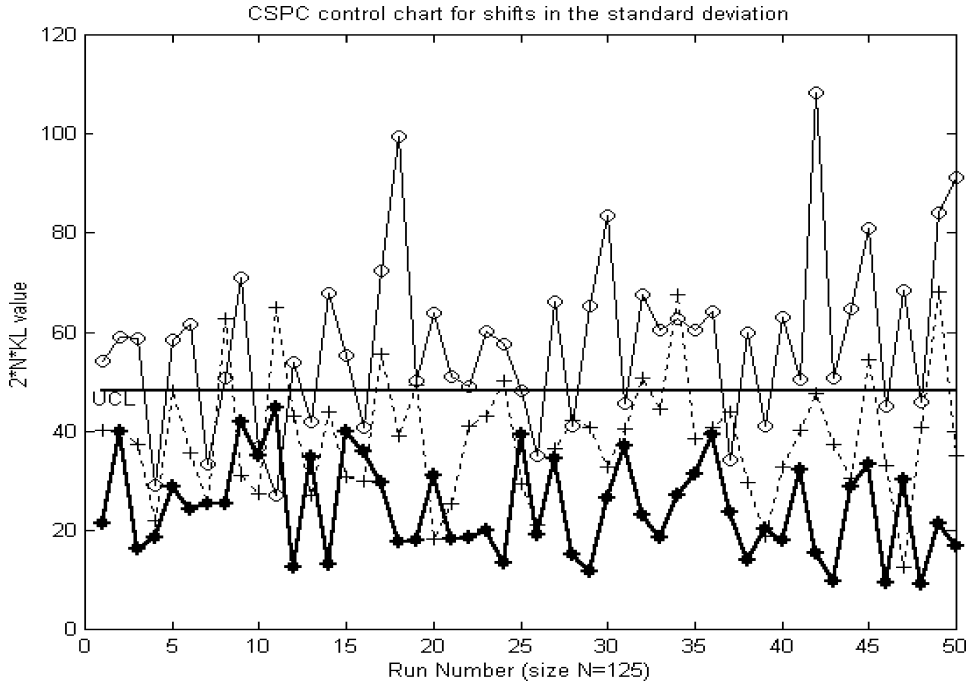


Figure 9. Shifts in the Process Underlying Normal Standard Deviation, $\lambda = 1, 1.5, 2$. A portion of 50 runs is presented. (—+—, Std = 1 (in control); -+-, Std = 1.5 (out-control); —○—, Std = 2 (out-control); ——, UCL.)

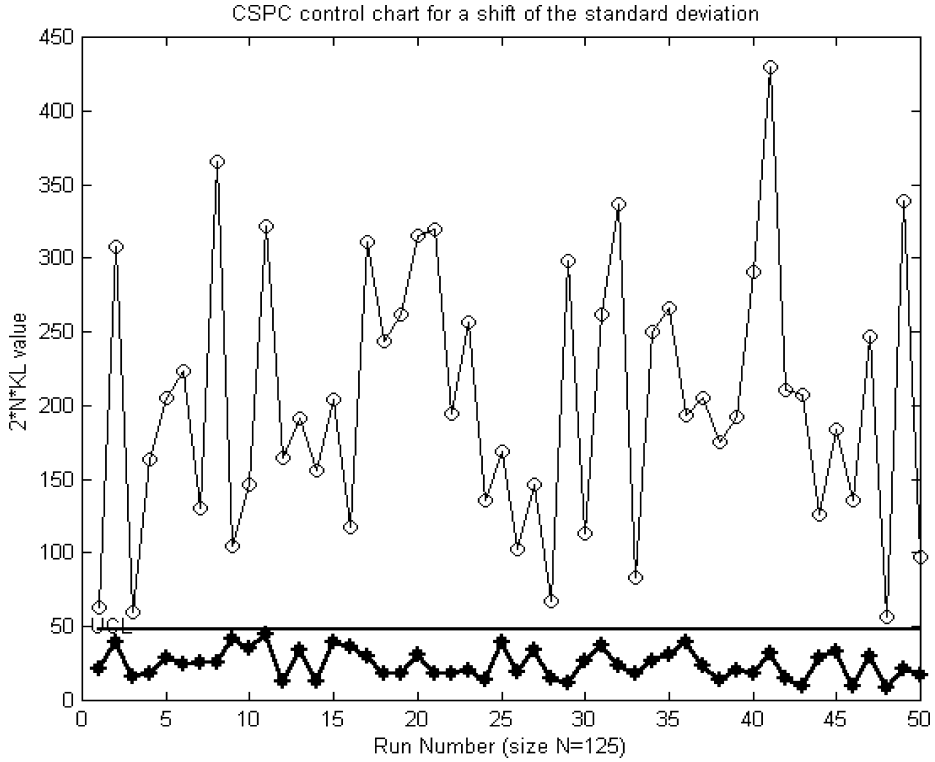


Figure 10. Shift in the Process Underlying Normal Standard Deviation, $\lambda = .5, 1$. A portion of 50 runs is presented. (—✚—, Std = 1 (in-control); —⊖—, Std = .5 (out-control); ——, UCL.)

exemplified in the next scenario, the CSPC is capable of identifying both types of shifts by using a single monitoring chart, because both modify the transition probabilities that affect the KL estimates.

Scenario 2: Shifts in the Mean of the Underlying Normal Distribution. The CSPC performance in detecting mean shifts of the underlying normal distribution, $\mu' = \mu_0 + \delta \cdot \sigma_0$, is presented by the operating characteristics (OC) curve. The OC curve plots the type II error ("in control" probability) as a function of the mean shift in standard deviations magnitude, denoted by δ.

Figure 11 presents the OC curve for the KL estimates. The runs were generated from the modified random-walk process, where the mean shift of the underlying normal distribution varied between 0 and 3 standard deviations ($\delta \in [0, 3]$). For comparison purposes, we also plotted an OC curve for a traditional $\overline{X}$ statistic of an iid Gaussian process with a sample size $N = 5$. Indeed, such a difference in the sample sizes, as well as the fact that each statistic is applied to a different process, makes the comparison problematic. However, as shown in the next section, when traditional SPC methods (Shewhart and time series based) were applied to the same random-walk process, they were incapable of monitoring this state-dependent process, regardless of the sample size.

