An Anomaly Event Correlation Engine: Identifying Root Causes, Bottlenecks, and Black Swans in IT Environements

Mazda A. Marvasti, Arnak V. Poghosyan, Ashot N. Harutyunyan, and Naira M. Grigoryan
Management BU
VMware Inc.
{mazda;apoghosyan;aharutyunyan;ngrigoryan}@vmware.com

***Abstract*—A data-agnostic approach to automated problem (fault or change) root cause determination in complex IT systems based on monitoring and anomaly event data correlation is developed. It also includes a framework for identification of bottlenecks and black swan type issues in IT infrastructures. The relevant anomaly event correlation engine (AECE) is designed, prototyped, and successfully applied to data from real environments. This approach is based on a historical analysis of abnormality events (obtained by application of an IT management analytics) with their probabilistic correlations. The general theory that applies information measures between the random variables embodying those events is described and a particular scenario is discussed. The method is capable in detecting origins of problems and generating real-time recommendations for their locations in a hierarchical system. Moreover, it allows localizing the bottleneck IT resources which are persisting causes of historical problems. The black swan type events are shown to be retrospectively identified and used in their risk estimation on-line. The technology and algorithms are based on statistical processing of virtual directed graphs produced from historical anomaly events.**

Keywords-network monitoring; fault management; anomaly detection; anomaly event; event correlation; root cause analysis; bottlenecks, black swans, pattern detection; probability; directed graph.

I. INTRODUCTION AND RELATED WORK

Modern economies and business services are running on complex, dynamic, and heterogeneous Information Technology (IT) infrastructures. This is the reason that the past generation rule-based and static diagnostic tools and methods that are designed to maintain the "health" of those systems, and hence protect the underlying business processes from impacts of their environments, become outdated and useless. Furthermore, managing today's IT systems becomes an increasingly complicated problem if we take into account the deployment of rapidly growing cloud computing services and virtualized environments.

Automated anomaly detection and related problem root cause (RC) localization in IT-based business processes is an extremely important technological challenge. In such dynamic environments this results in usage of statistical inference models. Bayesian networks are widely used (see [1]-[3] and references therein) for such and many different inference applications, however, with exponential complexity in network's tree-width. Interactive models such as in [4] for incremental improvement of diagnosis do not allow complete automation of the management solutions and self-healing capabilities that adaptively learn and update system knowledge and react to environmental changes adequately.

The goal of the current IT management area is development of models, methods, and apparatus for estimation of dynamic performance normalcy of systems, their abnormality detection and control (which include misbehaving RC determination), bottleneck identification, change point detection, and other analytical capabilities.

The first step towards adaptive and universal solutions to the above mentioned problems provides measuring the business and its IT performance metrics (such as on-line users count and application response time) and thus creating the "data image" of the system containing of devices, applications, and operating systems. Nowadays the quantitative monitoring of IT networks and data collection for the whole system in terms of various metrics (time series) is a common practice. The next stage and the main technological and scientific challenge is knowledge mining from reservoirs of measured data and its proactive application in fault management of systems. New mathematical/statistical methods and tools are needed to turn this data storage into real value for performance management of those systems. The latter includes a retrospective analysis of collected time series data and on-line decision making schemes based on that analysis. In this context, the identification of network faults and slow-downs, as well as their precursor's detection is the goal of those statistical tools.

Determination of a potential RC when IT systems are misbehaving is the primary solution our method and the corresponding engine provide. It can also identify potential critical abnormality patterns in IT infrastructures. The study relies on statistical modeling of abnormality events life-times and their correlation in those infrastructures. At the same time, it does not assume a contextual knowledge of systems or applications running in networks. In comparison with existing

approaches we adopt a totally data-driven approach that allows handling a large volume of fault events and produce recommendations for highly probable root events and associated IT resources responsible for the out-of-normal situations. Other algorithms in this field (such as [4]) rely on heavy utilization of rules and topology knowledge of the applications or systems to enable effective recommendations for root problems identification and localization.

The methodology employed here relies on information measures applied on the system abnormality events space to create historical recommendation lists of components (resources) for identification/localization of RC's of mal-functions in the past. The current active events are then mapped into those historical lists to create active recommendations. What this algorithm produces is a set of RC lists rank ordered based on a computed "likelihood index" which identifies the most probable RC event in the generated list and the whole events space. It is assumed that we have no knowledge of casual and other relationships in the environment. However, if the system topology information is available, it can be used to achieve a more contextual recommendation.

Here we treat also two other problems: system bottleneck identification and black swan type events risk estimation.

Bottleneck identification is critical for improving the resources which are hindering the functioning of networks in long term perspectives and are hidden in the whole anomaly history of those systems. Hence their localization and redesign in large infrastructures prevent further causes of serious impacts on the system performance or lower their probability.

The second problem we consider is on-line estimation of risk of black swan type events [5] which are extremely rare in occurrence, however very high in potential to destroy the system or bring a significant influence on its functioning. This can be done also based on the analysis of past behavior of the system being monitored and its projection onto present with highly probable scenarios.

As mentioned, for our analysis we use the anomaly events historically produced on IT resources under monitoring and abnormality control. An example of such an analytics and software engine is the VMware vCenter Operations Manager that constructs/recognizes the normal behavioral patterns of various IT resources and alarms on "out-of-normal" situations in the system via dynamic process control or thresholding. This is a technology that implements sophisticated statistical analysis for predicting the normalcy ranges of monitored data based on its category, variability, change point occurrence, and cyclicality. The reference [6] is an example of such a dynamic thresholding algorithm. Figure 1 illustrates a metric within predicted dynamic thresholds (DT) and illustrates the misbehaving areas subject to alarming as anomalies.

The correlation method adopted here is parametric, statistically counting for life times of DT-based anomaly events. Similar event correlation method that is non-parametric and also applicable beyond IT sphere is studied in [7] in the context of pattern detection in semi-structured data such as log files.

