CSPC: A Monitoring Procedure for State Dependent Processes

Irad Ben-Gal[1]^, Gail Morag^, Armin Shmilovici*

^Department of Industrial Engineering, Tel-Aviv University
Ramat-Aviv, Tel-Aviv 69978, Israel
*Department of Information Systems Engineering
Ben-Gurion University, Beer-Sheva 84105, Israel

ABSTRACT

Most Statistical Process Control (SPC) methods are not suitable for monitoring non-linear and state-dependent processes. This paper 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 behaviour of the process; the second is a *monitored tree*, generated periodically from a sample of sequenced observations, that represents the behaviour 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.

Keywords: Statistical Process Control, Context Tree, Kullback-Leibler statistic

[1]Corresponding Author
Bengal@eng.tau.ac.il

CSPC: A Monitoring Procedure for State Dependent Processes

1. Introduction

1.1. SPC 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: i) there exists an *a priori* knowledge of the underlying distribution (often, observations are assumed to be normally distributed); and ii) the observations are independent and identically distributed (i.i.d.). In practice, these assumptions are frequently violated in various industrial processes. This paper will present a novel SPC method which 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 ARIMA, versus methods that are termed *model-generic*. The latter methods try to estimate the underlying model with minimum *a priori* assumptions.

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

///* insert Figure 1 about here ***///**

The Information Theoretic Process Control (ITPC) is an *independent-data* based and *model-generic* SPC method proposed by Alwan, Ebrahimi and Soofi (1998). It utilizes 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 that 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 extensively implemented 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) and 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.

The majority of *model-specific* methods for *dependent data* are time-series based. 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 traditional SPC schemes to the stream of residuals. In particular, the ARIMA (Auto Regressive Integrated Moving Average) 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, to which 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) and 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) where 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 the ARIMA parameters. Moreover, the method performs poorly for a large positive autocorrelation, since the mean shift tends to stabilize rather quickly to a steady state value, thus the shift is poorly manifested on the residuals (see Wardell, Moskowitz and Plante (1994) and 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 these models are based on *a priori* knowledge regarding the data source and require an explicit parameter estimation. It was demonstrated in Runger and Willemain (1995) that for certain autocorrelated processes, the use of traditional SPC yields an improved performance compared to ARIMA-based methods. The Generalized Likelihood Ratio Test – GLRT – method proposed by Apley and Shi (1999) takes advantage of the residuals 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, in Apley and Shi (1999) and in Runger and Willemain (1995) it is emphasized that modeling errors of ARIMA parameters have 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 since they allow effiecient 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 to use 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 showed that Montgomery's approximation performed poorly. He proposed to employ 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 in order that the autocorrelation among batches reduces to zero. 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. While this can be true in some industrial environments, such an approach cannot capture non-linear process dynamics that depend on the state in which the system operates, for example, processes that are described by Hidden Markov Models (HMM) (Elliot, Lakhdaraggoun and Moore (1995)).

This paper presents the Context-based SPC (CSPC) methodology for state-dependent data. It 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 *et al.* 1995). The context-tree provides a compact model of the sequence of observations even for complex, non-linear processes such as HMM 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 is a *reference tree* that represents the 'in control' reference behaviour of the process; the second is a *monitored tree*, generated periodically from a sample of sequenced observations, that represents the behaviour 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* (non-parametric). 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 non-linear state-dependent processes. Later we show that even special SPC methods that were designed to handle correlated data fail to monitor a non-linear 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: first, it requires a relatively large amount of data, and second, it is limited to handle discrete measures over a finite alphabet. Albeit, there are many areas to which these limitations do not apply. Three examples for such potential areas are briefly described next.

***Image monitoring*:** Rissanen (1983) in his original paper, proposed to apply 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 them 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 constructs proteins; intron sequences which are transcribed yet not 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 E. Coli sequences of DNA-spaced reading-frames. The upper series represent promoter sequences and the lower series represent non-promoter 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 principally is similar to SPC schemes. Further details on this promising direction of research are beyond the scope of the paper, and can be found in (Ben-Gal et al, 2002).

///* insert Figure 2 about here ***///**

Production monitoring via buffers: A common practice in the analysis of production systems is the use of 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 (among the numerous publications, see for example, 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. Part of the reasons are that the statistical behavior of buffer levels is complex, highly non-linear and often cannot be described by a known stochastic process, and therefore, is inadequate for traditional SPC methods, as will be described 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, since 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 which 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 enough to monitor only a few buffers.

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

The rest of the paper is organized as follows: Section 2 introduces the theoretical background for the context-tree model and the principles of its construction (a detailed construction algorithm and a walk-through example are presented in appendix 2); Section 3 develops the control limits for the CSPC methodology based on the Kullback-Leibler (KL) statistic and presents the CSPC methodology; Section 4 illustrates the CSPC by a detailed numerical example and compares it to conventional SPC methods; Section 5 concludes with some discussion.

2. Modeling process dependence with context trees.

In this section we introduce the context trees model for state-dependent data and the concepts of its construction algorithm following the definitions and notations in Rissanen (1983) and in Weinberger *et al.* (1995). A detailed walk-through example presenting the context-tree construction is given in appendix 2. Appendix 1 includes a glossary of terms used in this paper.

Consider a sequence (string) of observations $x^N = x_1,...,x_N$, with elements x_t $t = 1,..,N$ 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 $X = \{0,1,2,...,c\}$ 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,… is defined over the set $\{X^N\}$ of all stationary sequences of length *N*, such that the marginality condition

$$\sum_{x \in X} P_{N+1}(x^N x) = P_N(x^N) \tag{2.1}$$

holds for all *N*; $x^N x = x_1,...,x_N, x$; and $P_0(x^0) = 1$ where x^0 is the empty string. For simplification of notations, the sub-index will be omitted, so that $P_N(x^N) \equiv P(x^N)$.

One could opt to find a model that assigns the probability measure (2.1). A possible finite-memory source model of the sequences defined above 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,

$$s(x^{N+1}) = f(s(x^N), x_{N+1}) \tag{2.2}$$

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 $f : \Gamma \times X \rightarrow \Gamma$ 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 is equal to the conditional probability to obtain the symbol given the past state, implying that

$$P\left(x \mid x^{N}\right) = P\left(x \mid s\left(x^{N}\right)\right). \tag{2.3}$$

A special case of FSM is the Markov process. The Markov process satisfies (2.2) and is distinguished by the property that for a *k*th-order Markov process $s\left(x^{N}\right) = x_{N},...,x_{N-k+1}$. 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 (2.3) 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, since 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 (non-continuous) 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* that was suggested by Rissanen (1983) for data compression purposes and modified later in Weinberger *et al.* (1995). The tree presentation of a finite-memory source is advantageous since 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 efforts 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 seen later.

A *context*, $s\left(x^{t}\right)$, 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, since here, they follow similar properties),

$$s\left(x^{t}\right) = x_{t},...,x_{\max\{0,t-k+1\}} \tag{2.4}$$

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 since the symbols observed prior to x_{t-k+1} do not affect the occurrence probability of x_{t+1}. For the set of *optimal contexts*, $\Gamma = \{s : \text{shortest contexts satisfying} (2.3)\}$, *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, i.e., nearly satisfying (2.3). 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 one does not need to fix *k* 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 above 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 between sequences of observations and identifying whether they are generated from the same source. Graphically, the context-tree is a *d*-ary tree which 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 2 for a proper definition of a partial leaf).

///* insert Figure 3 about here ***///**

Figure 3: Illustration of a context-tree with *S*=5 optimal contexts.

Figure 3 exemplifies a *context-tree* that was constructed from a sequence of observed buffer levels in a production line. Since in this case the buffer has a finite capacity of c = 2, there are d = 3 symbol types, where observation, $x_t \in \{0,1,2\}$, refer to the number of parts in the buffer at time *t*. Following the context algorithm, which is detailed in Appendix 2, S = 5 optimal contexts are found (marked by bolded 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. Employing the above definitions, the optimal context of the next element, $x_7 = 0$, is $s(x^6) = 1,0,1,0$, i.e., following the reverse string from the root until reaching an optimal context. Accordingly, the probability of x_7 given the context is $P(x_7 = 0 | s(x^6)) = 0.33$. For a detailed example, refer to appendix 2.

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

2.2 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*, resulting 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 (2.3). 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 $\hat{P}(x|s)$, $x \in X$ $s \in \Gamma$, and the estimated marginal probabilities of optimal contexts $\hat{P}(s)$, $s \in \Gamma$, are derived. As noted in Appendix 2, both $\hat{P}(x|s)$ and $\hat{P}(s)$ are asymptotically multinomial distributed and used to obtain the CSPC control limits. The estimated joint probabilities of symbols and optimal contexts, $\hat{P}(x,s) = \hat{P}(x \mid s) \cdot \hat{P}(s)$, $x \in X, s \in \Gamma$, 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 2.

3. CSPC: The context-based SPC

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 $Q_0(x)$:

$$K(Q(x), Q_0(x)) = \sum_{x \in X} Q(x) \log \frac{Q(x)}{Q_0(x)} \geq 0 \tag{3.1}$$

The measure, known later as the Kullback Liebler (KL) measure, is positive for all non-identical pairs of distributions and equals zero 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 $\hat{Q}(x)$, is asymptotically chi-square distributed with *d*-1 degrees of freedom (dof):

$$2N \cdot K\left(\hat{Q}(x), Q(x)\right) \rightarrow \sum_{x \in X} \frac{(n(x) - NQ(x))^2}{NQ(x)} \sim \chi^2_{d-1}, \qquad (3.2)$$

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 $\hat{Q}(x) = n(x)/N$ 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 representing the distance between the conditioning random variable and the other representing the distance between the conditioned random variable (Kullback (1959)):

$$K(Q(x,y), Q_0(x,y)) = \sum_{y \in S} Q(y) \log \frac{Q(y)}{Q_0(y)} + \sum_{y \in S} Q(y) \sum_{x \in X} Q(x \mid y) \log \frac{Q(x \mid y)}{Q_0(x \mid y)} \qquad (3.3)$$

In this paper, we implement the KL measure to detect the relative distance between two context-trees (other distance measure may be used, such as *Jensen-Shannon,* e.g., Ben-Gal et al. (2002)). The first tree, denoted by $\hat{P}_i(x,s)$, represents the monitored distribution of symbols and contexts, as estimated from a string of length N at the monitoring time $i = 1,2,...$. 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, Ebrahimi and Soofi (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 a direct use of the KL measure. New observations are being collected and used for updating the monitored tree counters and its statistics (equations (A.3) (A.4) in appendix 2). A significant change in the monitored process is manifested in the tree counters and its resulting probabilities.