4.3 Comparison Between Context-Based and Conventional Statistical Process Control Methods

In their seminal book, Box and Jenkins (1976) applied various time series models to discrete and nonlinear series, such as monthly stock closing prices, yearly sunspot numbers, and rounded-up yields of a chemical process. They showed that simple ARIMA models, such as AR(1) or IMA(1, 1), can effectively filter a wide range of autocorrelated processes, even if those do not fit the model exactly. Apley and Shi (1999) proposed applying simple ARIMA models to complex processes, because these models require less estimation effort and can successfully model a general autocorrelated process. Shore (2000) demonstrated the capability and robustness of the Shewhart method to monitoring processes that deviate from the normality assumptions. Nonetheless, as discussed in this section, all

Table 2. Performance of CSPC During Scenario with Respect to the ARL and Type I Error

Standart Shift	Probability of the random-walk steps			CSPC performance (ARL and type II error)		ARL of S-chart (benchmark)	
				Test power $1-\beta$		ARL	ARL
λ	+1	0	−1	(number of "out of control" runs)	Estimated ARL	$N = 5$	$N = 125$
1	.16	.68	.16	0% (0)	∞	257.13	369.08
1.5	.25	.5	.25	20% (10)	5	6.96	1
2	.31	.38	.31	74% (37)	1.35	2.35	1
.5	.02	.976	.02	100% (50)	1	∞	1

NOTE: The last two columns present the ARL of a traditional *S*-chart for sample sizes $N = 5$ and $N = 125$.

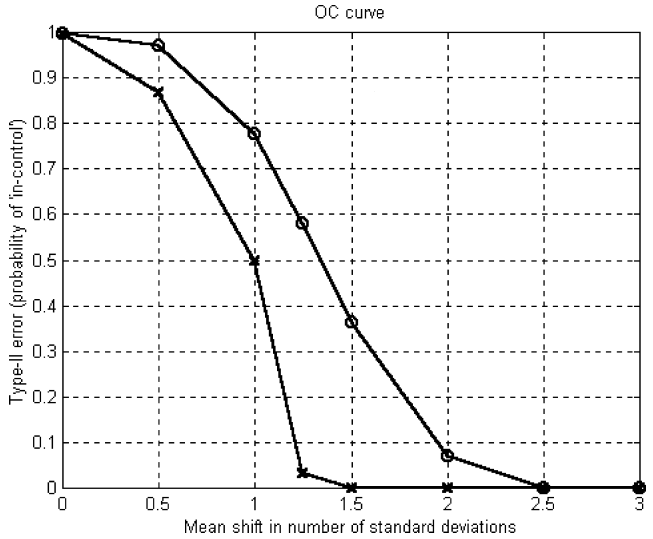


Figure 11. The Operating Characteristics Curve for Mean Shifts. —×— CSPC (Markov process N = 125); —◇— Shewhart (Gaussian i.i.d N = 5).

of the abovementioned assumptions do not hold if the system is highly nonlinear and the data depart significantly from the assumed underlying model.

In this section we apply known SPC methods for dependent data to the random-walk process of Section 4.1. We focus on the performance of Shewhart- and ARIMA-based approaches suggested for "general-purpose" monitoring. We show that all of the conventional SPC methods failed to monitor the considered process.

We started by implementing the SCC method, suggested by Alwan and Roberts (1988), to the "in control" data. The SCC monitors the residuals of an ARIMA(p, d, q) filtering. We used the *Statgraphics* software package to obtain the best-fitting ARIMA model (i.e., that with the lowest mean squared error) describing the random-walk process. It turned out to be the following AR(2) model: $\hat{x}_t = 1.948 + .406x_{t-1} - .0376x_{t-2}$.

The residuals of the AR(2) predictions, $\hat{\varepsilon}_t = x_t - \hat{x}_t$, were accumulated in subgroups of size $N = 5$ and charted by the *Statgraphics* software. In linear processes, the residuals of the best-fitting ARIMA filter should be approximately uncorrelated and normally distributed (see, e.g., Apley and Shi 1999). Figure 12, which presents the SCC of the "in control" data, indicates that

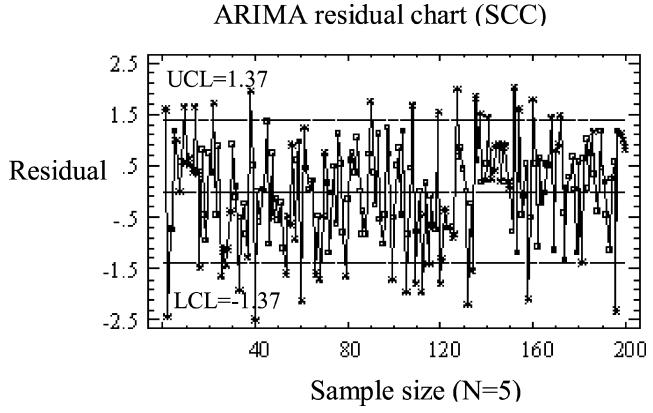


Figure 12. The SCC Control Chart for "In Control" Data.

these assumptions are violated here. More than 40% of the data points in Figure 12, denoted by the star signs, are marked as "out of control," although the random-walk process remained unchanged. This renders the SCC uninformative. The same scenario was tested against various time series models, including the AR(1), AR(2), and MA(1), which are often recommended for general autocorrelated processes. The tests were performed on both the residuals and the observations, using sample sizes of $N = 5$ and $N = 1$ (i.e., individuals). The results of all tests were unanimous: either a large number of points were marked as "out of control," although part of the process was "in control," or a large number of points were marked within the control limits although the process was "out of control." In both cases, it was impossible to distinguish between "in control" data and "out of control" data.

It should not be surprising that ARIMA charts are inadequate in this example. The linear ARIMA series cannot model the nonlinear state-dependent behavior of a Markov chain (even a simple first-order chain). Using these models resulted in violation of the independence and the normality assumptions of the residuals that are crucial for the success of the method.

Next, the Matlab software was used to obtain the $\overline{X}$ and the S Shewhart charts for further investigation. To evaluate the Shewhart performance, half of the runs were generated from the "in control" random walk, whereas the other runs were generated from an "out of control" random walk. The latter process was generated by shifting the mean of the underlying normal distribution by 1 standard deviation, that is, where $\mu' = \mu_0 + 1 \cdot \sigma_0$.

Figures 13 and 14 present the $\overline{X}$ and the S charts of both the "in control" data (solid line) and the "out of control" data (dashed line). The estimated parameters of the underlying distribution using a sample size $N = 5$ are $\hat{\mu} = \overline{\overline{X}} = 3.036$ and $\hat{\sigma} = \overline{S}/c_4 = .6148/.94 = .654$, where c_4 is the correction constant to obtain an unbiased estimator of σ. Notice the high rate of both types of statistical errors. The high type I error results because neighboring observations in the random walk tend to generate a small sample variance (high probability for a step size of 0), whereas the variance between samples in the "in control" string is large. The high type II error is due to the fact that $\overline{\overline{X}}$ remains approximately unchanged between the "in control" and "out of control" processes. Even though the process standard deviation is slightly larger in the "out of control" case (because the probability for a step size of $+1$ is greater), the variance of the sample averages is smaller than in the "in control" case. The same phenomena were identified for shifts of two standard deviations from the underlying process mean.