Event correlation techniques have been used over past two and half decades as a principle tool for integrated management in networks in terms of detection causes of performance problems. The paper by Martin-Flatin et al [8] surveys the evolution of those techniques and challenges need to be addressed in design of systems and algorithms for modern, ever complex and tightly integrated networks. An early event correlation model, strictly deterministic, aimed at improving telecommunications networks surveillance and fault diagnosis is introduced by Jakobson and Weissman [9] (see also references therein). Its conceptual framework relies on, in particular, network topology. The systems designed within this kind of frameworks substantially restrict their applicability in fuzzy environments with growing capacities and depth requiring scalability and universality. On the other hand, a uniform framework employing casual correlation codebook formalism (problem is an encoded alarm set with decoding to identify the original problem) for network management is proposed by Kliger et al [10]. The codebook approach does not quickly adapt to changes in network configuration. An extended, single, rigorous, and conceptual framework for casual and temporal correlations was introduced by Hasan et al in [11].

In the context of the related work discussion above, the current research belongs to the temporal correlation framework. At the same time, we consider a general and universal abstraction of anomaly events that are not network-inherent with a defined collection of fault indications, but a statistically inferred dynamic normalcy range violations of IT resources indicating change against the system historically "representative" behavior. Those dynamic ranges are obtained by self-tunable learning from monitoring data. In other words, in this way we introduce the highest possible degree of generality (fuzziness) in alarm correlation theory independently of any physical context, system topology, or expert knowledge inserted in causality relationships. Moreover, in our interpretation, a *problem* (*fault or change*) in an IT system is a bundle of alarms or statistical anomaly events that need to be depressed (or warned, learned, and estimated its impact if it is a change not necessarily yielding a malfunction) in an efficient manner, proactively preventing potential regression of a system state into a serious performance issue. It means that the shortest way to achieve the run-time problem elimination is inferring the root cause alarms with their high-to-low "ratings" to follow in system stabilization/recovery. These definitions encompass the models studied in prior related works. Our framework implies also that any infrastructural transformation, evolution, or variations in the background physical flows within the system, dynamically (statistically) updates the alarm correlations

As soon as the DT's are historically constructed (recognized) for an IT process under monitoring, they can be projected into the future as prediction for time-based normalcy ranges. Any data point appearing above or below those thresholds (or, alternatively, a series of points consecutively staying at out-of-ranges) is an abnormality *event* that can be characterized by an ID triple: {Resource ID, Attribute ID,

Symptom ID}. Under Resource ID we interpret any IT resource such as server, computer, etc. The Attribute ID identifies a specific process (metric, time-series) monitored for that resource (such as CPU, memory, active user count, etc.). The Symptom ID specifies the abnormality character detected at that metric (such as DT above or below).

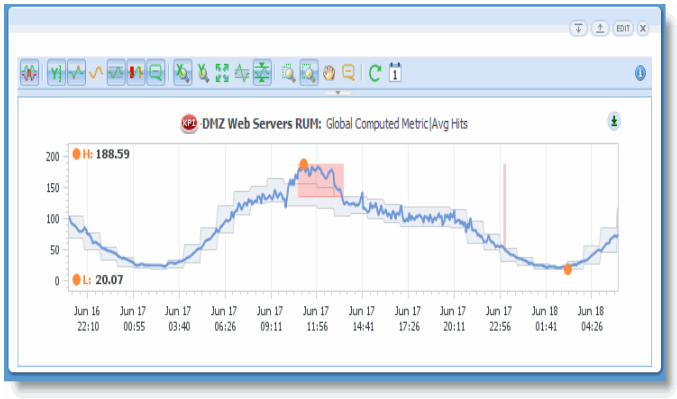


Figure 1. Anomaly detection via dynamic ranges.

The basic layer for this study and our general Anomaly Event Correlation Engine (AECE) is the construction of IT event correlation model (Section 2) which we employ to recognize the historical RC's of past abnormality situations in IT systems. (This is a methodology with less complexity compared with the approaches adopted in the works [1]-[3]). It allows introducing a graph representation for those events with probabilistic interpretations (topic of Section 3). Section 4 introduces the principles of statistical elaboration of the graph and demonstrates the algorithm behind our RC analyzer. It results in bottleneck identification and black swan risk estimation capabilities in Sections 5-6. Notes on experimental results are made in Section 7 and the paper's conclusion is included in Section 8.

II. Modeling Anomaly Events

We analyze the historically detected abnormality events accumulated (for a training period of several weeks) in result of execution of the DT management technique explained in Section 1 and try to observe correlations between their pairs within time proximity. Those correlations we present in form of a probabilistic directed graph. The nodes of the graph stand for the events and the directed connections between two nodes represent the conditional probabilities of event pairs. In general the conditional probability of an event pair (E_j, E_i) can be computed by dividing the joint probability of the two events by the prior probability of event E_i:

$$P(E_j|E_i, a, \Delta t) = \frac{P(E_i, E_j|\alpha, \Delta t)}{P(E_i|\alpha)}$$

where Δt is the span of time where events i and j are considered to be coincident; α represents a function of the event life-times; $P(E_i, E_j|\alpha, \Delta t)$ is the joint probability of events i and j; and $P(E_i|\alpha)$ is the prior probability of event i.

For simplicity we will omit the parameters Δt and α in the notations of probabilities. For joint probabilities we'll use also the notation

$$P_{ij} \triangleq P(E_i, E_j|\alpha, \Delta t).$$

The function α can be computed by an appropriate modeling of event life-times. This is accomplished by representing a typical event life-time as a log-normal distribution:

$$f(t) = \frac{1}{t\sigma'\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{\ln(t)-\mu'}{\sigma'}\right)^2}$$

$$\mu' \equiv \frac{1}{M}\sum_{i=1}^{M} \ln(t_i)$$

$$\sigma' \equiv \sqrt{\frac{1}{M-1}\sum_{i=1}^{M} (\ln(t_i) - \mu')^2}$$

where t is the time from the start to the end of the event, σ' is the standard definition of the log-normal distribution.

Using a slightly different convention,

$$t' \equiv \ln(t)$$

$$k' \equiv \frac{t' - \mu'}{\sigma'}$$

$$f(t') = \frac{1}{\sigma'\sqrt{2\pi}} e^{-\frac{1}{2}(k')^2}$$

the variable α can be defined as:

$$\alpha \equiv \int_{-\infty}^{t'_\alpha} f(t')dt'$$

which can be interpreted as the probability of the logarithm of time to be less than t'_α. Rewriting the integral we get:

$$\alpha = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{k'_\alpha} e^{-\frac{(k')^2}{2}} dk'.$$

And t_α can be obtained as:

$$t_\alpha = e^{(\sigma' k'_\alpha + \mu')}.$$

The effectiveness of the assumption of log-normal behavior for the event life-time distribution is shown in Figure 2. The two graphs in Figure 2 show the cumulative distribution of the actual alert life-times vs. the log-normal distribution for two typical IT infrastructure events. These two images are very representative of the various observed events data in real IT environments.