Using (3.3), 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,

$$K\left(\hat{P}_i(x,s),P_0(x,s)\right)=\sum_{s\in\Gamma}\hat{P}_i(s)\log\frac{\hat{P}_i(s)}{P_0(s)}+\sum_{s\in\Gamma}\hat{P}_i(s)\sum_{x\in X}\hat{P}_i(x|s)\log\frac{\hat{P}_i(x|s)}{P_0(x|s)} \qquad (3.4)$$

one 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 $\hat{P}_i(x,s)$ is generated from the same source that generated $P_0(x,s)$ and by using the multinomial approximation (3.2) together with (3.4), we derive the asymptotic probability density function of the KL measure between $\hat{P}_i(x,s)$ and $P_0(x,s)$, i.e., for a long string,

$$\begin{aligned}
&K\left(\hat{P}_i(x,s),P_0(x,s)\right)\rightarrow\\
&\frac{1}{2N}\chi^2_{S-1}+\sum_{s\in\Gamma}\hat{P}_i(s)\cdot\frac{1}{2n(s)}\chi^2_{d-1}=\\
&\frac{1}{2N}\chi^2_{S-1}+\sum_{s\in\Gamma}\frac{n(s)}{N}\cdot\frac{1}{2n(s)}\chi^2_{d-1}=\\
&\frac{1}{2N}\chi^2_{S-1}+\frac{1}{2N}\sum_{s\in\Gamma}\chi^2_{d-1}=\\
&\frac{1}{2N}\left(\chi^2_{S-1}+\chi^2_{S(d-1)}\right)=\frac{1}{2N}\chi^2_{Sd-1},
\end{aligned} \qquad (3.5)$$

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-square distributed with dof depending on the number of symbol types and the number of optimal contexts (the number of dof 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,

$$0\leq 2N\cdot K\left(\hat{P}_i(x,s),P_0(x,s)\right)\leq\chi^2_{Sd-1,1-\alpha}. \qquad (3.6)$$

Thus, the upper control limit (UCL) is the 100(1-α) percentile of the chi-square distribution with ($Sd-1$) degrees of freedom.

The control limit (3.6) has some appealing characteristics:

i) It is a one-sided bound; if the KL value is larger than the UCL, the process is assumed to be ‘out-of-control’ for a given level of significance.

ii) It lumps together all the parameters of the context-tree, in contrast with traditional SPC where each process parameter is controlled separately. Nevertheless, the KL statistic of the tree can be easily decomposed to monitor separately each node in the context-tree. This can be beneficial when seeking for the cause of an ‘out-of-control’ signal.

iii) If *Sd* is large enough, 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 eq. A.4 in appendix 2) where all probability values assigned to any of the symbol types are strictly positive, or with the non-predictive approach (see eq. A.4) by defining $\frac{0}{0} \equiv 0$.

3.2 The CSPC monitoring Procedure

The following steps outline briefly the CSPC monitoring procedure.

Step 1. Obtain the reference context-tree $P_0(x,s)$, either analytically or by employing the context algorithm to a long string of data.

Step 2. For any monitoring point of time take a sample of sequenced observations of size N and generate the monitored tree $\hat{P}_i(x,s)$. 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 equations (A.3), (A.4).

Step 3. Compute the KL estimates, measuring the relative 'distance' between the estimated monitored distributions $\hat{P}_i(x,s)$ and the reference distributions $P_0(x,s)$.

Step 4. Plot the KL statistic value in the control chart against the upper control limit found using (3.6). If the KL value is larger than the UCL it indicates that a significant change has likely occurred in the process. Search for special-cause variability and eliminate it.

Step 5. For the next monitoring point, collect a new string of data, increment $i = i + 1$ and go to Step 2. If there are no data available, stop the monitoring stage.

4. Numerical Example: Buffer monitoring in a production line

Consider a production line with K machines modeled as a network of reliable service-stations (M1, $M_2,\ldots,M_K$) 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 by methods shown in Gershwin (1994). The monitored subsystem consist of two machines, M_k and M_{k+1}, and a buffer B_k with a finite capacity c.

///* insert Figure 4 about here ***///**

Figure 4: A Subsystem of a production line of K machines

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, $b = 0,...,c$, 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 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):

$$\begin{aligned}
&P(x_{t+1} = b | x_t = b-1) = P(0|c) = p_k q_{k+1} \quad b = 1,...,c \\
&P(x_{t+1} = b | x_t = b+1) = P(c|0) = p_{k+1} q_k \quad b = 0,...,c-1 \qquad (4.1)\\
&P(x_{t+1} = b | x_t = b) = 1 - p_k q_{k+1} - p_{k+1} q_k \quad b = 0,...,c
\end{aligned}$$

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 since 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, since 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 machines' 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 'in-control' model

Figure 5 presents the first-order markov diagram of the buffer levels that is obtained by specifying a buffer capacity $c = 4$ and production probabilities $p_k = p_{k+1} = 0.8$. This production probability value (that can be estimated in practice by the production rate of the machine) has been selected for two reasons. First, it represents a relatively high production rate, which is not too high for monitoring purposes, since anomalies at higher production rates are easier to detect. Second, it guarantees a steady state buffer level which is equal to 2 (the steady state probabilities are all equal to 0.2 as seen later in Figure 6) – exactly half of the buffer capacity. Substituting these production probabilities to eq. (4.1) yields a state transition probability where $P(x_{t+1} = b | x_t = b-1) = P(x_{t+1} = b | x_t = b+1) = 0.16$; $P(x_{t+1} = b | x_t = b) = 0.68$, $b = 0,...,4$.

In order to obtain a direct comparison to conventional SPC methods that are 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 i.i.d 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 $z_i \in \{-1,0,1\}$, $i=1,2,\ldots,N$, that represent the change in the buffer level, where $P(z_i = -1) = P(z_i = 1) = 0.16$ and $P(z_i = 0) = 0.68$.
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) Since 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, will permit a straightforward comparison to conventional SPC methods in later steps of the example. Table 1 exemplifies a short realization of generating process for *N*=5.

///* insert Table 1 and Figure 5 about here ***///**

Table 1: Feedback-controlled process generation example

Figure 5: State transition diagram for the process.

An analytical context-tree (Figure 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.

Since the analytical model is often unknown in practice, let us illustrate the convergence of the estimated context-tree, $\hat{P}_0(x,s)$, 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 zero. Figure 7 presents the asymptotic convergence of the KL 'distance' between $\hat{P}_0(x,s)$ and $P_0(x,s)$ as a function of N – the string length.

///* insert Figure 6 and Figure 7 about here ***///**

Figure 6: The analytically derived singled-level context-tree.

Figure 7: The KL value between the analytical tree and the estimated tree as a function of the input string length *N*

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 exhibits the rapid convergence of context algorithm to the 'true' model. The dotted line indicates the weighted upper control limit, $\chi^2_{(24,0.9975)}\big/(2\cdot N)$, as derived from (3.6). Notice that for *N*>325, the KL value is constantly below the weighted UCL. Figure 7 can assist an experimenter in determining the string length, *N*, required for a satisfactory estimation of the reference 'in-control' context-tree. In particular, note that approximately for *N*<300 the KL measure is in its transient mode, while 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 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) summarize several techniques to determine the sample size required for the 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 0.5):

$$N = 1 + \operatorname{int}\left(\max_{x \in X} \left[\frac{0.25 z^2_{(1-\alpha_x/2)}}{\Delta^2_x} - z^2_{(1-\alpha_x/2)} \right] \right) \tag{4.8}$$

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 $x \in X$, $\sum_{x \in X} \alpha_x \le \alpha$; and int(·) stands for the integer function. He provided a table of recommended sample sizes based on (4.8) for the case of equal interval width and equal significance levels, i.e., Δ_x = Δ and $\alpha_x = \frac{\alpha}{d}, \forall x \in X$. For example, for $\alpha = 0.05$, $\Delta = 0.05$ 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=1000 observations determined by the results in Figure 7. The predictive approach (see eq. A.4 in appendix 2) was employed to compute the estimated conditional probabilities $\hat{P}_0(x \mid s), \forall x \in X, s \in T_N$.

///* insert Figure 8 about here ***///**

Figure 8: Estimated reference context-tree as a result of the implementation of context algorithm to N=1000 observations.

To conclude, the context algorithm performed well with respect to both the monitored tree structure and its estimated probability measures. The algorithm accurately identified $\hat{P}_0(x,s)$ 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.

4.2 CSPC procedure: the monitoring stage

Based on the structure of the reference context-tree, the UCL was calibrated to obtain a type I error probability of $\alpha = 0.0025$, which corresponds to the typical choice of "3 sigma" in traditional SPC. Following (3.6), the UCL for the KL statistic was determined to $\chi^2_{(S\cdot d-1,1-\alpha)} = \chi^2_{(24,0.9975)} = 48$.

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

i. Shifts in the *standard deviation* of the underlying normal process, denoted by $\sigma' = \lambda \cdot \sigma_0$, where $\sigma_0 = 1$; and λ taking respectively the values of 1 ('in-control'), 0.5 , 1.5 and 2.

ii. 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 varying between 0 to 3.