A reasonable assumption favoring the Shewhart method is that due to the central limit theorem, using a larger sample size would improve its performance. To check this assumption, the experiment was repeated with a larger sample size

Table 3. Change in the Process Transition Probabilities due to a Shift of $\delta = 1$

Mean shift	Transition probabilities of the random-walk steps		
δ	$+1$	0	-1
0 ("in control")	.16	.68	.16
1 ("out of control")	.5	.49725	.00275

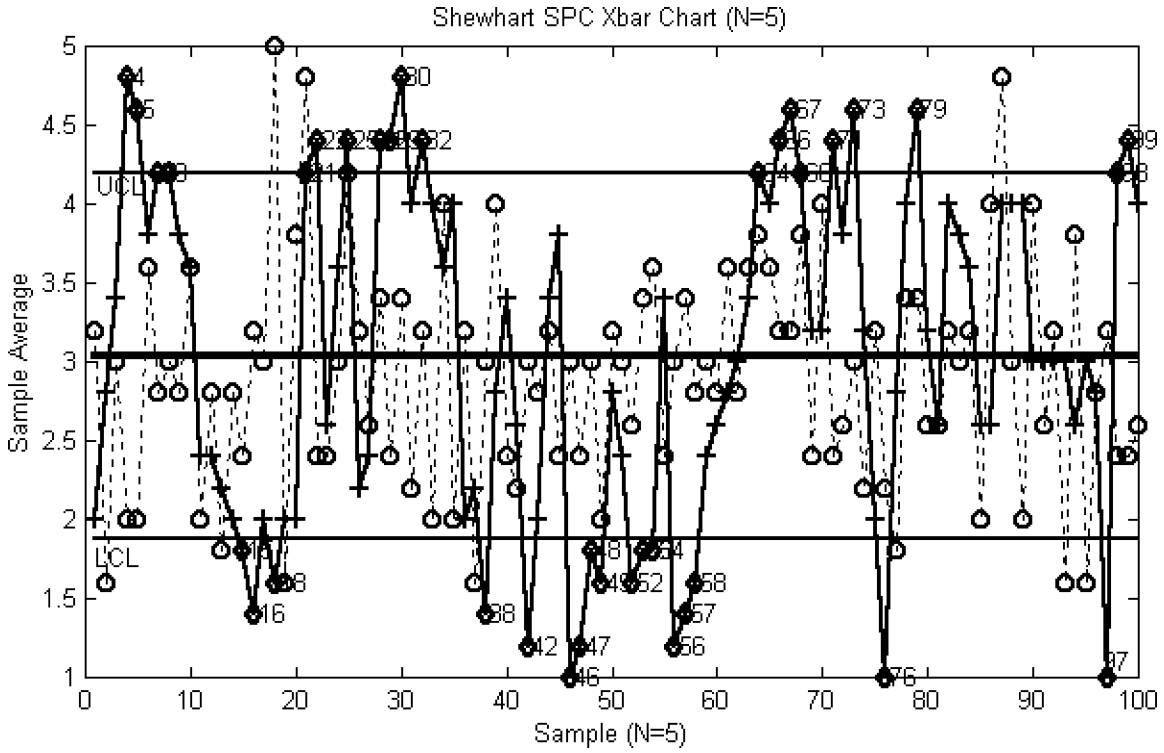


Figure 13. Shewhart $\overline{X}$ Chart for "In Control" Data (μ_0; —+—) and "Out of Control" Data ($\mu = 1$; - -○- -).

of $N = 125$, which equals the sample size used by the CSPC to construct the context trees. Figures 15 and 16 present the $\overline{X}$ and S charts of both the "in control" data (solid line) and the "out of control" data (dashed line). The estimated parameters of the underlying distribution are $\hat{\mu} = \overline{\overline{X}} = 3.0129$ and $\hat{\sigma} = \overline{S}/c_4 = 1.3765/.998 = 1.379$.

As expected, the estimated standard deviation doubled due to the increase in the sample size. Yet many samples generated by the "in control" random walk were out of the control limits. Paradoxically, the samples generated by the "out of control" random walk seem to be steadier and more concentrated around the central line in both charts. An explanation for this phenom-

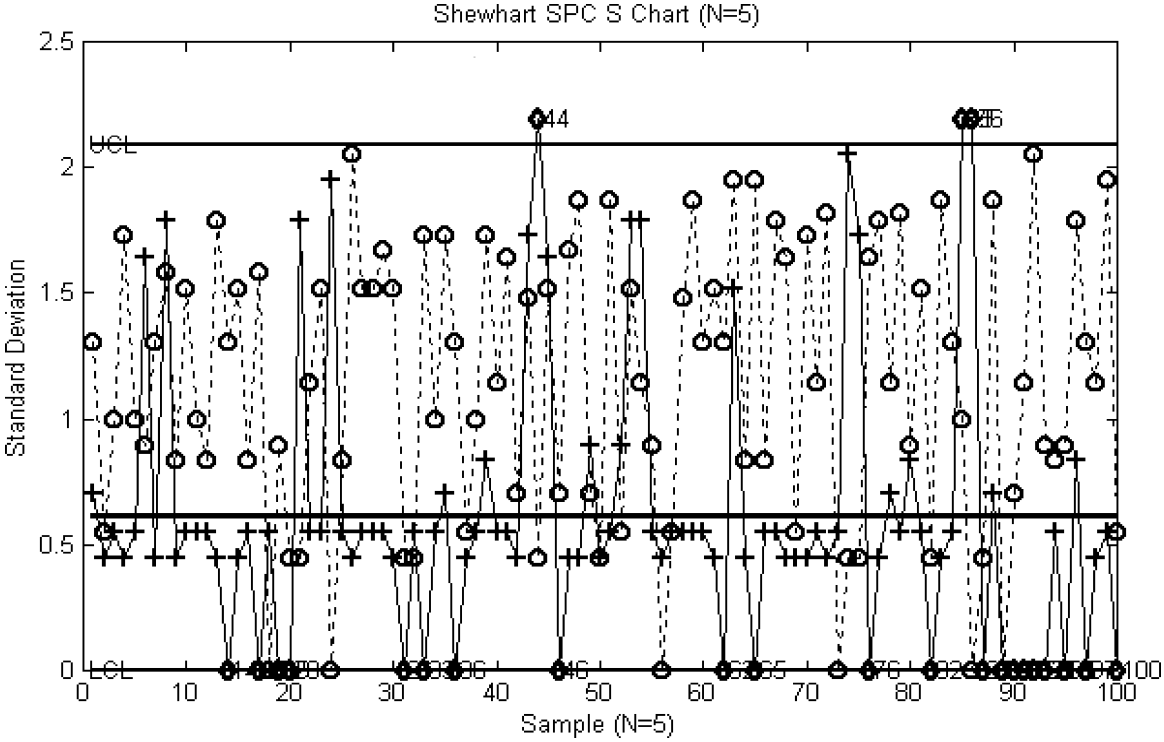


Figure 14. Shewhart S Chart for "In Control" Data (μ_0; —+—) and "Out of Control" Data ($\mu = 1$; - -○- -).