Once the events and the conditional probabilities are known for a system, the graph can be constructed as shown in Figure 3. Here all the connections are bidirectional and on the edges the joint probabilities are indicated.

The graph represents the initial probabilistic correlation structure of system anomaly events. The next natural task is to

reduce this graph in a way to preserve all the important aspects while reducing the total number of variables that have to be dealt with.

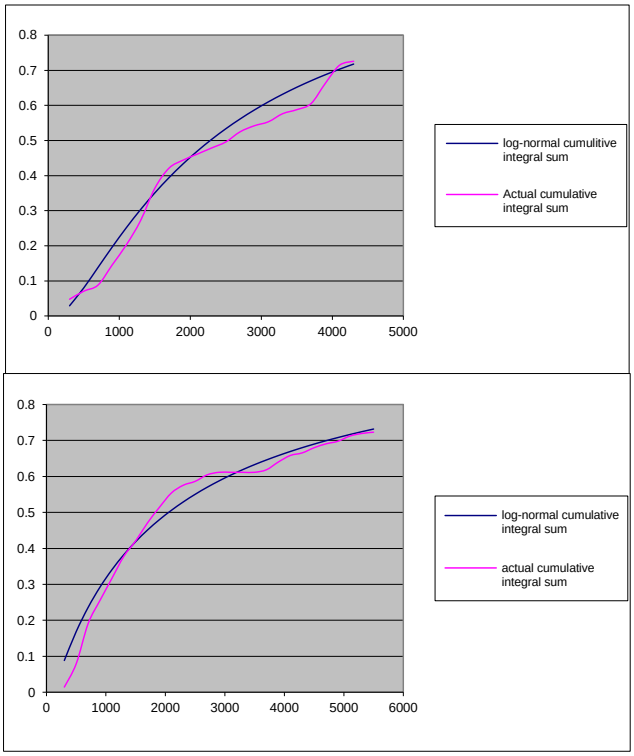


Figure 2. Events lifetimes vs. the log-normal distribution.

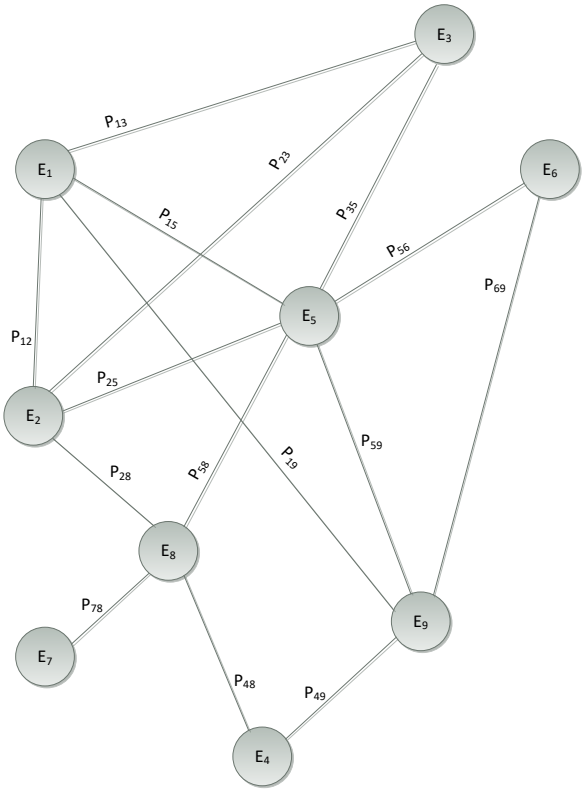


Figure 3. A graph of events with probability connections.

III. Information Measures

The initial graph representing the IT infrastructure events can be of very high complexity. Therefore, we aim to make a reduction of that complexity through elimination of low correlations which are not able to significantly lessen the main information volume contained in the system/graph.

The graph reduction can be accomplished by computing the mutual information [12] contained in correlation between two different events representing random variables:

$$I(E_i; E_j) = \sum_{l,m} P(E_{i_l}, E_{j_m}) \log \frac{P(E_{i_l}, E_{j_m})}{P(E_{i_l})P(E_{j_m})}$$

where indices l and m indicate particular realizations of events E_i and E_j, respectively. This shows that two events within our model contain the following mutual information:

$$I(E_i; E_j) = \log \frac{P(E_i, E_j)}{P(E_i)P(E_j)}.$$

In the latter form it is a measure of independence for the random realizations E_i and E_j.

Note that the mutual information is commutative:

$$I(E_i; E_j) = I(E_j; E_i).$$

To weigh the impact of a random variable event E_i on a set (neighborhood) of correlated events

$$E_i(1), E_i(2), \dots, E_i(N_i)$$

represented by

$$E_i^S \stackrel{\text{def}}{=} \bigcup_{n=1}^{N_i} E_i(n)$$

the conditional entropy measure can be applied:

$$H(E_i^S|E_i) \stackrel{\text{def}}{=} -\sum_{E_i(1),E_i(2),\dots,E_i(N_i)} P(E_i, E_i^S) \log P(E_i^S|E_i)$$

where $P(E_i, E_i^S)$ and $P(E_i^S|E_i)$ are the corresponding joint and conditional probability distributions for the events E_i and E_i^S, respectively.

When coming back to the events modeled in Section II, then the following impact formula weighs the risk of an event E_i together with its influence on a set of correlated events

$$F_1(E_i) = P(E_i) \sum_{n=1}^{N_i} P(E_i(n)|E_i)$$

where

$$\sum_{n=1}^{N_i} P(E_i(n)|E_i)$$

can be interpreted as a "probability flow" from E_i, embracing both the strength of outgoing connections and their coverage. To the leaf nodes we assign a zero impact.