During the monitoring stage of both scenarios consecutive strings of length of *N*=125 data points were used to generate fifty monitored context-trees, $\hat{P}_i(x,s)\ i = 1,2,...,50$, for each shifted process. A segment of 50 runs was used for a clear graphical presentation, enabling to sketch both the 'in-control' and 'out-of-control' charts on the same graph. String lengths of *N*=125 adhere to the Chi-square sampling principle suggested by Cochran (1952) requiring that at least eighty percent of the sampling bins (corresponding in this case to the non-zero 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 are to be investigated in future research.

***Scenario* 1: Shifts in the standard deviation of the underlying normal distribution**

The CSPC monitoring procedure (outlined in Section 3.2) was applied to shifts in the standard deviation of the underlying normal distribution. Fifty runs with respective λ values of 1, 1.5, 2 and 0.5 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.

///* insert Figure 9, Figure 10 and Table 2 about here ***///**

Figure 9: Shifts in the process underlying normal standard deviation, λ=1, 1.5, 2
(a portion of 50 runs is presented).

Figure 10: Shift in the process underlying normal standard deviation, λ=0.5, 1
(a portion of 50 runs is presented)

Table 2: Performance of CSPC during scenario No. 1

As can be seen, for the 'in-control' process (λ=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 zero. For shifted processes, the rate of successful shift detection by the CSPC is relative to the transition probability change. A standard deviation shift of λ=0.5 decreased the transition probability significantly from 0.16 to 0.02 and resulted in 100% of the runs out of the control limits (Figure 10). However, a standard deviation shift of λ=1.5 increased the transition probability from 0.16 to 0.25 and resulted in 20% of the runs above the UCL (Figure 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 how to shorten the CSPC's sample size are given in the conclusions section.

Time-series-based SPC methods are designed to detect shifts either in process means or in the process standard deviation. As exemplified in the next scenario, the CSPC is capable of identifying both types of shifts by using a single monitoring chart, since both of them 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 zero to three standard deviations ($\delta \in$ [0,3]). For comparison purpose, we also plotted an OC curve for a traditional $\overline{X}$ statistic of an i.i.d 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 seen 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.

///* insert Figure 11 about here ***///**

Figure 11: The Operating Characteristics curve for mean shifts

4.3 CSPC procedure: comparison to conventional SPC methods

In their seminal book, Box and Jenkins (1976) applied a variety of time series models to discrete and non-linear series, such as monthly stocks closing prices, yearly sunspot numbers and rounded-up yields of a chemical process. They have shown that simple ARIMA models, such as AR(1) or IMA(1,1), can effectively filter a wide range of auto-correlated processes, even if those do not fit the model exactly. Apley and Shi (1999) proposed to apply simple ARIMA models to complex processes since 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 in monitoring processes that deviate from the normality assumptions. Nonetheless, as seen in this section, all the above assumptions do not hold if the system is highly nonlinear and the data depart severely from the assumed underlying model.

In this Section we applied known SPC methods for dependent data to the random-walk process of Section 4.1. We focused on the performance of Shewhart and ARIMA based approaches that were suggested for ‘general-purpose’ monitoring. It is seen that all the conventional SPC methods failed to monitor the considered process.

We started by implementing the Special Cause Chart (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-fit ARIMA model (i.e., with lowest MSE) describing the random-walk process. It turned out to be the following AR(2) model: $\hat{x}_t = 1.948 + 0.406x_{t-1} - 0.0376x_{t-2}$.

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-fit 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 these assumptions are violated here. Over 40% of 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 the MA(1) that 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 non-linear 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.

/// insert Figure 12 about here ***///***

Figure 12: The SCC control chart for 'in-control' data

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, while 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 one standard deviation, i.e., where $\mu' = \mu_0 + 1 \cdot \sigma_0$.

Figure 13 and Figure 14 present, respectively, 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 $\hat{\sigma} = \frac{\overline{S}}{c_4} = \frac{0.6148}{0.94} = 0.654$, 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 is caused since neighboring observations in the random-walk tend to generate a small sample variance (high probability for a step size of zero), while 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 (since 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.

///* insert Figure 13 and Figure 14 about here ***///**

Figure 13: Shewhart $\overline{X}$ chart – 'in-control' data (solid line) and 'out-of-control' data (dashed line)
Figure 14: Shewhart *S*-chart – 'in-control' data (solid line) and 'out-of-control' data (dashed line)

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 of *N*=125, which equals to the sample size used by the CSPC to construct the context trees. Figure 15 and Figure 16 present the $\overline{X}$ and *S* charts of both the 'in-control' data (solid line) and the 'out-of-control' data (dashed line) respectively. The estimated parameters of the underlying distribution are: $\hat{\mu} = \overline{\overline{X}} = 3.0129$, and $\hat{\sigma} = \frac{\overline{S}}{c_4} = \frac{1.3765}{0.998} = 1.379$.

As expected, the estimated standard deviation doubled due to the increase in the size of the sample. 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. The explanation to this phenomenon 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.

///* insert Table 3 about here ***///**

Table 3: Change in the process transition probabilities due to a shift of δ=1

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, while 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 maintaining 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 Figure 13 and Figure 14 presenting the Shewhart SPC for a sample size of *N*=5.

///* insert Figure 15 and Figure 16 about here ***///**

Although it has been shown that the Shewhart method can sometimes be implemented on processes that deviate from its underlying assumptions (e.g., see 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. It is shown that the ARIMA-based SPC, that is often applied to complicated non-linear processes, fails in monitoring state-dependent processes. Using the *model-generic* CSPC, however, does not come without a price. The major drawback of CSPC is the relatively large sample size which is 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, 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, since 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.

Acknowledgements

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

CSPC website

For available CSPC web server see http://www.eng.tau.ac.il/~bengal/

References

Alwan L.C., Ebrahimi N., 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** (1), 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., 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**, NOS. 2-4, 85-93.

Ben-Gal I., Singer G., (2001), "Integrating Engineering Process Control and Statistical Process Control via Context Modeling," to appear in *IIE Transactions on Quality and Reliability*.

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 02-12*. Submitted.

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** (1-2), 95-133.

Boardman, T. J., 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", *American Statistician*, **47** (3), 203-206.

Buzacott, J. A., and Yao, D, (1986a), "On queueing network models of flexible manufacturing systems", *Queueing systems*, **1**, 5-27.

Buzacott, J. A., and Yao, D, (1986b), "Flexible manufacturing systems: A review of analytical models", *Management Science*, **32** (7), 890-905.

Cochran, W. G., (1952), "The Chi-square test of goodness of fit," *Ann. Math. Statistic*, **23**, 315-345

Elliott R.J, Lakhdaraggoun, 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., 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 (reprinted in 1978 by MA: Peter Smith).

Lu, C.W., Reynolds, M.R., (1999), "EWMA Control Charts for Monitoring the Mean of Autocorrelated Processes," *Journal of Quality Technology*, **31** (2), 166-188.

Lu, C.W., Reynolds, M.R., (1997), "Control charts for Monitoring processes with Autocorrelated data," *Nonlinear Analysis: Theory, Methods & Applications*, **30** (7), 4059-4067.

May, W.L., Johnson W.D., (1997), "Properties of Simultaneous Confidence Intervals for Multinomial Proportions," *Communications in Statistics-Simulation and Computation*, **26** (2), 495-518.

Montgomery, D.C. and Mastrangelo, C.M. (1991), "Some statistical process control methods for autocorrelated data," *Journal of Quality Technology*, **23** (3), 179-193.

Ohler U. and Niemann H., (2001), “Identification and analysis of eukaryotic promoters: recent computational approaches”, *TRENDS in Genetics*, **17** (2), 56-60.

Rissanen, J., (1983), "A Universal Data Compression System," *IEEE transactions on Information Theory*, **29** (5), 656-664.

Rissanen, J., (1999), "Fast Universal Coding with Context Models," *IEEE transactions on Information Theory*, **45** (4), 1065-1071.

Runger, G.C, Willemain, T.R. and Prabhu, S. (1995), "Average-run-lengths for CUSUM control charts applied to residuals," *Communications in statistics: Theory and Methods*, **24** (1), 273-282.

Runger, G., Willemain, T., (1995), "Model-based and Model-free Control of Autocorrelated Processes," *Journal of Quality Technology*, **27** (4), 283-292.

Runger, G.C and Willemain, T.R. (1996), "Batch-means control charts for autocorrelated data," *IIE Transactions*, **24**, 66-80.

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** (1), 3-17.

Weinberger, M., Rissanen, J., Feder, M., (1995), "A Universal Finite Memory Source," *IEEE Transactions on Information Theory*, **41** (3), 643-652.

Zhang, N.F. (1998), "A Statistical Control Chart for Stationary Process Data," *Technometrics*, **40** (1), 24-38.

Appendix 1: 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		Context-tree construction algorithm, suggested in Rissanen (1983) and elaborated in Weinberger, et al. (1995).
Context-based SPC	CSPC	SPC method for state-dependent data based on context tree modeling.
Context-tree		A *context-tree* is 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 above is the Finite State Machine (FSM), which is 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 is denoted by $\hat{P}_0(x,s)$.
Monitored context-tree	$\hat{P}_i(x,s)$	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	$x \in X$	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 2: The Context Algorithm

The context-tree construction algorithm, named 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 in Rissanen (1983) and modified in Weinberger *et al.* (1995) and Ben-Gal et al. (2001). The differences are mainly in stage 3 and stage 4 of the algorithm.

The input of the algorithm is a sequence of observations $x^N = x_1,\ldots,x_N$, with elements $x_t \in \{0,1,2,3,4\}$, $t=1,..,N$ 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 follows.

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

Step I.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 one until reaching the tree's current deepest node, say $x_t, ..., x_{t-l+1}$. If necessary, use an initial string preceding x^t in order to account for initial conditions.

Step I.2. If the last updated count is at least 1, and $l < m$, where *m* is the maximum depth, create new nodes corresponding to x_{t-r}, $l < r \leq m$, and initialize all its symbol counts to zero 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. *r* is the depth of the new deepest node, reached for the current path, after completing step I.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 is composed of a symbol set of *d*=5, and its length is *N*=6. Table A1 presents the tree growing and counter update process. Note that *s* is the reverse string, $s = x_t, x_{t-1}, ...$

Figure A1 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.