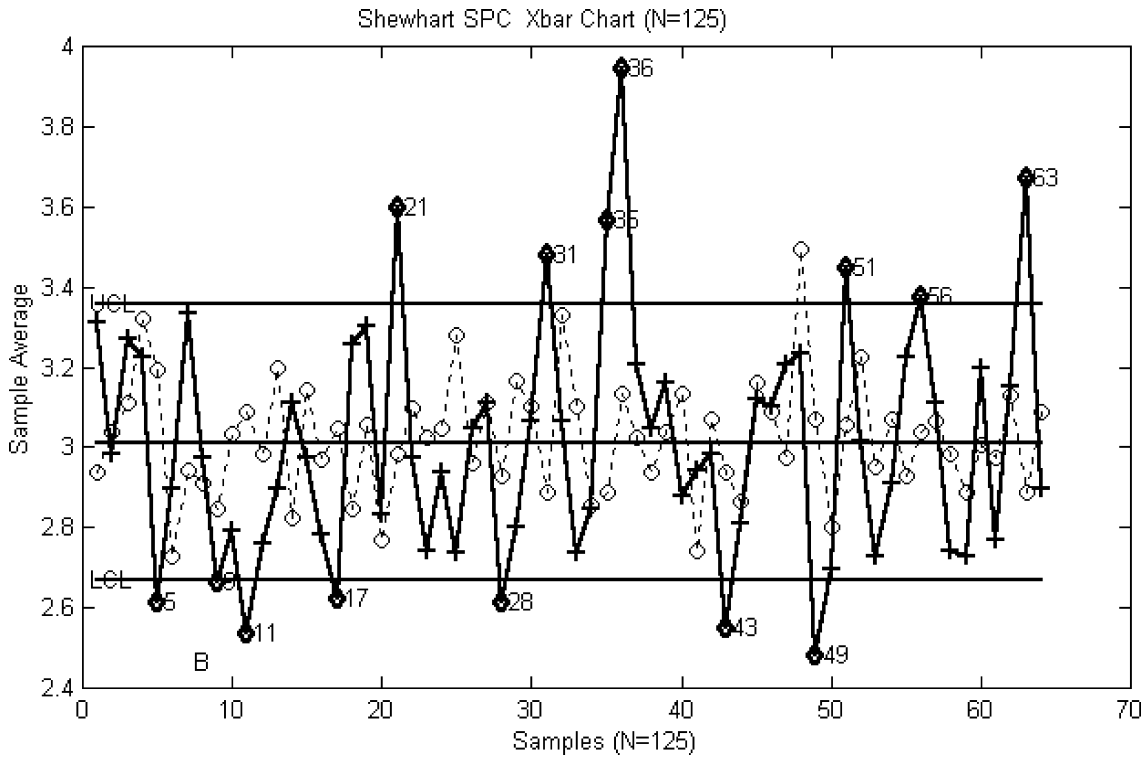


Figure 15. Shewhart $\overline{X}$ Chart for $N = 125$ Sample Size "In Control" Data (μ_0; —+—) and "Out of Control" Data ($\mu = 1$; - -○- -).

enon can be found by observing Table 3, which presents the effect of the mean shift of the underlying normal distribution on the transition probabilities of the random walk.

Note that the probability for a step size of $+1$ in the "out of control" process is larger than that of the "in control" process, whereas the probability for a step size of -1 is much smaller. Consequently, the "out of control" process changes much faster than the "in control" process. This results in a constant intervention of the controller (modeled here by the modulo function), keeping the process values within the fixed range and maintain-

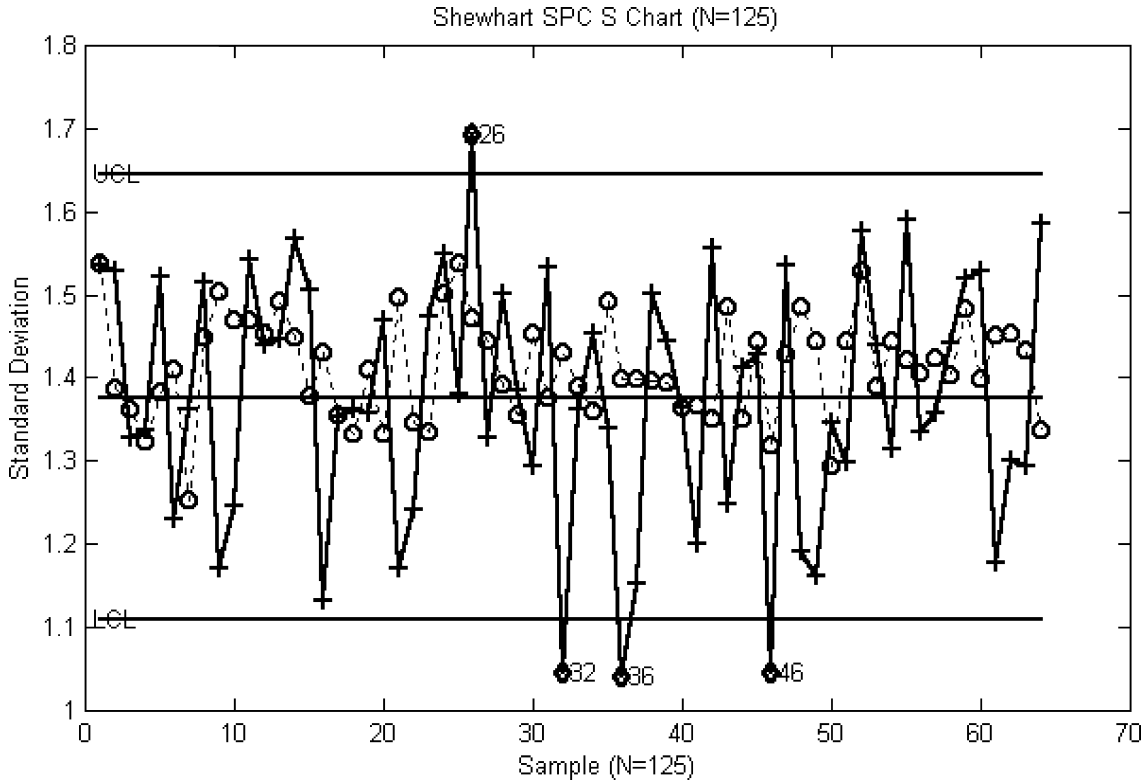


Figure 16. Shewhart S Chart for $N = 125$ Sample "In Control" Data (μ_0; —+—) and "Out of Control" Data ($\mu = 1$; - -○- -).

ing a smaller variance of the sample averages. This phenomenon is presented by the small fluctuation of both the "out of control" average in Figure 15 and the "out of control" standard deviation in Figure 16, although the average standard deviation is larger. In the "in control" case, where the step size probability of $+1$ and -1 is equal, the controller's intervention occurs in fewer cases, and the process remains around the same values for neighboring observations. This results in a greater fluctuation between samples, as seen in Figures 13 and 14 presenting the Shewhart SPC for a sample size of $N = 5$.

