Problem Localization for Automated System Management in Ubiquitous Computing*

Shunshan Piao, Jeongmin Park, and Eunseok Lee**

School of Information and Communication Engineering Sungkyunkwan University
300 Chunchun Jangahn Suwon, 400-746, Korea
{sspiao, jmpark, eslee}@ece.skku.ac.kr

Abstract. The increasing complexity of Ubiquitous computing leads to the challenges in managing systems in an automated way, which accurately identifies problems and solves them. Many Artificial Intelligent techniques are presented to support problem determination. In this paper, a mechanism for problem localization based on analyzing real-time streams of system performance for automated system management is proposed. We use Bayesian network to construct a compact network and provide both inductive and deductive inferences through probabilistic dependency analysis throughout the network. An algorithm for extracting a certain factors that are highly related to problems is introduced, which supports network learning in diverse domains. The approach enables us to both diagnose problems on the underlying system status and predict potential problems at run time via probabilities propagation throughout network. A demonstration focusing on system reliability in distributed system management is given to prove the availability of proposed mechanism, and thereby achieving self-managing capability.

Keywords: Fault Diagnosis, Prognosis, Problem Localization, Probabilistic Dependency Analysis, Self-Managing.

1 Introduction

With the rapid growth in size and flexibility in distributed computing systems nowadays, complexity appears frequently in all places especially in Ubiquitous Environment. Due to the fact that the more the requirements demanded, the more the complexities created [1], it brings much more burdens and hardness for administrators to handle abnormities and maintain high system reliability, which is very important to system manager for managing the computer system and to users for running their applications. However, as autonomic computing [2] requirements emerged, self-managing ability appears on the IT stage as a challenging topic. It implies that the system can recover from faults on its own initiative instead of system administrators'

* This work was supported in parts by Ubiquitous Autonomic Computing and Network Project, 21th Century Frontier R&D Program, MIC, Korea, ITRC IITA-2006-(C1090-0603-0046), Grant No. R01-2006-000-10954-0, Basic Research Program of the Korea Science &Engineering Foundation, and the Post-BK21 Project.
** Corresponding author.

M. Denko et al. (Eds.): EUC Workshops 2007, LNCS 4809, pp. 158–168, 2007.

direct handling, for the purpose of providing services to maintain high reliability without interruptions. As faults are unavoidable in the whole lifecycle of computer systems, problem localization techniques [3] generates a variety of challenging applications for the Artificial Intelligent techniques to provide fault localization technology, root cause analysis and other approaches applied to the fields of problem analysis.

With current increasing complexity, knowledge of the system and environment is not sufficient as we need to analysis the exact cause of unexpected problems in large scale of distributed environment; so much as exceptions and abnormalities occur without any anticipation. Existing techniques such as rule-based or case-based algorithms are not competent. In some cases, it is not popular in uncertain domain with missing information and inferring with low accuracy, and it becomes large size as increasing states [4]. Moreover, most existing researches on analyzing causes of problems [5] focus on post-treatment, which means that dealing with problems is time consuming, error-prone, and requires much experience and prior information.

Problem localization is a process of deducing the exact root cause of problems based on a set of observed information. Clearly, it is critical to designing an effective self-managing system, by which the system determines and solves problems automatically. In this paper, we propose a mechanism for fault localization based on Bayesian machine learning method to determine the cause of problems and also enable it to forecast under given observations via probabilistic dependency analysis. We add preconditioning course before learning structure, which improves the efficiency of structure learning without degrading the quality of learning. Following the proposed approach in performance problem domain, bidirectional inferences including fault diagnosis and prognosis are possible to conduct automated system management in complex distributed system, and hence improve system reliability.

The rest of this paper is organized as follows. First, we provide related work on autonomic computing and list some problems in existing research, which focuses on fault diagnosis and fault management. Second, following the introduction of Bayesian network fundamental, we describe the proposed fault localization model structure in detail, then introduce preprocessing and structure learning. Third, we examine a straightforward application of learning network and discuss how to implement problem localization under the proposed approach. In the last section, we conclude this paper and provide directions for future research.

2 Related Works

Self-managing system tasks in Ubiquitous environment such as real-time fault localization and problem diagnosis, call for higher levels of automation. Many recent studies introduce various methods for automated system management [6], attempting to explore new approaches to improve self-managing capability, such as IBM self-aware distributed systems and Sun fault management in predictive self-healing.

- IBM Self-Aware Distributed Systems

IBM research on self-aware distributed systems aims at automating an increasingly complex and expensive task of real-time problem diagnosis in large-scale distributed system by using state-of-art machine learning - Bayesian inference, probabilistic

reasoning and information –theoretic approaches. It shows an architecture of diagnosis system called RAIL (Real-Time Active Inference and Learning), which uses the probe outcomes to make inferences about the system state, and actively requests the next most-informative probes to improve its diagnosis. [7]

The most current focus of the work is on:

- Active diagnosis: Adjusting the probe set dynamically to improve diagnosis;
- Extending local approximation techniques to incremental, real-time scenarios;
- Handling intermittent failures, dynamic routing, and other nonstationarity in the network state and behavior using on-line learning;
- Active learning using flexibility in probe selection.