///* insert Table A1 and Figure A1 about here ***///**

II. 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 (2.3). 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, i.e., $|w| \leq \log(t+1)/\log(d)$; and, *Pruning rule* 2: the information obtained from the descendant nodes, $sb \ \forall b \in X$, compared to 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 counter values similar to that of the parent node. In particular, calculate $\Delta_N(sb)$, the (ideal) *code-length-difference* of the descendant node *sb*, $\forall b \in X$,

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

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 A2 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 A1.

///* insert table A2 about here ***///**

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 A2 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 A1.

///* insert figure A2 about here ***///**

One can notice that the pruned tree of Figure A2 is of depth one. Recall that the restricted random walk process of Section 4 is a Markov chain of order one (see the state transition diagram in Figure 5). Therefore, the pruning stage of context algorithm identified the process-data dependence.

III. Stage 3: Selection of optimal contexts

In this stage, the set of *optimal contexts*, Γ, containing the *S* shortest contexts satisfying (2.3) 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 which is not a leaf, however, for certain symbol(s) its path defines an optimal context satisfying (2.3) (for other symbols, eq. (2.3) is not satisfied and a descendant node(s) has to be created). The set of optimal contexts is specified by applying the following rule:

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

which means that Γ contains only those contexts that are not part of longer contexts. When the inequality in (A.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 therefore does not satisfy (A.2). Such inconsistency can be solved by introducing an initiating symbol string as suggested in Weinberger *et al.* (1995). In summary, Γ contains all the leaves in the tree and the partial leaves satisfying (A.2) for certain symbols.

Running example: Applying (A.2) to the pruned counter context-tree presented in Figure A2 results with five optimal contexts $\Gamma = \{0,1,2,3,4\}$. All the contexts, in this case, are leaves. Note that the root node is not an optimal context since it is fully contained in its descendant nodes.

IV. Stage 4: Estimation of the context-tree probability parameters

This stage is composed of three steps: 1) the probabilities of optimal contexts are estimated and denoted by $\hat{P}(s)$, $s \in \Gamma$; 2) the conditional probabilities of symbols given the optimal contexts are estimated and denoted by $\hat{P}(x \mid s)$, $x \in X$, $s \in \Gamma$; 3) the estimated joint probabilities of symbols and optimal contexts are calculated $\hat{P}(x, s)$, $x \in X$, $s \in \Gamma$.

Step IV.1: Given the set of optimal contexts and the pruned counter tree, the probability of optimal contexts in the tree, $\hat{P}(s)$, $s \in \Gamma$, are estimated by their frequency in the string:

$$\hat{P}(s) = \frac{n(s)}{\sum_{s \in \Gamma} n(s)} = \frac{\sum_{x \in X} \left(n(x \mid s) - \sum_{b \in X} n(x \mid sb) \right)}{\sum_{s \in \Gamma} \sum_{x \in X} \left(n(x \mid s) - \sum_{b \in X} n(x \mid sb) \right)} \qquad \forall x \in X,\ s \in \Gamma \qquad \text{(A.3)}$$

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 symbols 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 distincive optimal contexts is approximated by the multinomial distribution.

Running example: The estimated probabilities of optimal contexts in Figure A2 are, respectively:

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

Step IV.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),

$$\hat{P}(x|s)=\frac{n(x|s)-\sum\limits_{b\in X} n(x\mid sb)+{}^{1}\!/_{v}}{n(s)+{}^{d}\!/_{v}} \qquad \forall x,b\in X,\ s\in\Gamma\,, \qquad \text{(A.4)}$$

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

An alternative approach is a *non-predictive approach*, where $v\to\infty$ and $\frac{0}{0}\equiv 0$. The choice 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 non-predictive case, the number of degrees of freedom is adapted according to the number of categories that are not equal to zero (see, e.g., May and Johnson, 1997) since the multinomial theory stands for non-zero probability categories.

Running example: Figure A3 presents the estimated conditional probabilities of symbols given contexts. These estimates are generated by applying the non-predictive approach to the *counter context-tree* presented in Figure A2. For example, the conditional probability of a symbol type $x\in\{0,1,2,3,4\}$ given the context $s=0$, is estimated as $\hat{P}(x|0)=\left(\frac{14}{22},\frac{3}{22},0,0,\frac{5}{22}\right)$ The probabilities of symbols in the root are presented also for general information.

Step IV.3: The joint probabilities of symbols and optimal contexts that represent the context-tree in its final form are evaluated $\hat{P}(x,s)=\hat{P}(x|s)\cdot\hat{P}(s),\ x\in X,\ s\in\Gamma.$

///* insert figure A3 about here ***///**

General note: It is possible to consider non-sequential contexts. Rissanen (1983) proposed to use a *permutation function* to map the correct dependency order.

List of Figure and Table captions in article

Figures in article:

Tables in article:

List of Figure and Table captions in the appendices

Figures in Appendix:

Tables in Appendix:

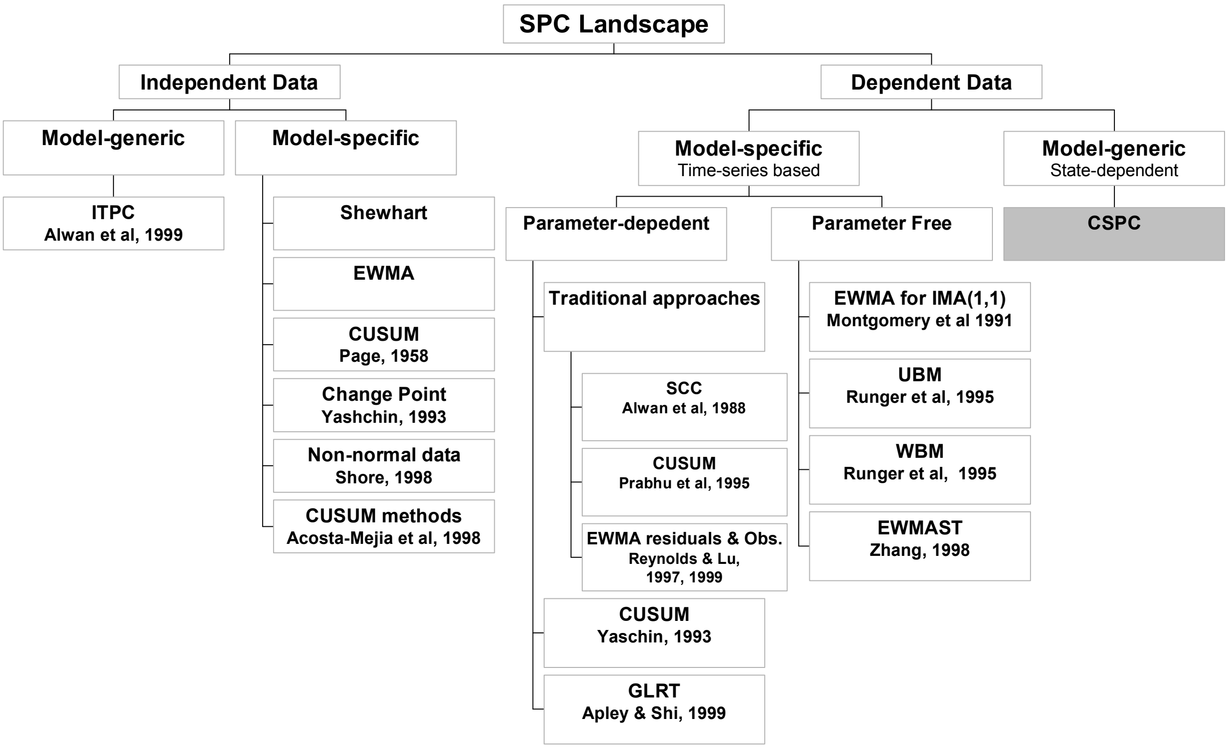


Figure 1: Taxonomy of SPC methods

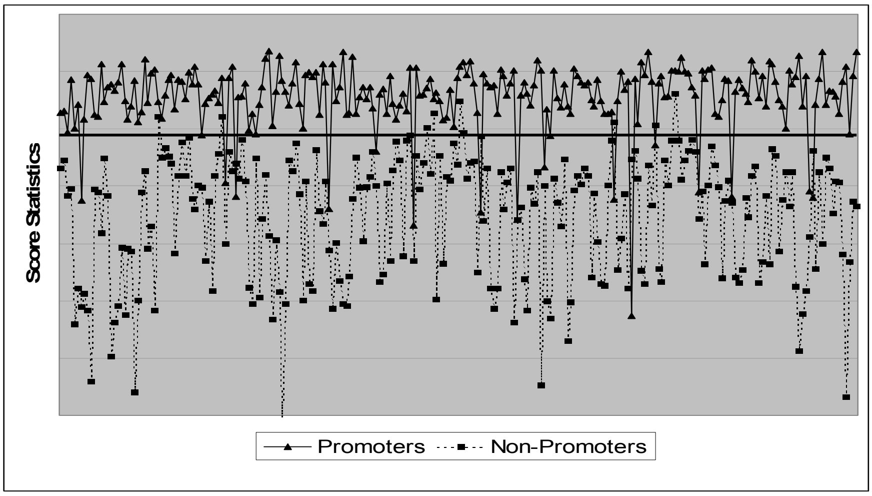


Figure 2. Score statistics of two E. Coli sequences of DNA-spaced reading-frames. The upper series represent promoter sequences and the lower series represent non-promoter sequences. Values of the score statistics are computed by the context-tree model.