Note that the impact $F_1(E_i)$ does not comprise distant graph connections from E_i, hence quantifying the first order impact. Therefore, we can extend our theory to also include hierarchical impact calculation of nodes. This concerns the case when the events are presumably correlated indirectly, within Δt time window, but also intermediately, within a larger time span ($h\Delta t$) defined by a multiple of Δt. In such a scenario, to E_i a hierarchical impact factor $F_h(E_i)$ is assigned that can be recursively computed over the graph bottom to top in the following way:

$$F_h(E_i) = F_{h-1}(E_i) + \sum_{n=1}^{N_i} F_{h-1}(E_i(n)), \quad h \geq 2.$$

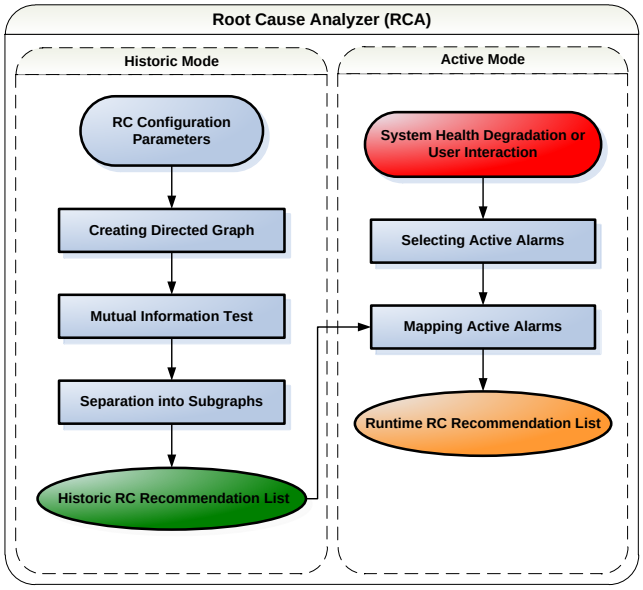
where $F_h(E_i)$ is the h-order impact factor of E_i. For simplicity, we will only reference the first order impact factor omitting the order index h.

IV. The RC Analyzer

Flowchart 1 shows the architecture of the graph-based Root Cause Analyzer (RCA) we apply to obtain run-time problem RC recommendations based on historical events correlation analysis. Therefore, the RC algorithm consists of historic and active modes. To produce an online RC recommendation for a situation with ongoing active alarms that significantly degrade the system's health (estimated by the basic IT management engine), an automatic wake-up of the active mode is foreseen. Alternatively, the run-time RCA can be executed by the user's interaction at any scenario of anomaly events to get the corresponding list of provocative events and associated resources for problem(s) removal.

The particular blocks of the RCA are described in details below.

Flowchart 1. Main blocks of RCA.



1) Correlation graph establishment. Based on the available probability distributions, we generate the directed graph of system anomaly events (Figure 3 depicts a complete graph of pair-wise correlations with joint probabilities P_{ij} on edges). Additionally, we filter out those conditional probabilities $P(E_i|E_j)$ which are obtained artificially without any evidence of E_j preceding E_i.

2) Graph reduction (with statistically derived parameters Δ^+ and Δ^-). Assume a user defined parameter $\varepsilon \in [0,1]$ which regulates the sensitivity of graph reduction. It allows the user to introduce a control on tradeoff between the complexity and accuracy of the analysis. Compute the mutual information for each pair (i,j) and classify those values according to their signs. Let $Q^+_{0.25}$ and $Q^+_{0.75}$ be the 0.25 and 0.75-quantiles of positives, with similar notations for negative data set. The algorithm performs a graph reduction by eliminating "unessential" correlation edges, applying the whiskers model, where the inter-quartile range $Q^+_{0.75} - Q^+_{0.25}$ is an important criterion. Namely, if

$$I(E_i; E_j) < \Delta^+$$

for

$$I(E_i; E_j) \geq 0$$

or

$$I(E_i; E_j) > \Delta^-$$

for

$$I(E_i; E_j) < 0$$

then the edge connecting the node i to j is eliminated, where Δ^+ and Δ^- are defined by the formulas

$$\Delta^+ = Q^+_{0.25} - (0.5 + \varepsilon)(Q^+_{0.75} - Q^+_{0.25})$$

$$\Delta^- = Q^+_{0.75} + (0.5 + \varepsilon)(Q^-_{0.75} - Q^-_{0.25}).$$

Figure 4 illustrates (dash-lined) the edges which are removable according to the above conditions.

In Figure 5 another example is given, where the reduction results in a loss of graph adjacency. The algorithm proceeds with its strategy differently, depending on whether the situation in Figure 4 or in Figure 5 has occurred.

3) Graph connectivity test. This test can be done by an algorithm on the adjacency matrix $A(i,j)$ of the graph ($A(i,j) = 1$ if i-th and j-th nodes are connected, otherwise $A(i,j) = 0$). For this we can apply a bit-wise OR algorithm to the rows of this matrix described in [13]. Then we need to identify each sub-graph. That can be done applying a flood fill coloring algorithm (for its classical version in terms of graphs see the book by Skiena [14]). Next, filter out the nodes which are left without neighbors.

4) Computing impact factors of events. For each node E_i in the reduced graph calculate $F(E_i)$ (for simplicity denoting it by $F(i)$, conventionally calling it "impact factor" of E_i on its neighbor nodes) such that all the conditional probabilities $P(E_i(n)|E_i)$, $n = \overline{1,N}$, exist. Arrange them in decreasing order. If the test against connectivity is negative, then repeat step 4 for all sub-graphs in produced unconnected graph. In

the general case of RC events apply step 2 and arrange the nodes according to decreasing order of the entropies/flows.