Although it has been shown that the Shewhart method can sometimes be implemented on processes that deviate from its underlying assumptions (see, e.g., Shore 2000), this is not the case for state-dependent processes. Shewhart SPC is effective only when changes in the transition probabilities of a state-dependent process significantly affect the process mean. When the Markovian property violates the independence assumption, which is in the core of the center limit theorem, the Shewhart charts may be unreliable.

5. CONCLUSIONS

The proposed CSPC extends the scope of conventional SPC methods. It allows the operators to monitor varying-length state-dependent processes as well as independent and linear ones. The CSPC is more generic and less model-biased with respect to time series modeling. We have shown that the ARIMA-based SPC, which is often applied to complicated nonlinear processes, fails in monitoring state-dependent processes. Using the model-generic CSPC does not come without a price, however. The major drawback of CSPC is the relatively large sample size required during the monitoring stage. Therefore, it should be applied primarily to processes with high sampling frequency, such as the buffer-level monitoring process considered here, or image data, as explained in Section 1.2. Note that as frequent automated monitoring of industrial processes becomes common, the amount of available observations increases, whereas the dependence of these observations prevents the implementation of traditional SPC methods. Future research to decrease the required CSPC's sample size includes the use of new comparative statistics other than the KL, the clustering of symbols to reduce the alphabet size, and the development of an overlapping sampling scheme via a sliding window. These developments might shorten the average run length once the underlying process has been changed.

In addition, because the CSPC, as introduced here, is currently limited to discrete and single-dimensional processes, future research can account for a continuous and multidimensional signal space. As an intermediate solution, a clustering algorithm could be used to find the optimal finite set of d clusters. A web server for utilizing Context-Based SPC is available at *http:www.eng.tau.ac.il/∼bengal/.*

ACKNOWLEDGMENTS

This research was partially funded by the Magnet/Consist consortium. The proposed method is patent-pending (U.S. Provisional Patent Application No. 60/269,344, filed February 20, 2001).

APPENDIX A: GLOSSARY

Term	*Notation*	*Description*
Context	s	A context in which the next symbol occurs is defined as the reversed string $s(x^t) = x_t, \ldots, x_{\max\{0,t-k+1\}}$ for some $k \geq 0$.
Context algorithm		A context tree construction algorithm, suggested by Rissanen (1983) and elaborated by Weinberger et al. (1995).
Context-based SPC	CSPC	An SPC method for state-dependent data based on context tree modeling.
Context-tree		An irreducible set of probabilities that fits the symbol sequence x^N generated by a FSM source. The tree assigns a distinguished optimal context for each symbol in the string.
Finite-state machine	FSM	A finite-memory source model of the sequences defined earlier, characterized by the function $s(x^{N+1}) = f(s(x^N), x_{N+1})$.
"In control" reference context tree	$P_0(x, s)$	The estimated version, denoted by $\widehat{P}_0(x, s)$.
Monitored context tree	$\widehat{P}_i(x, s)$	At monitoring point i.
Optimal context	$s \in \Gamma$	The shortest context for which the conditional probability of a symbol given the context is practically equal to the conditional probability of that symbol given the whole data.
Symbol	$x \in X$	A finite set of values that a process variable can assume. For example, finite capacity buffer levels with values of $X = \{0, 1, 2, \ldots, c\}$, where c is the buffer capacity. The symbol size is denoted by $\lvert X\rvert = d$.

APPENDIX B: THE CONTEXT ALGORITHM

The context tree construction algorithm, termed the *context algorithm*, is described here along with a walk-through example. The walk-through example is based on the restricted random-walk process that was presented in Section 4. The context algortihm presented herafter is an extended version of the algorithm *Context* introduced by Rissanen (1983) and modified by Weinberger et al. (1995) and Ben-Gal, Shmilovici, and Morag (2001). The differences are mainly in stages 3 and 4 of the algorithm.

The input of the algorithm is a sequence of observations $x^N = x_1, \ldots, x_N$, with elements $x_t \in \{0, 1, 2, 3, 4\}$, $t = 1, \ldots, N$, defined over a finite symbol set, X, of size $d = 5$. The output of the algorithm is the context tree $\mathbf{T}_N$ that contains the set of optimal contexts, the estimated marginal probabilities of the optimal contexts, and the estimated conditional probabilities of symbols given the optimal contexts. The algorithm stages follow.

Stage 1: Tree Growing and Counter Updating

The first stage in the algorithm constructs the tree from its root upward (for a downward version where the tree is pruned recursively, see Ben-Gal et al. 2002):

Step 1.0. Start with the root as the initial tree, $\mathbf{T}_0$, where its symbol counts $n(x|s_0)$ are initialized to 0.

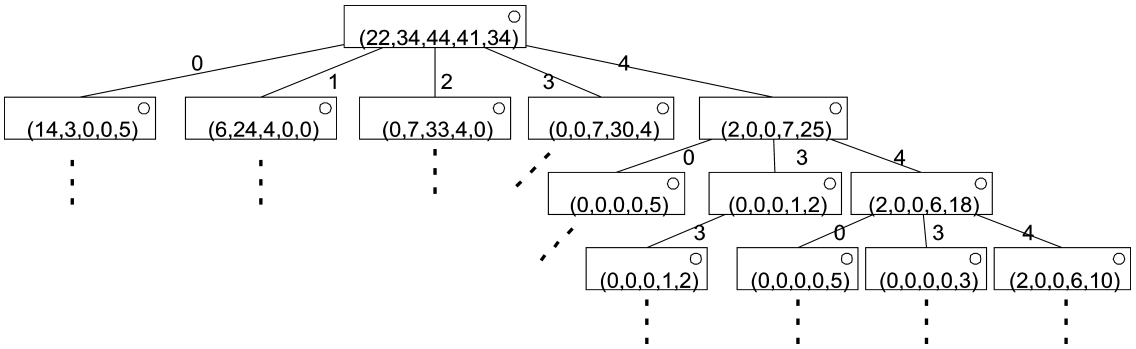


Figure B.1. A Portion of the Counter Context Tree for the Restricted Random-Walk Process for an Input String of N = 175 Observations (after stage 1).

Step 1.1. Recursively, having constructed the tree $\mathbf{T}_t$ from x^t, read the next symbol x_{t+1}. Traverse the tree along the path defined by $x_t, x_{t-1}, \ldots$, and for each node visited along the path, increment the counter value $n(x|s)$ of the symbol x_{t+1} by 1 until reaching the tree's current deepest node, say $x_t, \ldots, x_{t-l+1}$. If necessary, use an initial string preceding x^t to account for initial conditions.