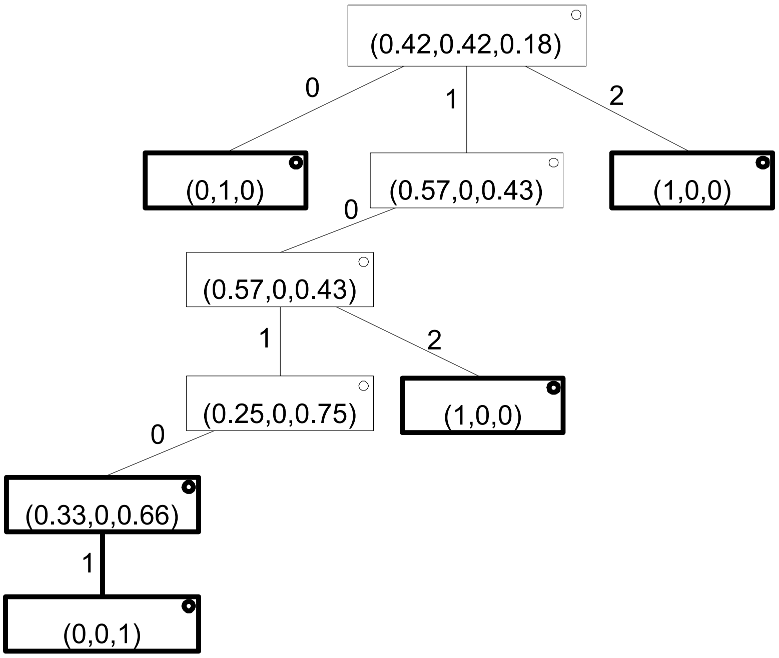


Figure 3: Illustration of a context-tree with *S*=5 optimal contexts (bolded frame)

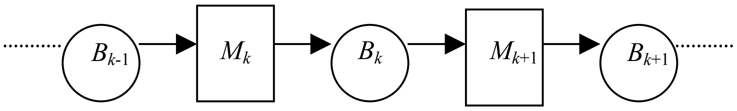


Figure 4: A Subsystem of a production line of *K* machines

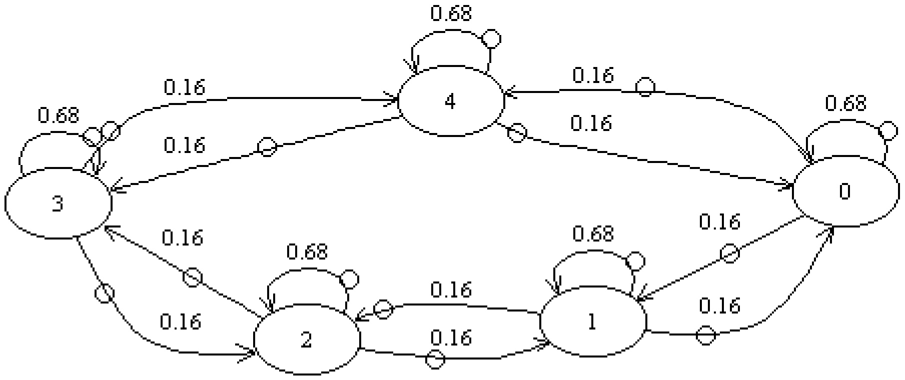


Figure 5: State transition diagram for the process.

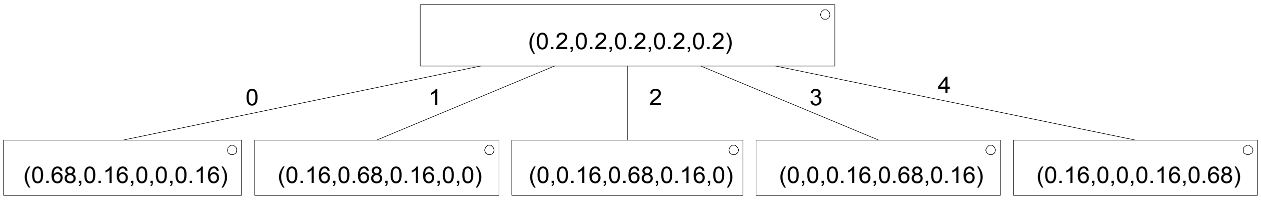


Figure 6: The analytically derived singled-level context-tree.

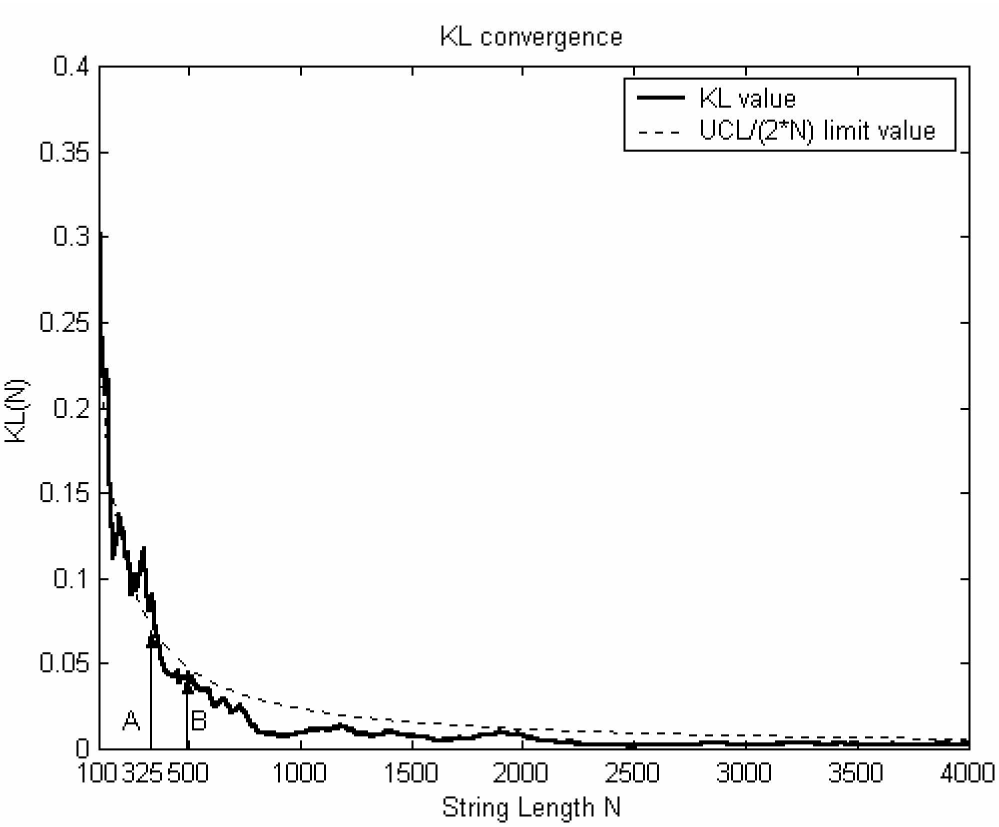


Figure 7: The KL value between the analytical tree and the estimated tree as a function of the input string length *N*.

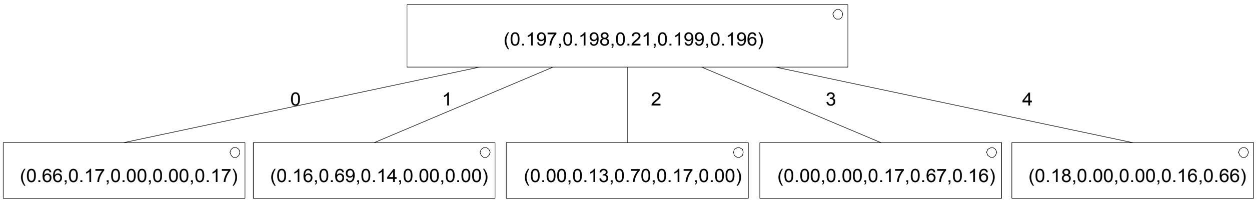


Figure 8: Estimated reference context-tree as a result of the implementation of context algorithm to *N*=1000 observations.

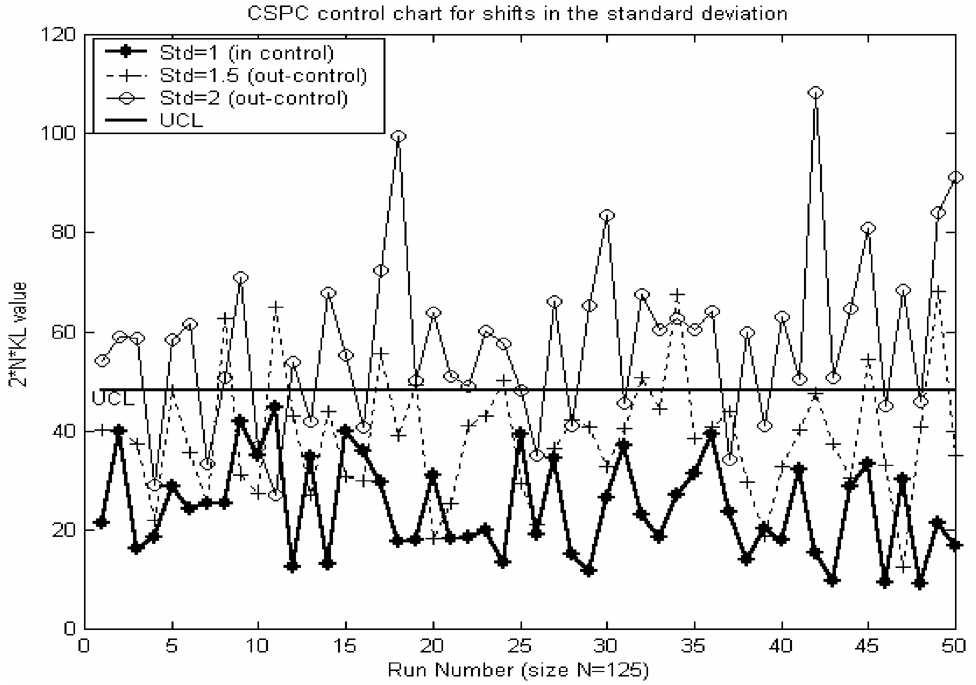


Figure 9: Shifts in the process underlying normal standard deviation, λ=1, 1.5, 2 (a portion of 50 runs is presented).