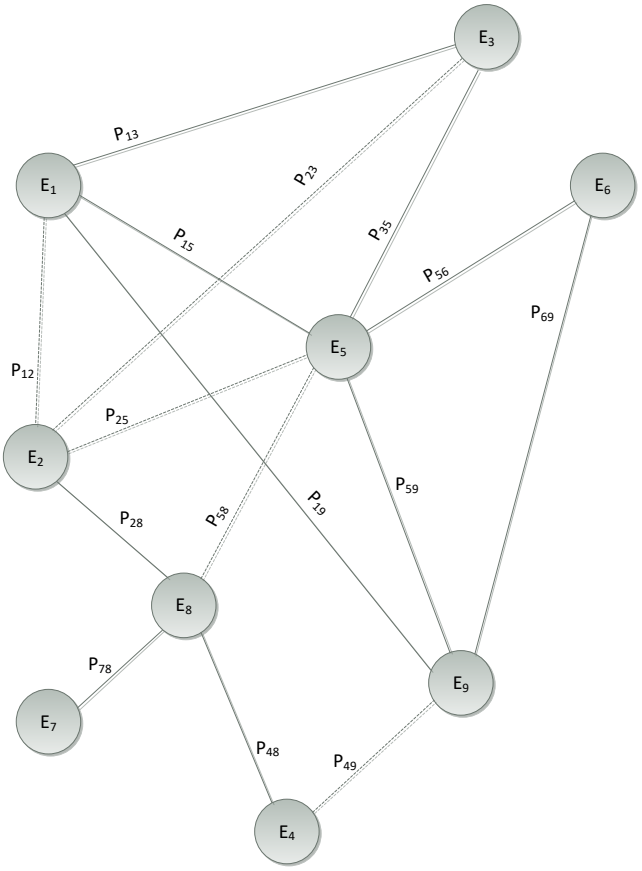


Figure 4. Graph with negligible edges.

5) Obtaining historical list(s) of recommendations. Arrange the nodes in succession of impact factors obtained in step 4. If the test against adjacency is negative, perform the arrangement for every separate sub-graph. Applying prioritization it is possible to order those lists as well. For example, taking the sub-graph with maximum nodes ("the most wide-penetrable area for a RC") and put its recommendation list in the first position in the general recommendation. Equal-size sub-graphs are prioritized according to their "weights", namely, sum of prior probabilities of the nodes comprising the sub-graphs (see step 6). For final construction of historical recommendations, in each sub-list a further filtering is applied to remove the nodes which are left without descendants, i.e., without conditional probabilities which condition on those nodes. Moreover, a user defined parameter is provided to control the number of displayable recommendation sub-lists subject to cardinality criterion.

6) Relative Recommendation Index (RRI) computation. To show the relative "strength" of a recommendation relative to the top element of the recommendation list (with highest index being 100), to each i-th node underneath an index is assigned with values computed by the formula:

$$RRI(i) = 100 - \frac{F(1) - F(i)}{F(1)} 100, \;\; i > 1.$$

It is evident that $RRI(i) \in (0,100]$. To reduce the ultimate displayable recommendation list, a user-defined parameter is set to a value between 0 and 100. It indicates the set of nodes which have RRI above that value.

7) Lists arrangement subject to List Relative Rank (LRR). In analogy with step 6, the separate recommendation lists are also subject to ordering via an index which is called LRR. That rank is defined by the "probabilistic weight" of the corresponding sub-graph. That is the sum of prior probabilities of sub-graph's vertices (events). Let V_l be the set of vertices v of the l-th sub-graph or list (among obtained L lists). The weight of V_l is measured as follows:

$$W(V_l) \stackrel{\text{def}}{=} \sum_{v \in V_l} P(v).$$

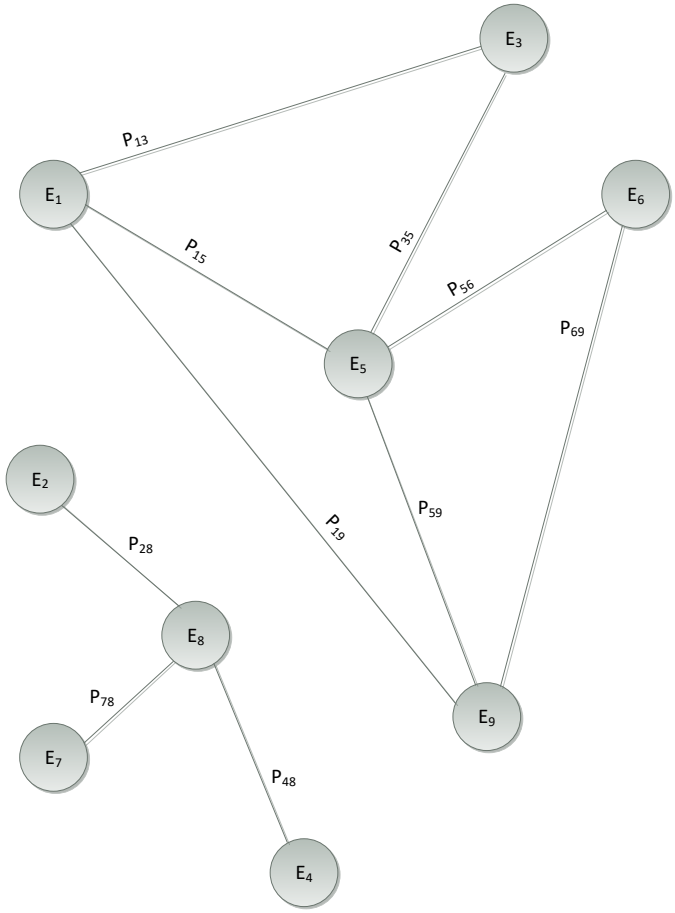


Figure 5. Example of connectivity loss.

Similar to RRI computation, those weights are converted into LRR's. Suppose that they are already indexed in their decreasing order, so $W(V_1)$ is the maximum. Then we assign:

$$LRR(V_1) \stackrel{\text{def}}{=} 100$$

$$LRR(V_l) \stackrel{\text{def}}{=} 100 - \frac{W(V_1) - W(V_l)}{W(V_1)} 100, \quad l = \overline{2, L}.$$

Evidently,

$$LRR(V_l) \in (0,100] \text{ for every } l.$$

8) Active events mapping and displaying the final recommendations. In each historical list, indicate those recommendations which correspond to the current active events (here denoted by $a_{l,s}$ for the l-th list (from L) doubly indexed with $s = 1, \dots, S_l$). Apply the RRI computation method in step 6 to them for each list and display the result as a final recommendation list for on-line RC check. However, the positions for active resources are re-computed according to the "probability flow" and listed in decreasing order. Namely, first calculate the following flows

$$F(a_{l,s}) = \sum_{n=1}^{N_{l,s}} P(a_{l,s}(n) | a_{l,s})$$

for each active node from each list, where $a_{l,s}(n)$ denotes the n-th neighbor of $a_{l,s}$ among $a_{l,s}$.