Step 1.2. If the last updated count is at least 1, and $l < m$, where m is the maximum depth, then create new nodes corresponding to x_{t-r}, $l < r \leq m$, and initialize all its symbol counts to 0 except for the symbol x_{t+1}, whose count is set to 1. Repeat this step until the whole past string is mapped to a path for the current symbol, x_{t+1}, or until m is reached. Here r is the depth of the new deepest node, reached for the current path, after completing step 1.2.

Running Example. We demonstrate the construction of a context tree for an example string, $x^6 = 4, 4, 4, 3, 3, 2$, from the restricted random-walk process of Section 4. The string comprises a symbol set of $d = 5$, and its length is $N = 6$. Table B.1 presents the tree-growing and counter-update process. Note that s is the reverse string, $s = x_t, x_{t-1}, \ldots$.

Figure B.1 presents a portion of the counter context tree for a string of $N = 175$ observations generated by the restricted random-walk process described in Section 4.

Stage 2: Tree Pruning

The second stage in the algorithm prunes the tree to obtain the optimal contexts of T_N. This is performed by keeping the deepest nodes w in T_N that practically satisfy (3). The following two pruning rules apply:

- Pruning rule 1. The depth of node w denoted by $|w|$ is bounded by a logarithmic ratio between the length of the string and the number of symbol types, that is, $|w| \leq \log(t+1)/\log(d)$.
- Pruning rule 2. The information obtained from the descendant nodes, sb, $\forall b \in X$, compared with the information obtained from the parent node s, is larger than a "penalty" cost for growing the tree (i.e., of adding a node). The driving principle is to prune a descendant node having a distribution of countervalues similar to that of the parent node. In particular, calculate $\Delta_N(sb)$, the (ideal) *code-length difference* of the descendant node sb, $\forall b \in X$,

$$\Delta_N(sb) = \sum_{x \in X} n(x|sb) \log\left(\frac{\widehat{P}(x|sb)}{\widehat{P}(x|s)}\right) \qquad \text{(B.1)}$$

and then require that $\Delta_N(w) > C(d+1)\log(t+1)$, where logarithms are taken to base 2; C is the pruning constant tuned to process requirements (with default $C = 2$ as suggested in Weinberger et al. 1995). This process is extended to the root node with $\Delta_N(x^0) = \infty$.

Running Example. Table B.2 presents the pruning stage for the string, $x^6 = 4, 4, 4, 3, 3, 2$, for which the counter context tree is constructed in Table B.1.

For the short string $x^6 = 4, 4, 4, 3, 3, 2$, the difference in distribution is not sufficient to support an extension of the tree to two levels. Hence the level-one nodes are trimmed, and the dependence of the process data is expressed by the root node.

Figure B.2 presents the pruned counter context tree constructed by applying the first two stages of the context algorithm on a string containing $N = 175$ observations from the restricted random-walk process of Section 4. The counter context tree for this string is presented in Figure B.1.

Notice that the pruned tree of Figure B.2 is of depth 1. Recall that the restricted random-walk process of Section 4 is a Markov chain of order 1 (see the state transition diagram in Fig. 5). Therefore, the pruning stage of context algorithm identified the process–data dependence.

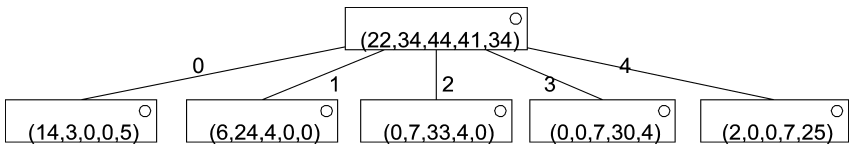


Figure B.2. The Pruned Counter Context Tree of the Restricted Random-Walk Process for an Input String of N = 175 Observations (after stage 2).

Table B.1. Tree Growing and Counter Updating Stage in Context Algorithm for String $x^6 = 4, 4, 4, 3, 3, 2$

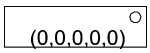
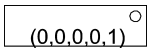
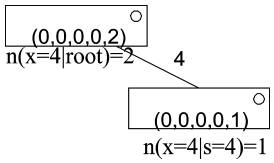
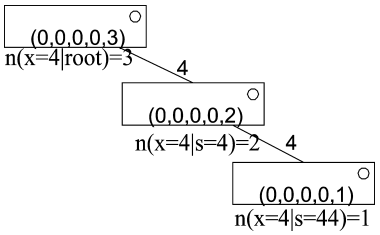
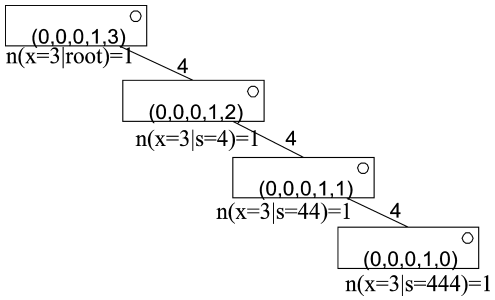
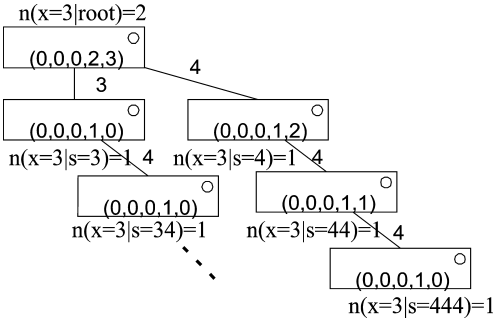
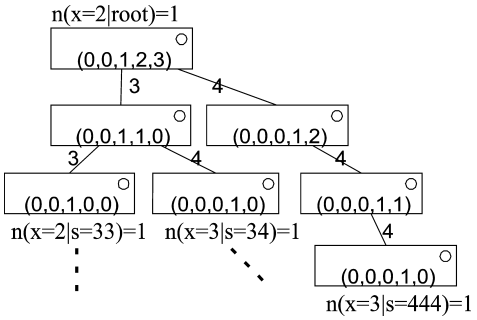
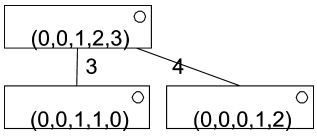
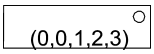
Steps	*Tree*	*Description*
Step 0: T_0		Initialization: The root node, λ, denotes the empty context.
Step 1: T_1 $x^1 = 4$		The only context for the first symbol is λ; the counter $n(x = 4\|\lambda)$ was incremented by 1.
Step 2: T_2 $x^2 = 4, 4$		The counters $n(x = 4\|\lambda)$ and $n(x = 4\|s = 4)$ are incremented by 1. The node of the context $s = 4$ is added to accumulate the counts of symbols given the context $s = 4$.
Step 3: T_3 $x^3 = 4, 4, 4$		The counter of symbol 4 is incremented by 1 in the nodes from the root along the path defined by the past observations. In this case, the counters $n(x = 4\|\lambda)$ and $n(x = 4\|s = 4)$ are incremented by 1. A new node is added for the context $s = 44$, and $n(x = 4\|s = 44) = 1$.
Stage 4: T_4 $x^4 = 4, 4, 4, 3$		The counters $n(x = 3\|\lambda)$, $n(x = 3\|s = 4)$ and $n(x = 3\|s = 44)$ are incremented by 1, because the past contexts are $s = \lambda$, $s = 4$, and $s = 44$. A new node is added for the context $s = 444$ of the observation $x_4 = 3$.
Stage 5: $x^5 = \ldots, 4, 3, 3$		Add new nodes for the contexts $s = 3$, $s = 34$, and $s = 344$. Update the counter of the symbol $x = 3$ from the root to the deepest node on the path of past observations.
Stage 6: $x^6 = \ldots, 3, 3, 2$		Update the counter of the symbol $x = 2$ from the root to the deepest node on the path of past observations. Add the contexts $s = 33$ and so on.