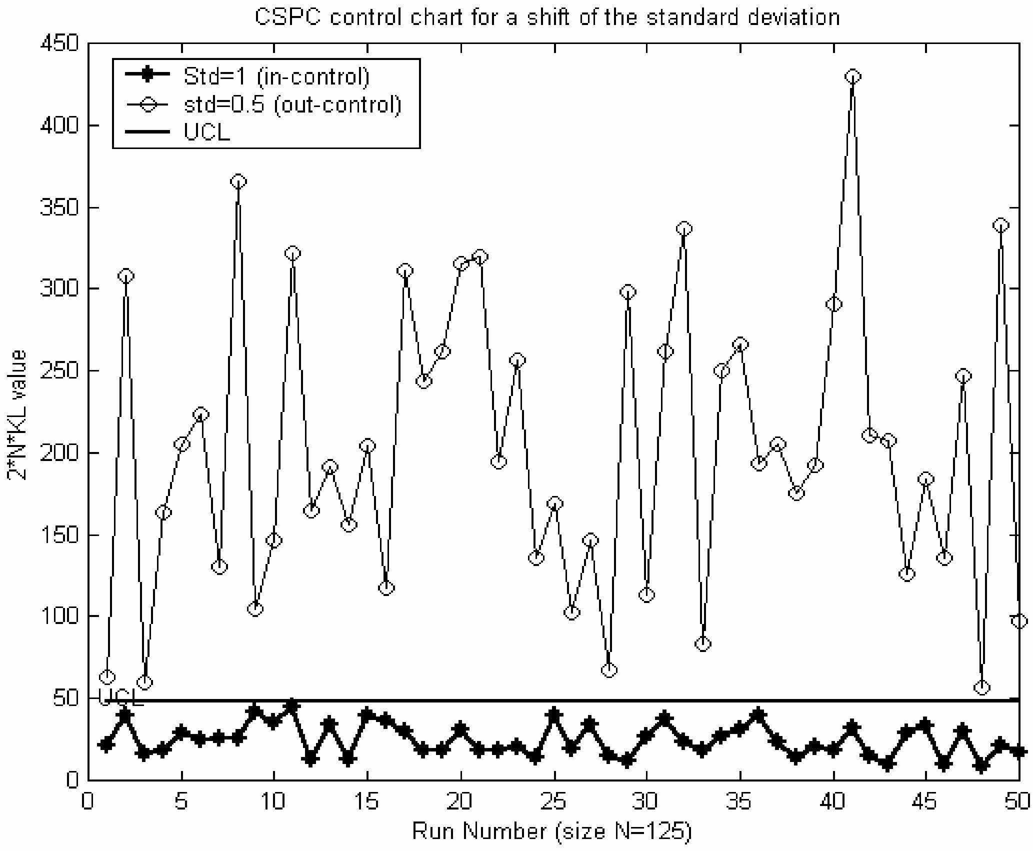


Figure 10: Shift in the process underlying normal standard deviation, λ=0.5, 1
(a portion of 50 runs is presented)

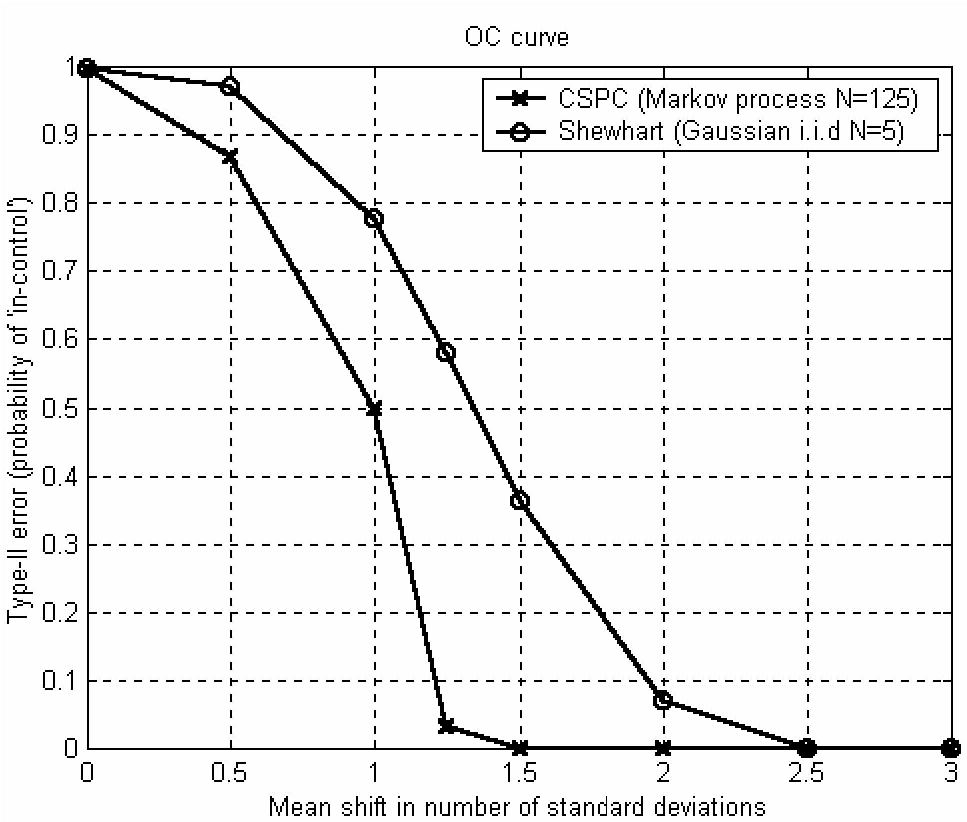


Figure 11: The Operating Characteristics curve for mean shifts

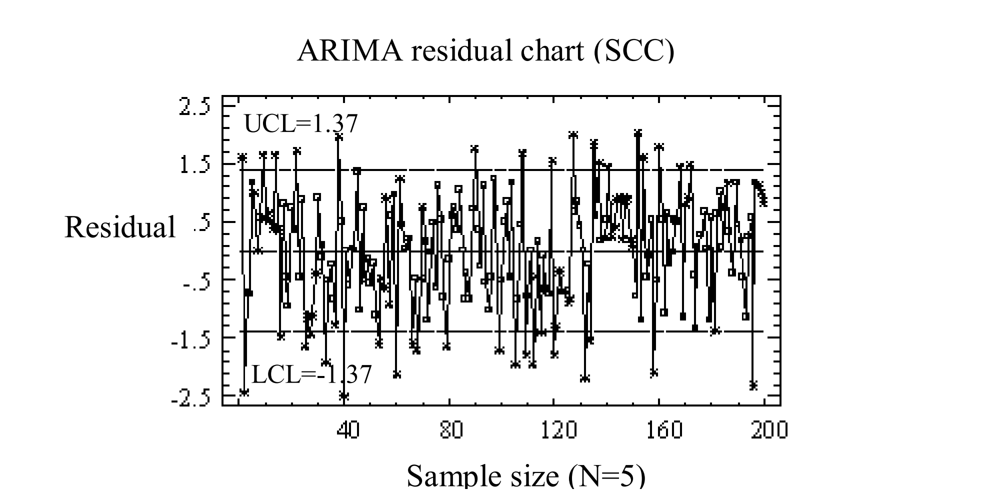


Figure 12: The SCC control chart for 'in-control' data

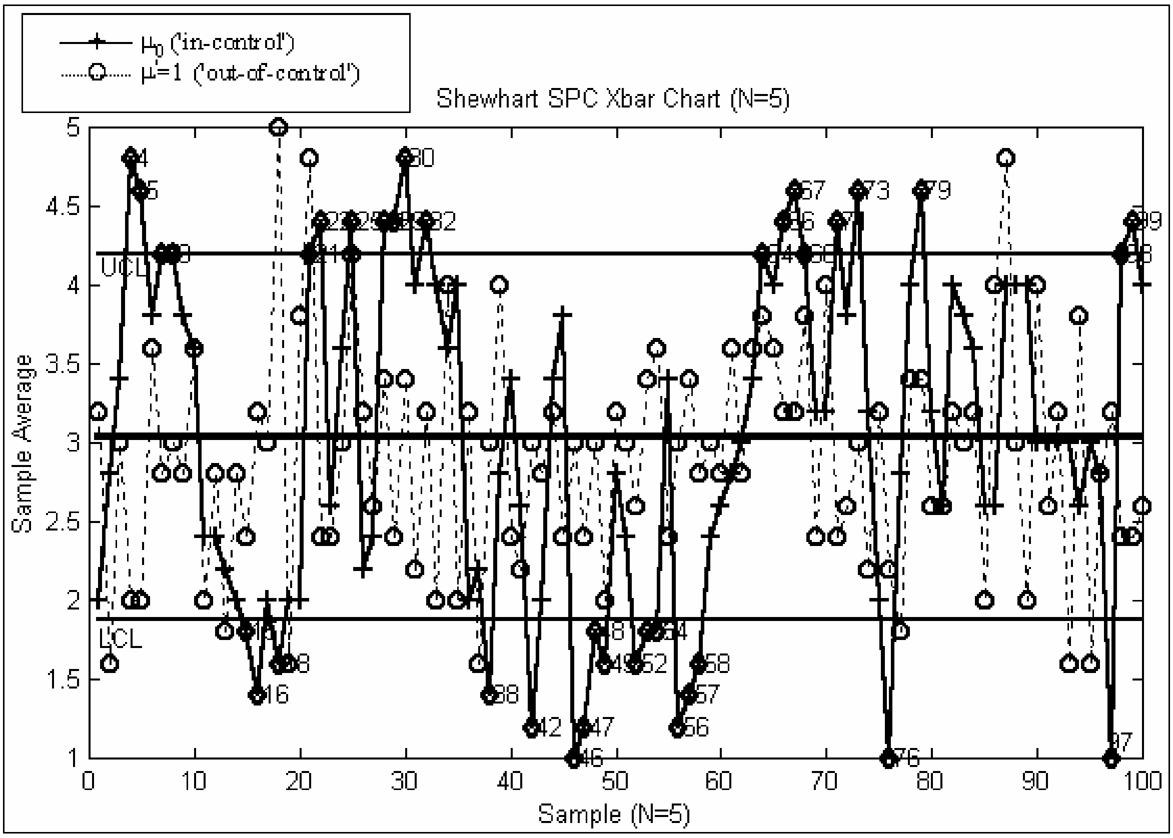


Figure 13: Shewhart $\overline{X}$ chart - 'in-control' data (solid line) and 'out-of-control' data (dashed line)