Let for each l the numbers $F(a_{l,s})$ be already indexed (by s) in their decreasing order. Now apply the RRI technique:

$$RRI(a_{l,1}) = 100$$

$$RRI(a_{l,s}) = 100 - \frac{F(a_{l,1}) - F(a_{l,s})}{F(a_{l,1})} 100$$

where $s = 2, \dots, S_l$, $l = 1, \dots, L$.

9) Impacted resources display. For each recommendation line, a sub-list can be attached with the impacted (by that event) resources. For each recommendation node, separate those neighbors which are end points of the arrows coming from that node and list them in decreasing order of corresponding conditional probabilities. Then apply the RRI technique of step 6 on those conditional probabilities to further organize the list of impacted resources.

V. Bottleneck Analyzer

The bottleneck is a network component or group of components (resources) with significant and permanent impact on the network.

We define a bottleneck as a resource(s) with persisting presence in network failures, highly recommendable in historic recommendation lists, and largely deviating from the rest of resources in its RRI.

In this context, a correlation graph-based Bottleneck Analyzer is designed to optimize the users' efforts in bottlenecks localization/identification and removal. It automatically produces a guide in form of a recommendation list for the user to advice the most probable bottleneck resources. If the bottleneck is not unique, then it recognizes separable origins resulting in several recommendation lists. The module can be configured to give recommendations on different hierarchical levels of the system topology. Each recommendation in a list is positioned according to its likelihood or RRI. The latter shows how much the recommendation deviates from the top one in its confidence. In case of multiple bottlenecks, the respective parallel recommendation lists are prioritized according to their LRR's. Flowchart 1 resumes the main blocks of this algorithm.

Table I shows an example of bottleneck generating resources. In this case the impact generating areas are isolable and hence potential RC's are separable. Detected bottlenecks are marked by bold and red. Their RRI's deviate largely from the rest of resources in the same list. In the third sub-list the absence of the bottleneck is due to the closeness of the RRI's. In the fourth sub-list the absence of the bottleneck is due to the shortness of the list. The resource ID's are associated with the events they generate. Here the sub-lists and resources with low LRR's and RRI's, respectively, are not included into the table.

Flowchart 1. Main blocks of Bottleneck analyzer.

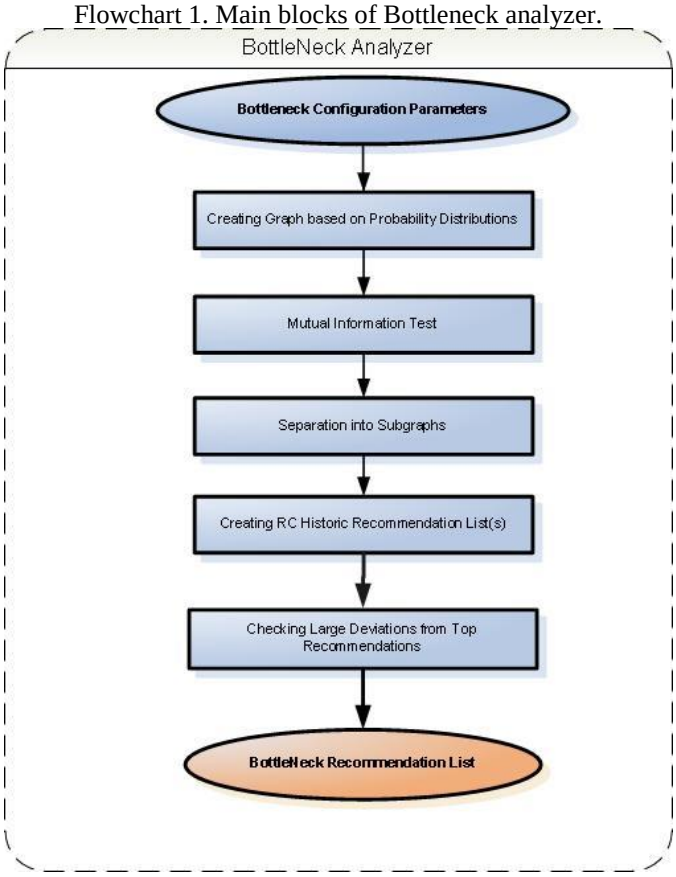


TABLE I. BOTTLENECKS IDENTIFIED IN HISTORIC RL'S.

Sub-List N	LRR	Resource	RRI
1	100	**27**	**100**
		26	**99**
		17	**98**
		18	20
		8	18
		11	18
		25	18
2	90.2	**14**	**100**
		24	56
		15	55
		10	40
		32	40

		33	39
3	40.7	12	100
		29	99
		23	98
		9	98
		13	97
		22	97
4	21.9	28	100
		19	50

VI. Criticality Patterns and Black Swans

Graph-based modeling and analysis of anomaly events enables additional automation capabilities in terms of pro-active fault management in IT environments. There are graph patterns (nodes, paths, and sectors) that constitute important criticality structures showing different scenarios of failure propagation. To instrument the AECE with this pattern detection functionality, we designed a Critical Component Analyzer (CCA) to categorize and reveal critical nodes, connections, and sectors in the system. It operates in two modes: historic and active. The historical analysis identifies those criticality patterns and produces related categorizations for the past events data. In the active mode, the algorithm performs a mapping of current alarms set into the "historical picture" and indicates the risk of run-time criticalities.

We define the following categories of criticality as graph patterns, where the idea of second and higher order impacts play an important role in terms of fault propagation:

- *Critical Node* is a node that has prior probability that is equal to or greater than the value of an upper threshold (see nodes B, G, K and P in Figure 6 for upper threshold = 0.9).
- *Root* is a node that has no "incoming" arrow, in other words, it is historically an impacting-only resource (see node D in Figure 6).
- *Critical Path* is a sequence of impacting nodes with extremely high conditional probabilities on the connections (see the path BRSTU in Figure 6).
- *Extreme Path* is a Critical Path with nodes that have prior probability that is equal to or greater than the value of an upper threshold (see the path HBAO in Figure 6 where the upper threshold is equal to, say, 0.7). Note that in the active mode the Critical and Extreme Paths are an appropriate source of predictions).
- A *Critical Sector* of some magnitude *M* (defined by its nodes volume or percentage) is a connected sub-graph with the joint probability connections all higher than some value.