Table B.2. Pruning Stage in the Context Algorithm for the String $x^6 = 4, 4, 4, 3, 3, 2$

Rule	*Tree*	*Description*
Rule 1		Rule 1: Maximum tree depth $\leq \log(t)/\log(d) = \log(6)/\log(5) = 1.11$. The maximum tree depth is of level 1. Thus all nodes of level 2 and below are trimmed.
Rule 2		Rule 2: For the rest of the nodes in level one and the root, we apply trimming rule 2. The threshold for $C = 2$ is $\Delta_6(u) > 2(d+1)\log(t+1) = 33.7$. For each of the nodes, $\Delta_6(sb = \lambda 3) = 0 + 0 + 1 \cdot \log\left(\frac{.5}{1/6}\right) + 1 \cdot \log\left(\frac{.5}{2/6}\right) + 0 = 2.17$ and $\Delta_6(sb = \lambda 4) = 0 + 0 + 0 + 1 \cdot \log\left(\frac{1/3}{2/6}\right) + 2 \cdot \log\left(\frac{2/3}{3/6}\right) = .83.$ The code-length difference is below the threshold; hence the first level nodes are trimmed.

Stage 3: Selection of Optimal Contexts

In this stage the set of optimal contexts, Γ, containing the S shortest contexts satisfying (3) is specified. An optimal context can be either a path to a leaf (a node with no descendants) or a partial leaf in the tree. A *partial leaf* is defined for an incomplete tree. It is a node that is not a leaf; however, for certain symbol(s) its path defines an optimal context satisfying (3). [For other symbols, (3) is not satisfied, and a descendant node(s) has to be created.] The set of optimal contexts is specified by applying the following rule:

$$\Gamma = \left\{ s : \sum_{x \in X} \left(n(x|s) - \sum_{b \in X} n(x|sb) \right) > 0 \right\} \qquad \forall s \in T_t, \quad \text{(B.2)}$$

which means that Γ contains only those contexts that are not part of longer contexts. When the inequality in (B.2) turns into equality, that context is fully contained in a longer context and thus is not included in Γ. Note that in each level in the tree, there is one context that does not belong to a longer context due to the initial condition and thus does not satisfy (B.2). Such inconsistency can be solved by introducing an initiating symbol string, as suggested by Weinberger et al. (1995). In summary, Γ contains all of the leaves in the tree and the partial leaves satisfying (B.2) for certain symbols.

Running Example. Applying (B.2) to the pruned counter context tree presented in Figure B.2 results with five optimal contexts, $\Gamma = \{0, 1, 2, 3, 4\}$. All of the contexts in this case are leaves. Note that the root node is not an optimal context, because it is fully contained in its descendant nodes.

Stage 4: Estimation of the Context Tree Probability Parameters

This stage comprises three steps:

1. The probabilities of optimal contexts are estimated and denoted by $\widehat{P}(s)$, $s \in \Gamma$.
2. The conditional probabilities of symbols given the optimal contexts are estimated and denoted by $\widehat{P}(x|s)$, $x \in X$, $s \in \Gamma$.
3. The estimated joint probabilities of symbols and optimal contexts are calculated $\widehat{P}(x, s)$, $x \in X$, $s \in \Gamma$.

Step 4.1. Given the set of optimal contexts and the pruned counter tree, the probability of optimal contexts in the tree, $\widehat{P}(s)$, $s \in \Gamma$, is estimated by their frequency in the string,

$$\widehat{P}(s) = \frac{n(s)}{\sum_{s \in \Gamma} n(s)} = \frac{\sum_{x \in X}(n(x|s) - \sum_{b \in X} n(x|sb))}{\sum_{s \in \Gamma}\sum_{x \in X}(n(x|s) - \sum_{b \in X} n(x|sb))}, \qquad \forall x \in X, \ s \in \Gamma, \quad \text{(B.3)}$$

where $n(s)$ is the sum of the symbol counters in the corresponding leaf (or partial leaf) belonging to the optimal context s and not to a longer context, sb, $b \in X$. Each symbol in the string thus belongs to one out of S optimal contexts, each of which contains $n(s)$ symbols. The allocation of symbols of a sufficiently long string to distinctive optimal contexts is approximated by the multinomial distribution.

Running Example. The estimated probabilities of optimal contexts in Figure B.2 are

$$\left\{\widehat{P}(0), \widehat{P}(1), \widehat{P}(2), \widehat{P}(3), \widehat{P}(4)\right\} = \left\{\frac{22}{175}, \frac{34}{175}, \frac{44}{175}, \frac{41}{175}, \frac{34}{175}\right\}.$$

Step 4.2. Once the symbols in the string are partitioned among S optimal contexts, the conditional probabilities of symbol types given an optimal context are estimated by their frequencies in the respective substring (Weinberger et al. 1995),

$$\widehat{P}(x|s) = \frac{n(x|s) - \sum_{b \in X} n(x|sb) + 1/\nu}{n(s) + d/\nu}, \qquad \forall x, b \in X, \ s \in \Gamma, \quad \text{(B.4)}$$

where $\nu = 2$ is the default value. The distribution of symbol types in a given optimal context is thus approximated by another multinomial distribution. Equation (B.4) with finite $\nu > 0$ assigns positive probabilities to realizations that never appeared in the sample string yet can occur in reality. We call this the *predictive approach*, which, similarly to some Bayesian approach, assigns a nonzero a posteriori probabilty even though the a priori probability is 0.