- Sun Fault Management Utilities in Predictive Self-healing

The Sun Fire X4500 server features the latest fault management technologies. This technology is incorporated into both the hardware and software of the server. Predictive Self Healing introduces a new software architecture and methodology for fault detection, diagnostics, logging, and system service management across Sun's product line. There are two major components in Predictive Self Healing [8]: Fault Management Architecture (FMA) and Service Management Facility (SMF).

Predictive self healing addresses two problems of commercial IT:

- Fix problems before they occur
- Circumvent operational problems with services

A critical event prediction for proactive management describes an attempt to build a proactive prediction and control system for large clusters either through prediction algorithms or root cause solutions using probabilistic networks, including time-series, rule-based classification and Bayesian network models [1]. Furthermore, a hybrid prediction model in ubiquitous computing system adopts a selective model according to the system context, using various algorithms with respective characteristics, which can predict system situations before errors occur [4].

Various machine learning algorithms are used in the automated system management. However, most of them rarely consider dependency relationships between collected information. In the case of such a situation, with the fact that there exist somewhat interrelated relationships between system metrics, it can start with representing a probabilistic dependency model among system elements rather than deeming them mostly independent.

3 Problem Localization

A key essential of self-managing is the ability of the system to perform real-time inferences and learning about its own behavior, to diagnose and predict various problems and performance degradations, namely, the capability of self-awareness. “Suit the remedy to the case”. Only with the root cause of a problem can we make the system take appropriate actions or repair strategies to solve the problem. Furthermore, adding proactive prediction ability makes it prevent from unexpected loss through pretreatment, and hence achieve automated system management. Fault diagnosis and prognosis based on real-time streams of computer events contribute to self-managing

for the purpose of determining root causes of problems i.e. fault localization and predicting future situations such as potential problems that going to occur.

In this paper, we use probabilistic machine learning method, which is mainly used as a modeling tool, to propose an inference model structure for fault diagnosis and prognosis in self-managing systems. It inferences the likelihood that a factor is in one state which is dependent on other factors' states that reflect the degrees of confidence. In terms of accuracy and efficiency of diagnosing problems and forecasting potential problems, we can deal with the data in the raw beforehand then combine prior information for inference.

3.1 Proposed Model Structure

Bayesian network based problem localization model structure is described in Fig. 1.

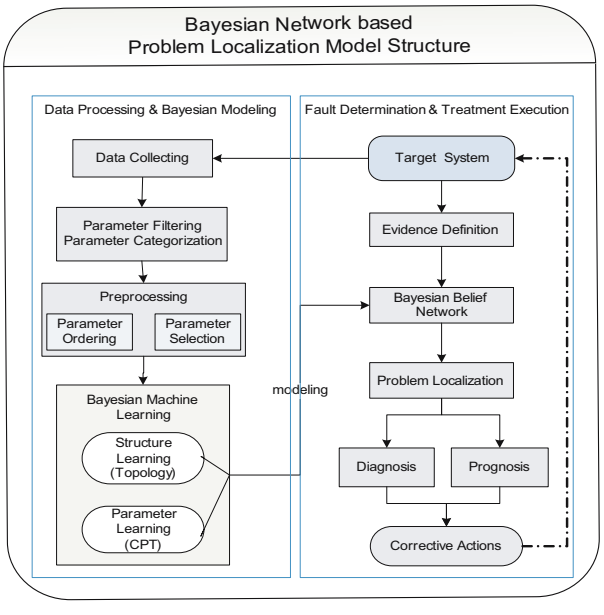

Fig. 1. Bayesian Network based Problem Localization Model Structure

- Data Collecting: System real-time performance data is collected which also includes the system health states stored in the log file from target system via monitoring.
- Parameter Filtering & Categorization: The dataset, collected from the system log file, consists of real-time continuous parameters which are then discretized.
- Preprocessing: This processing course, mainly affecting the ultimate inference, is processed before using a Bayesian network. Herein we propose an approach based using information theory among filtered parameters, select a certain parameters and rank them in a node list that will be applied in structure learning.
- Bayesian Machine Learning for modeling Bayesian Belief network: 1) Topology Structure Learning: It finds a network structure that is most probable matching to

the training data. 2) Parameter Learning: It decides on the conditional probability table of each node by learning from training data given a created network.