There is another category of event systems that is classified under the concept of *Black Swans* discussed in the next subsection.

The above mentioned patterns are detected, in particular, for log event data [7].

A. Black Swan Type Events

We define the *Black Swan* as a graph node which has very low prior probability, but extremely high impacting ability on a large set of neighbors (see Figure 6). Then we define the concept of the black swan type event. The *Black Swan Event* is a set of black swan nodes that in terms of significant impact covers a large set/sector of nodes. Actually the black swan event is a class of rare events that inevitably leads to a system crash.

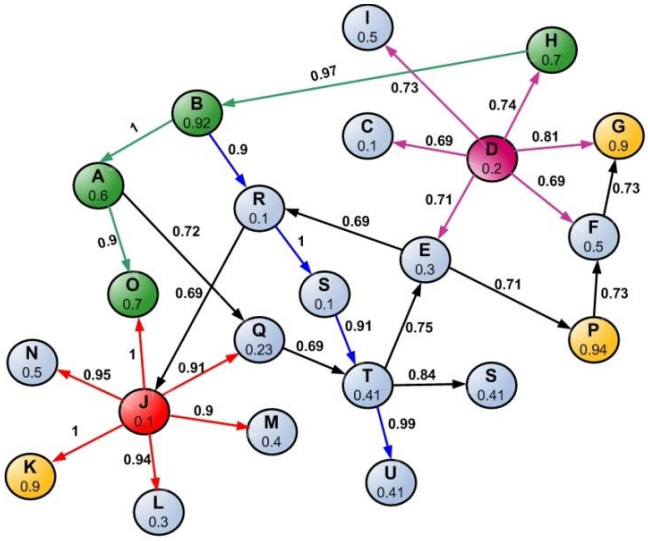


Figure 6. Criticality patterns in a directed graph.

A correlation graph-based Black Swan Analyzer is designed for revealing black swan type events in the system and on-line calculation of black swan risk. Given the above definitions, the main steps of the algorithm are the detection of black swan nodes and then the black swan events.

Step I. Black Swan Node determination. Formally, the black swan node is a node that has a small prior probability and large number of strong outgoing connections. In other words, it is a rare event-node with extremely high impacting ability on the neighbors in problematic situations. Simple queries can identify those nodes in the graph.

Step II. Black Swan Event determination. The *Black Swan Event* is a set of black swan nodes which with high conditional probabilities influence on a predefined area of the system or nodes. In Figure 6 two BSNs (J and D) impact 72% of the system (green nodes). A simple check identifies whether the black swan nodes constitute a black swan event.

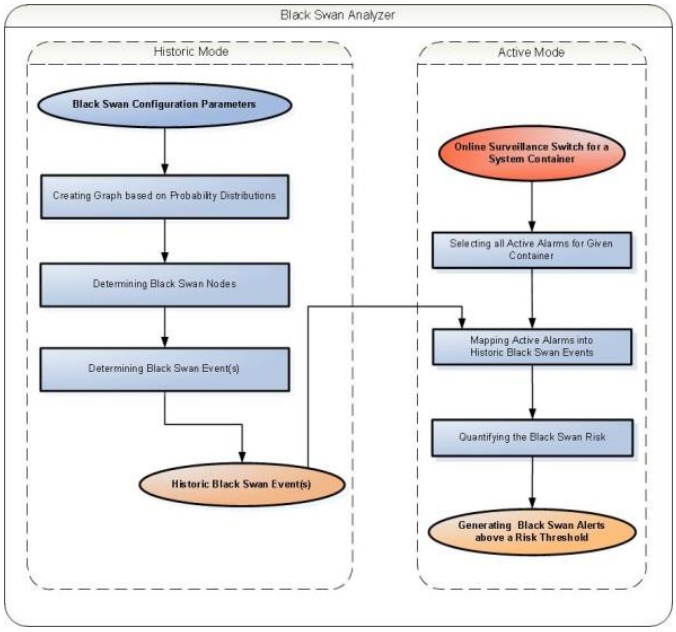
The above two steps belong to the historical part of analysis of the network. In online mode, it is always important to evaluate the risk of a black swan type event based on a registered set of abnormality alerts.

Step III. Black Swan Risk estimation. This kind of risk evaluation starts by matching the active alarms with the black swan nodes. Every black swan node contributes to the risk estimate based on its impact (a weighted coefficient) on the neighbors. In particular, 100% risk means that all black swan nodes have active alarms and it will probably lead to the

system crash based on the high impact that such nodes have on the neighbors.

Flowchart 2 shows the main components of this analyzer with indication of historic and active working modes. The active mode is executed when we start to follow the run-time status of an IT resource collection (container).

Flowchart 2. Black swan detection and risk estimation.



The quantification of the black swan risk (BSR) varying from 0 to 100 is performed as follows. Let the black swan event is determined by the black swan nodes that cover $C\%$ (a threshold) of the graph. In on-line mode, if at the time moment T the black swan nodes impact/cover $R\%$ of those $C\%$ of nodes, then the black swan risk R_{BS} as a function of T is

$$R_{BS}(T) = R\%.$$

Based on this analysis the management system can alert on high black swan risks for particular system containers as Flowchart 2 shows.

VII. On Experimental Results

The RC detection algorithm was successfully tested on data sets of several online banking companies with very complex IT infrastructures. Explicit presentation of numerical results might contain some sensitive information to those companies. Therefore, we'll discuss some common and abstract aspects of those outcomes.

The experimental set-up was focused on the data centers of those companies with a retrospective application of the vC Ops analytics to their IT resource metrics and relevant anomaly events processing. The experimental time span was chosen appropriately to involve the most "painful" (implying revenue and business impact) to those companies and their customers' out-of-service dates and prior period of DT alarms to have enough statistics. The reasons implied those "outage dates" were detected by manual intervention of the IT administrators with substantial investment in system remediation. Given this side information, the aim of our experiments was to retrospectively deduce the root causes that we would recommend run-time on those critical outages (with application of our engine at those infrastructures) and compare them with ones mined by the customers alternatively (non-automated, time and effort consuming manner).