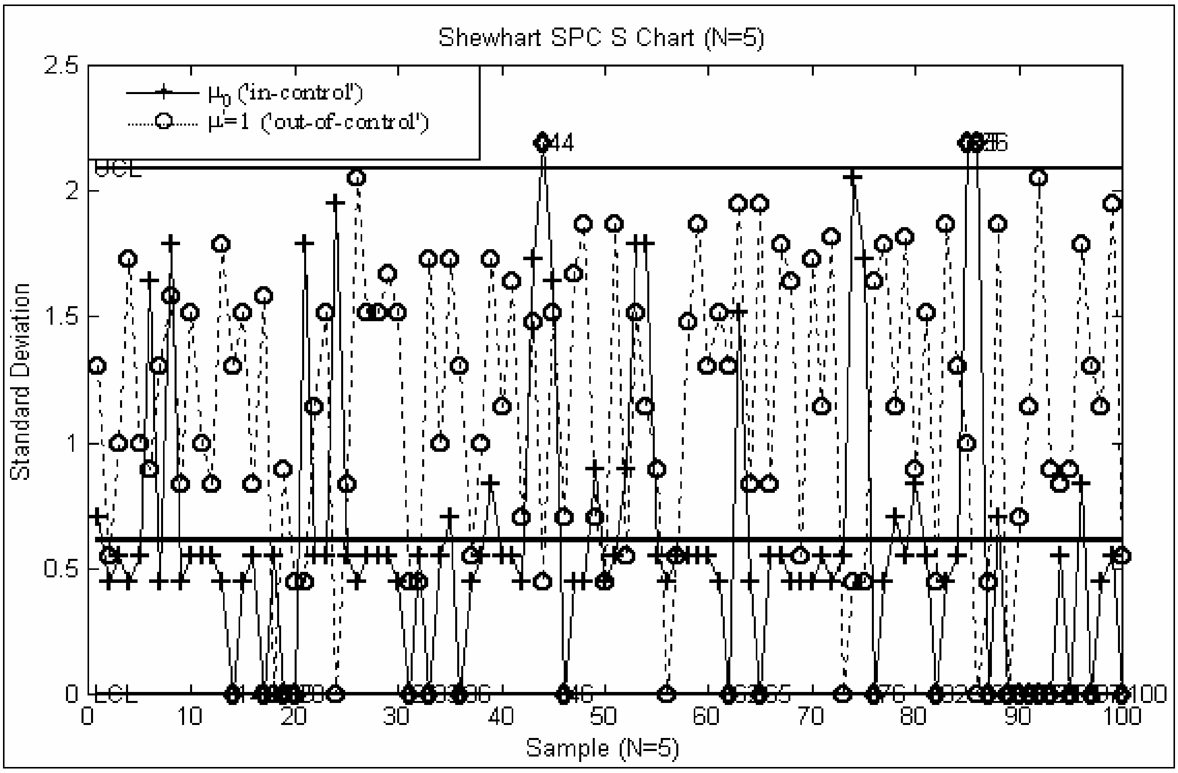


Figure 14: Shewhart *S*-chart - 'in-control' data (solid line) and 'out-of-control' data (dashed line)

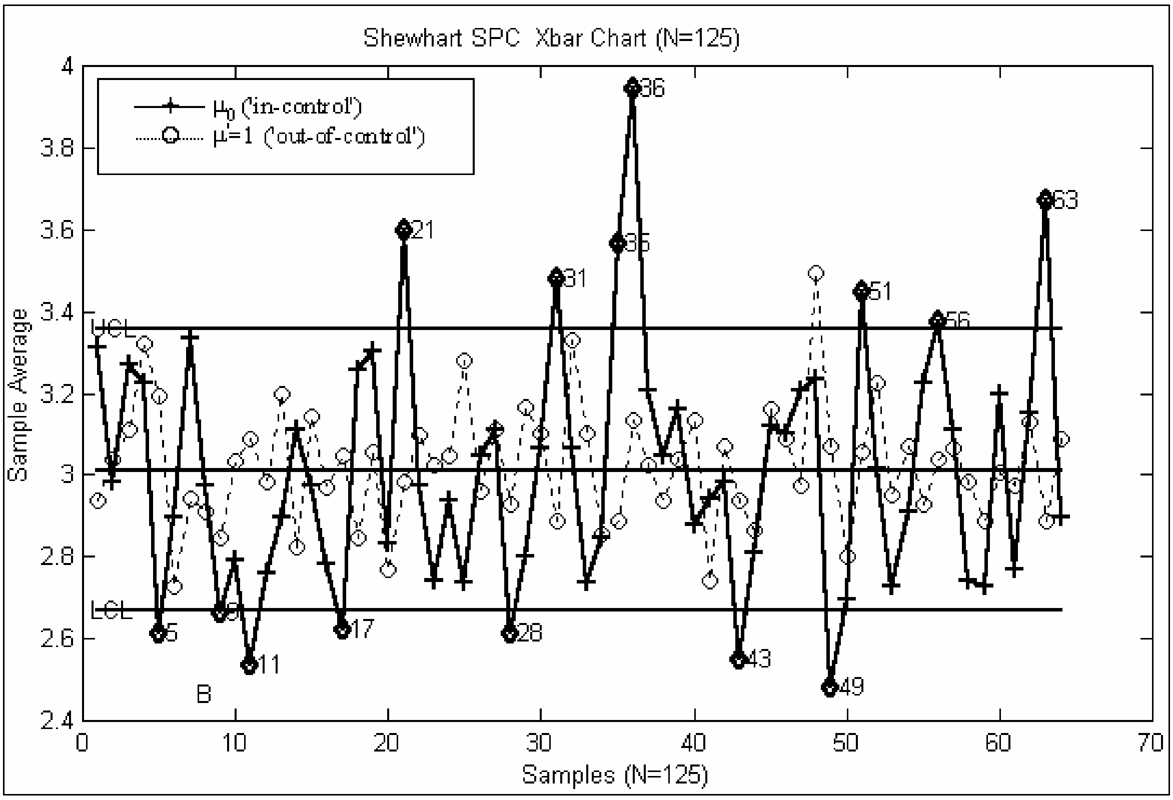


Figure 15: Shewhart $\overline{X}$ chart for *N*=125 sample size 'in-control' data (solid line) and out of control data (dashed line)

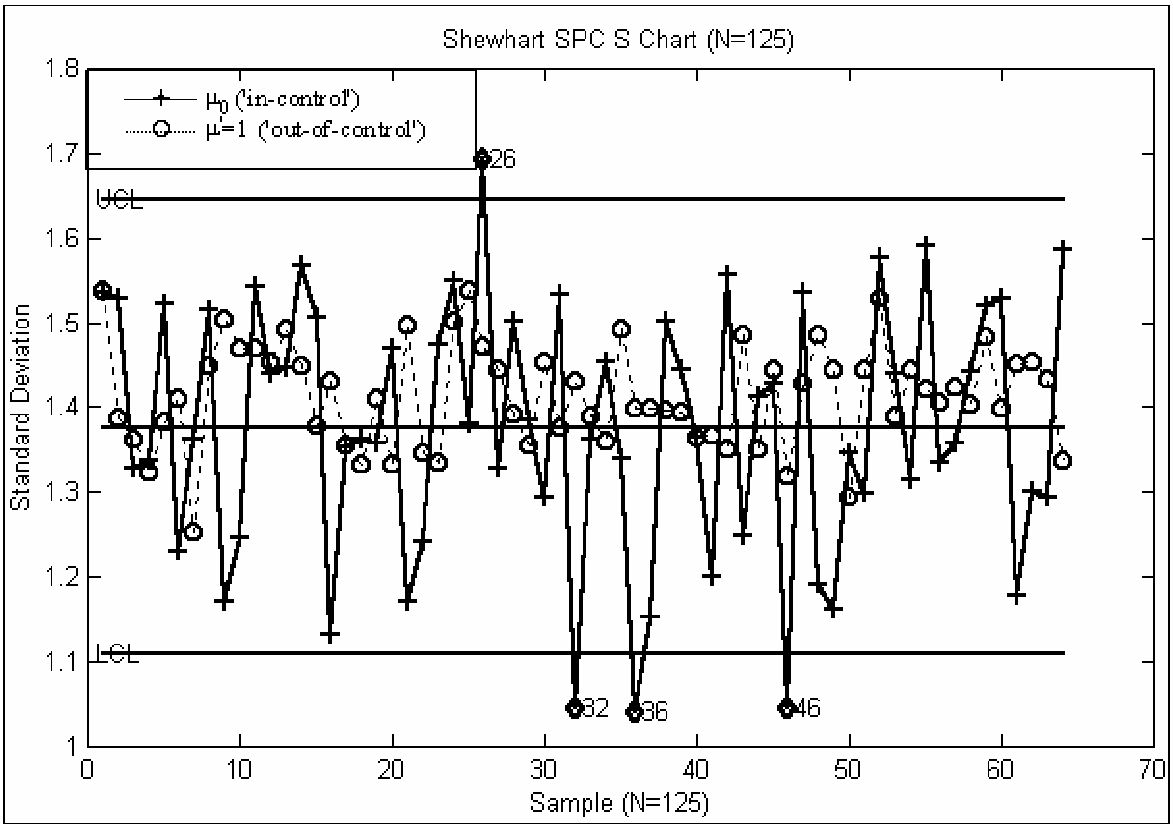


Figure 16: Shewhart *S* chart for *N*=125 sample 'in-control' data (solid line) and out of control data (dashed line)

	x_1	x_2	x_3	x_4	x_5
Step 1: i.i.d normal values string	-0.4326	-1.6656	0.1253	0.2877	-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

Table 1: Feedback-controlled process generation example

Std Shift	**Probability of the random-walk steps**			**CSPC performance (ARL and Type II error)**		**ARL of S-chart (benchmark)**	
λ	**+1**	**0**	**-1**	**Test power 1–β (No. of 'out-of-control' runs)**	**Estimated ARL**	**ARL *N*=5**	**ARL *N*=125**
1	0.16	0.68	0.16	0% (0)	∞	257.13	369.08
1.5	0.25	0.5	0.25	20% (10)	5	6.96	1
2	0.31	0.38	0.31	74% (37)	1.35	2.35	1
0.5	0.02	0.976	0.02	100% (50)	1	∞	1

Table 2: Performance of CSPC during scenario No. 1 with respect to the ARL and Type I error. The last two columns present the ARL of a traditional *S*-chart for sample sizes *N*=5 and *N*=125.