An alternative approach is a *nonpredictive approach*, where $\nu \to \infty$ and $0/0 \equiv 0$. The choice from among these alternative procedures depends both on the knowledge regarding the system states and on the length of the string used to construct the context tree. However, in the latter nonpredictive case, the number of degrees of freedom is adapted according to the number

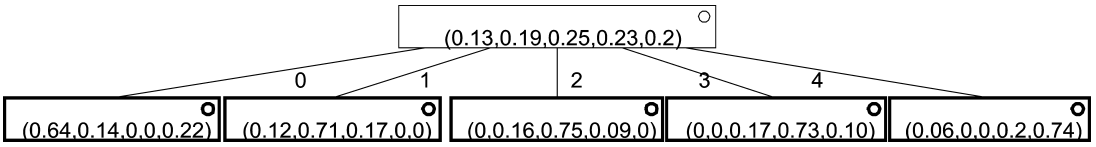


Figure B.3. The Context Tree Containing Vectors of Conditional Probabilities $P(x|s)$ as Obtained from the Counter Context Tree in Figure B.2. Optimal contexts are represented by the bolded frame.

of categories that are not equal to 0 (see, e.g., May and Johnson 1997), because the multinomial theory stands for nonzero probability categories.

Running Example. Figure B.3 presents the estimated conditional probabilities of symbols given contexts. These estimates are generated by applying the nonpredictive approach to the counter context tree presented in Figure B.2. For example, the conditional probability of a symbol type $x \in \{0, 1, 2, 3, 4\}$, given the context $s = 0$, is estimated as $\widehat{P}(x|0) = (\frac{14}{22}, \frac{3}{22}, 0, 0, \frac{5}{22})$. The probabilities of symbols in the root are also presented for general information.

Step 4.3. The joint probabilities of symbols and optimal contexts that represent the context tree in its final form are evaluated $\widehat{P}(x, s) = \widehat{P}(x|s) \cdot \widehat{P}(s)$, $x \in X$, $s \in \Gamma$.

General Note. It is possible to consider nonsequential contexts. Rissanen (1983) proposed using a permutation function to map the correct dependency order.

[Received April 2001. Revised May 2003.]

REFERENCES

Alwan, L. C., Ebrahimi, N., and Soofi, E. S. (1998), "Information Theoretic Framework for Process Control," *European Journal of Operations Research*, 111, 526–542.

Alwan, L. C., and Roberts, H. V. (1988), "Time-Series Modeling for Statistical Process Control," *Journal of Business and Economics Statistics*, 6, 87–95.

Apley, D. W., and Shi, J. (1999), "The GLRT for Statistical Process Control of Autocorrelated Processes," *IIE Transactions*, 31, 1123–1134.

Ben-Gal, I., Arviv, S., Shmilovici, A., and Grosse, I. (2002), "Promoter Recognition via Varying Order Markov Models," unpublished technical report, TA Univ. CIM lab.

Ben-Gal, I., Shmilovici, A., and Morag, G. (2001), "Design of Control and Monitoring Rules for State Dependent Processes," *The International Journal for Manufacturing Science and Production*, 3, 85–93.

Ben-Gal, I., and Singer, G. (2001), "Integrating Engineering Process Control and Statistical Process Control via Context Modeling," unpublished manuscript.

Bitran, G. R., and Dasu, S. (1992), "A Review of Open Queueing Network Models of Manufacturing Systems," *Queueing Systems: Theory and Applications*, Special Issue on Queueing Models of Manufacturing Systems, 12, 95–133.

Boardman, T. J., and Boardman, E. C. (1990), "Don't Touch That Funnel," *Quality Progress*, December 1990.

Box, G. E. P., and Jenkins, G. M. (1976), *Times Series Analysis, Forecasting and Control*, Oakland, CA: Holden-Day.

Bromaghin, J. F. (1993), "Sample-Size Determination for Interval Estimation of Multinomial Probabilities," *The American Statistician*, 47, 203–206.

Buzacott, J. A., and Yao, D. (1986a), "On Queueing Network Models of Flexible Manufacturing Systems," *Queueing Systems: Theory and Applications*, 1, 5–27.

——— (1986b), "Flexible Manufacturing Systems: A Review of Analytical Models," *Management Science*, 32, 890–905.

Cochran, W. G. (1952), "The Chi-Square Test of Goodness of Fit," *Annals of Mathematical Statistics*, 23, 315–345.

Elliott, R. J., Lakhdaraggoun, and Moore, J. B. (1995), *Hidden Markov Models*, New York: Springer-Verlag.

Gershwin, S. B. (1994), *Manufacturing System Engineering*, Engelwood Cliffs, NJ: Prentice-Hall.

Harris, T. J., and Ross, W. H. (1991), "Statistical Process Control Procedures for Correlated Observations," *The Canadian Journal of Chemical Engineering*, 69, 48–57.

Kullback, S. (1959), *Information Theory and Statistics*, New York: Wiley.

Lu, C. W., and Reynolds, M. R. (1999), "EWMA Control Charts for Monitoring the Mean of Autocorrelated Processes," *Journal of Quality Technology*, 31, 166–188.

——— (1997), "Control Charts for Monitoring Processes with Autocorrelated Data," *Nonlinear Analysis: Theory, Methods & Applications*, 30, 4059–4067.

May, W. L., and Johnson, W. D. (1997), "Properties of Simultaneous Confidence Intervals for Multinomial Proportions," *Communications in Statistics, Part B—Simulation and Computation*, 26, 495–518.

Montgomery, D. C., and Mastrangelo, C. M. (1991), "Some Statistical Process Control Methods for Autocorrelated Data," *Journal of Quality Technology*, 23, 179–193.

Ohler, U., and Niemann, H. (2001), "Identification and Analysis of Eukaryotic Promoters: Recent Computational Approaches," *TRENDS in Genetics*, 17, 56–60.

Rissanen, J. (1983), "A Universal Data Compression System," *IEEE Transactions on Information Theory*, 29, 656–664.

——— (1999), "Fast Universal Coding With Context Models," *IEEE Transactions on Information Theory*, 45, 1065–1071.

Runger, G., and Willemain, T. (1995), "Model-Based and Model-Free Control of Autocorrelated Processes," *Journal of Quality Technology*, 27, 283–292.

——— (1996), "Batch-Means Control Charts for Autocorrelated Data," *IIE Transactions*, 24, 66–80.

Runger, G., Willemain, T., and Prabhu, S. (1995), "Average-Run-Lengths for CUSUM Control Charts Applied to Residuals," *Communications in Statistics, Part A—Theory and Methods*, 24, 273–282.

Shore, H. (2000), "General Control Charts for Attributes," *IIE Transactions*, 32, 1149–1160.

Wardell, D. G., Moskowitz, H., and Plante, R. D. (1994), "Run-Length Distributions of Special-Cause Control Charts for Correlated Processes," *Technometrics*, 36, 3–17.

Weinberger, M., Rissanen, J., and Feder, M. (1995), "A Universal Finite Memory Source," *IEEE Transactions on Information Theory*, 41, 643–652.

Zhang, N. F. (1998), "A Statistical Control Chart for Stationary Process Data," *Technometrics*, 40, 24–38.