In all cases, the historical causes of system crashes were indicated within the top (1st to 3rd) positions of on-line recommendation lists. The qualitative aspect of this fact is that the historically most impact generation IT resources were the causes of real-time system outages. It means also that the impact computation graph-based method introduced here is a practically efficient tool to deploy it broadly.

In those experiments, only the first order impact was used to obtain the ranked list of historical and active RC lists. This fact interestingly shows that even for deeply sophisticated infrastructures the RC identification can be achieved by a naïve (simplest) impact quantifying formula. Moreover, that quantification has the lowest possible complexity in terms of the impact computation for each graph node that is a recursive process for the higher order factors.

Table II illustrates a typical recommendation list for prediction of RC's (in this case isolable) of a system abnormality. The resource ID's are associated with the events they generate.

At the same time this table is a fragment of results from real experiments on abnormality events (DT violations) generated by our IT management module for an online business infrastructure. Our methodology was able to correctly indicate the causes of a system down for a day that significantly impacted the company's business and its users' activities. Here the sub-lists and resources with low LRR's and RRI's, respectively, are not included into the table.

For construction of the corresponding probabilistic graph we processed more than 10,000 DT violation events registered during the monitoring and surveillance of this online banking company's infrastructure for approximately two months. Based on the historical RC lists derived from the correlation graph, we were able to produce the right recommendation for problem localization at the time of system outage with high number of alarming resources reported by our DT engine.

Tabke II. Recommendation List with Two Sub-Lists.

LRR	RESOURCE ID	RRI	IMPACTED RESOURCES
100	8	**100**	9, 7, …
	7	85.09	9, …
	9	84.82	8, 7, …
67.4	5	**100**	6, …
	6	80.86	…

The recommendation lists can be consecutively refined via a user feedback option of the technology. The algorithms shown above can be extended to incorporate user feedback information into the strategy to arrive at a better convergent solution of the recommendation list. This can be accomplished

by asking the user a set of questions and appropriately modifying the algorithm with the user response. According to the answers, we assign different scores to the resources in recommendation lists to use in further computations.

The question that arises in using a purely data agnostic and statistical analysis is, will the convergence of the answers to the actual RC be fast enough?

To accelerate the convergence, a user feedback mechanism is also incorporated which allows the algorithm to test the validity of its recommendation and improve its future recommendations. In general, the convergence rate is proportional to the RRI deviation of the desirable recommendation against the top recommendation.

In the same feedback context, further experimental insights can be achieved by monitoring and collecting statistics from the user on his/her satisfaction in RC recommendations in everyday issues that are not necessarily serious system breakdowns.

VIII. Conclusions

We introduced a new model of statistical inference for management of complex IT infrastructures based on their anomaly events data obtained from an intelligent monitoring engine. We demonstrated the components of the relevant prototyped module AECE that employs a directed virtual graph showing relationships and conditional probabilities between event pairs. An information-theoretic processing of this graph allows us to reliably predict the root causes of problems in IT systems. This technique was successfully applied to a number of data sets from real business environments running on huge IT infrastructures. The approach is also capable of identifying bottlenecks of network performance and evaluating risk of black swan type events along with several other critical patterns.

References

[1] A. Alaeddini and I. Dogan, “Using Bayesian networks for root cause analysis in statistical process control”, *Expert Systems with Applications*, vol. 38, issue 9, 2011, pp. 11230-11243.

[2] S. Pradhan, R. Singh, K. Kachru, and S. Narasimhamurthy, “A Bayesian Network Based Approach for Root-Cause-Analysis in Manufacturing Process”, *IEEE Int. Conf. on Computational Intelligence and Security*, Harbin, China, Dec. 15-19, 2007, pp. 10-14.

[3] S. Dey and J.A. Stori, “A Bayesian network approach to root cause diagnosis of process variations”, *International Journal of Machine Tools and Manufacture*, vol. 45, issue 1, 2004, pp. 75-91.

[4] R.L. dos Santos, J.A. Wickboldt, R.C. Lunardi, B.L. Dalmazo, L.Z. Granville, L.P. Gaspary, C. Bartolini, and M. Hickey, “A solution for identifying the root cause of problems in IT change management”, *Proc. Mini Conf. 12th IFIP/IEEE IM*, Dublin, Ireland, May 23-27, 2011, pp. 586-593.

[5] N.N. Taleb, *The Black Swan: The Impact of the Highly Improbable*, Random House, 2007.

[6] M.A. Marvasti, A.V. Grigoryan, A.V. Poghosyan, N.M. Grigoryan, and A.N. Harutyunyan, “Methods for the cyclical pattern determination of time-series data using a clustering approach”, US patent application 20100036857, published 2010.

[7] M.A. Marvasti, A.V. Poghosyan, A.N. Harutyunyan, and N.M. Grigoryan, “Pattern detection in unstructured data: An experience for a virtualized IT infrastructure”, to appear in *Proc.IFIP/IEEE International Symposium on Integrated Network Management*, Ghent, Belgium, May 27-31, 6 p., 2013.

[8] J.P. Martin-Flatin, G. Jakobson and L. Lewis, "Event correlation in integrated management: Lessons learned and outlook”, *Journal of Network and Systems Management*, vol. 17, no. 4, December 2007.

[9] G. Jakobson and M. Weissman, "Alarm correlation", *IEEE Network*, vol. 7, no. 6, pp. 52–59, November 1993.

[10] S. Kliger, S. Yemini, Y. Yemini, D. Ohsie and S. Stolfo, "A coding approach to event correlation", in Proc. *4th IEEE/IFIP International Symposium on Integrated Network Management* (ISINM 1995), Santa Barbara, CA, USA, May 1-5, 1995, pp. 266–277.

[11] M. Hasan, B. Sugla, and R. Viswanathan, "A conceptual framework for network management event correlation and filtering systems", in Proc. *6th* IFIP/IEEE *International Symposium on Integrated Network Management* (IM 1999), Boston, MA, USA, May 24-28, 1999, pp. 233–246.

[12] T.M. Cover and J.A. Thomas, *Elements of Information Theory*, Wiley, 1991.

[13] N. Deo, *Graph Theory with Applications to Engineering and Computer Science*, Prentice Hall, 1974.

[14] S.S. Skiena, *The Algorithm Design Manual*, Springer, 2008.