Mean shift	**Transition Probabilities of the random-walk steps**		
δ	**+1**	**0**	**-1**
0 ('in-control')	0.16	0.68	0.16
1 ('out-of-control')	0.5	0.49725	0.00275

Table 3: Change in the process transition probabilities due to a shift of δ=1

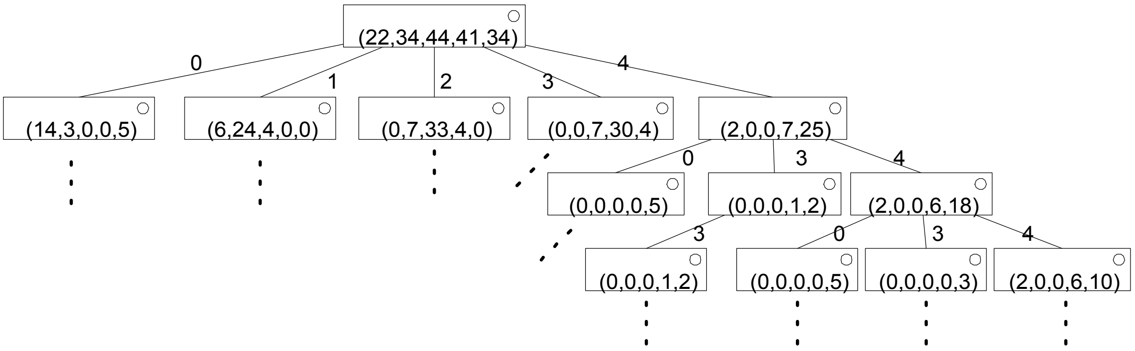


Figure A1: A portion of the counter context-tree for the restricted random walk process for an input string of *N*=175 observations (after stage 1)

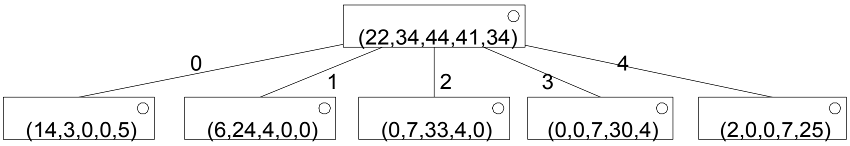


Figure A2: The pruned counter context-tree of the restricted random walk process for an input string of N=175 observations (after stage 2)

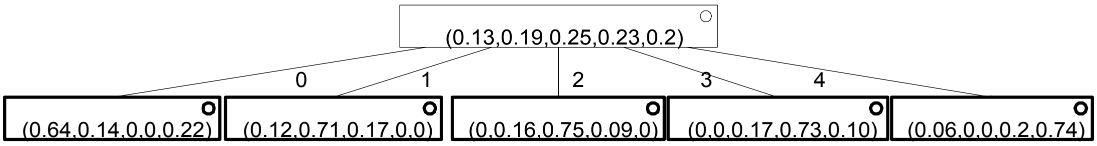


Figure A3: The context tree containing vectors of conditional probabilities $P(x|s)$ as obtained from the counter context-tree in figure A2. Optimal contexts are represented by the bolded frame.

[Table A1- see next page]

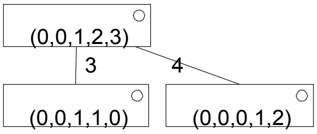
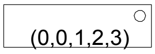
Rule	Tree	Description
Rule 1:		Rule 1: Maximum tree Depth ≤ log(t)/log(d) = log(6)/log(5)=1.11 The maximum tree depth is of level one. Thus, all nodes of level 2 and below are trimmed.
Rule 2:		Rule 2: for the rest of the nodes in level one and the root, we apply trimming rule 2. The threshold for C=2 is: $\Delta_6(u) > 2(d+1)\log(t+1) = 33.7$ And for each of the nodes: $\Delta_6(sb=\lambda 3) = 0+0+1\cdot\log\left(\frac{0.5}{1/6}\right)+1\cdot\log\left(\frac{0.5}{2/6}\right)+0 = 2.17$ $\Delta_6(sb=\lambda 4) = 0+0+0+1\cdot\log\left(\frac{1/3}{2/6}\right)+2\cdot\log\left(\frac{2/3}{3/6}\right) = 0.83$ The code-length difference is below the threshold, hence the first level nodes are trimmed.

Table A2: Pruning stage in context algorithm for the string x^6=4,4,4,3,3,2

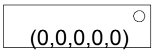
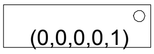
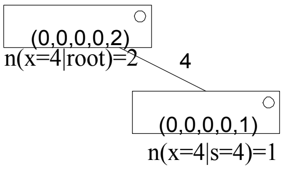
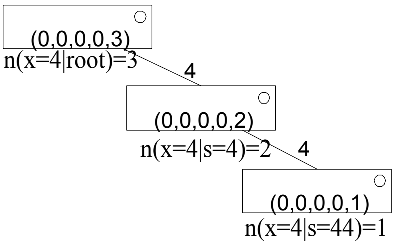
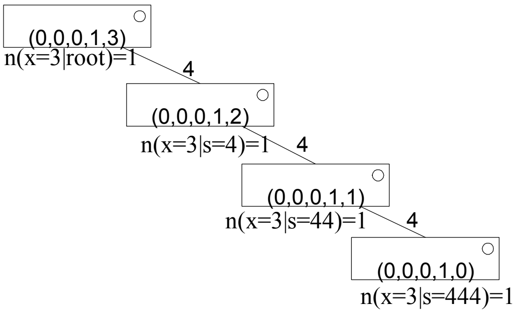
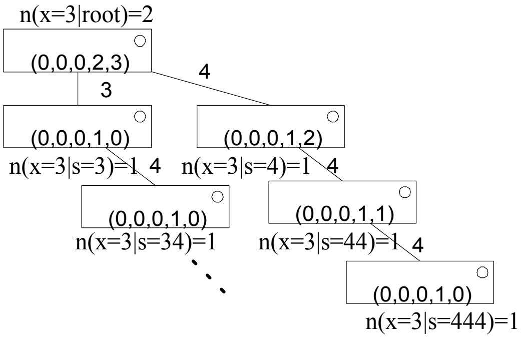
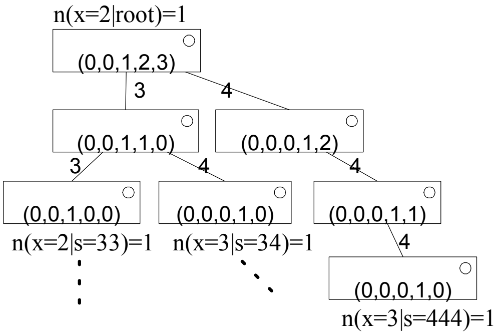
Steps	Tree	Description
Step 0: T_0		Initialization: the root node, λ, denotes the empty context
Step 1: T_1 $x^1 = 4$		The only context for the first symbol is λ, the counter $n(x=4\|\lambda)$ was incremented by one.
Step 2: T_2 $x^2 = 4,4$	 (0,0,0,0,2) n(x=4|root)=2 4 (0,0,0,0,1) n(x=4|s=4)=1	The counters $n(x=4\|\lambda)$ and $n(x=4\|s=4)$ are incremented by one. The node of the context s=4 is added to accumulate the counts of symbols given the context s=4.
Step 3: T_3 $x^3 = 4,4,4$	 (0,0,0,0,3) n(x=4|root)=3 4 (0,0,0,0,2) n(x=4|s=4)=2 4 (0,0,0,0,1) n(x=4|s=44)=1	The counter of symbol 4 is incremented by one 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 $x^4 = 4,4,4,3$	 (0,0,0,1,3) n(x=3|root)=1 4 (0,0,0,1,2) n(x=3|s=4)=1 4 (0,0,0,1,1) n(x=3|s=44)=1 4 (0,0,0,1,0) n(x=3|s=444)=1	The counters - $n(x=3\|\lambda)$, $n(x=3\|s=4)$ and $n(x=3\|s=44)$ are incremented by one since the past contexts are $s=\lambda$, s=4, s=44. A new node is added for the context s=444 of the observation $x_4 = 3$.
Stage 5: $x^5 = ...,4,3,3$	 n(x=3|root)=2 (0,0,0,2,3) 3 4 (0,0,0,1,0) (0,0,0,1,2) n(x=3|s=3)=1 4 n(x=3|s=4)=1 4 (0,0,0,1,0) (0,0,0,1,1) n(x=3|s=34)=1 n(x=3|s=44)=1 4 (0,0,0,1,0) n(x=3|s=444)=1	Add new nodes for the contexts s=3, s=34, 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: $x^6 = ...,3,3,2$	 n(x=2|root)=1 (0,0,1,2,3) 3 4 (0,0,1,1,0) (0,0,0,1,2) 3 4 4 (0,0,1,0,0) (0,0,0,1,0) (0,0,0,1,1) n(x=2|s=33)=1 n(x=3|s=34)=1 4 (0,0,0,1,0) n(x=3|s=444)=1	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 A1: Tree growing and counter updating stage in context algorithm for string x^6=4,4,4,3,3,2