- Evidence Definition: This defines degree of confidence information which is called evidence by presenting with probabilities.
- Problem Localization including Diagnosis & Prognosis: Decided evidences are posted to constructed network to reason out $P(Cause \mid Effect)$ or $P(Effect \mid Cause)$ in different cases for determining high impact factor of faults or predicting potential problems under certain conditions.

The parameter with the highest probability in the network is determined as the cause after probability propagation when making inferences. According to inference results, corrective repairs are taken to running system in order to keep continuous operation without pause. Analyzing statistic data from a given system, we can find patterns of system without knowing the inner running mechanism and conduct inference based on this.

3.2 Fundamental and Characteristics of Bayesian Network

Bayesian network or Bayesian belief network is a graphical structure to represent and reason about an uncertain domain, including nodes represent random variables of interest in the domain and arcs represent direct influences i.e. conditional dependencies between variables. It emphasizes that a link between two nodes does not, and need not, always imply causality, i.e. the network is not always a causal structure. It only implies a direct influence of parent node over child node in the sense that the probability of child node is conditional on the value of parent node, and two nodes may have a link between them even if there is no direct cause [9]. The formula (1) expressed below is a simple representation of Bayes' rule.

$$P(A \mid B) = \frac{P(B \mid A) \cdot P(A)}{\sum_{i} P(B \mid A_i) \cdot P(A_i)} \tag{1}$$

For more complex problems, it also has a mechanism that can propagate probabilities via extending Bayes' Rule throughout the whole network automatically. If a Bayesian network encodes the true independence assumptions of a distribution, we can use a factored representation for the distribution as follows:

$$\begin{aligned} \mathrm{P}(x_1, \ldots, x_n) &= \prod_{i=1}^{n} \mathrm{P}(x_i \mid x_{i+1}, \ldots, x_n) \\ &= \prod_{i=1}^{n} \mathrm{P}(x_i \mid \mathrm{Pa}(x_i)) \end{aligned} \tag{2}$$

Formula (2) shows that instead of the full joint distribution, we need only the conditional probabilities of a variable given its parents, which is based on Markov assumption. A distinct characteristic of Bayesian network is that it is especially useful in uncertainty domains with information about the past and/or the current situation being vague, incomplete, and conflicting. It's easy to explain how a system arrived at a particular recommendation, decision, or action as it can represent probabilistic relationships between nodes dynamically. Furthermore, Bayesian Network can be run in multiple directions, including bottom-up and top-down, which features of Bayesian

Network are applied in this paper. Another feature is that it can post evidence to a Bayesian belief network to predict a result or to diagnose a cause based on analyzing current beliefs. The evidence is information about a current situation and beliefs are the probability that a variable will be in a certain state based on the addition of evidence in a current situation [10].

3.3 Preprocessing

Although Bayesian network structure can be created by experts based on domain knowledge [11], more researches are interested in learning Bayesian network from data automatically. Learning structure is more crucial part of the whole course and the final results are directly related to it. Recently many methods for structure learning have been developed, finding the structure that is most suitable to training data. The score based search method uses approximate search algorithms to construct candidates and measures them using scoring evaluation. The dependency analysis method starts with analyzing dependency relationships between nodes to construct a network. However, both methods are not suitable when there are larger data, which in this case brings overfitting which is one of the main issues in using machine learning. The overfitting phenomenon occurs when too many parameters are considered in a given domain. In building Bayesian network structure, it occurs when considering too many parameters in structure learning. So in order to solve such problems and make structure learning more efficient, we can provide preconditioning course previously.

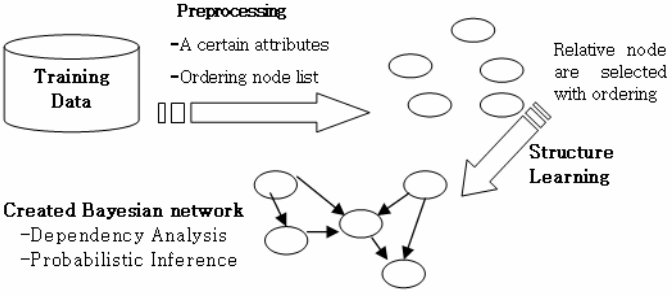

Fig. 2. Preprocessing of Training data

Fig.2 depicts the process before Bayesian network structure learning. Given training data, it selects certain relative factors with ordering, and then enters into the step of structure learning, on which probabilistic dependency analysis are based.

There are two phases included in preprocessing. First, from the given large dataset with more parameters, it can only consider factors that are more relative with focusing problems, i.e. choose relative factors describing the domain. We can downsize the number of factors by using information theory method to analyze relationships or clustering or other approaches. After determining certain factors, it arranges them in a special order, which means anterior one has direct influence on the posterior one in the same direction of arrow, by analyzing information gain between pairs of observing data. However, it emphasizes the assumption is that problematic parameters

are independent of each other when learning structure. All parameters in the ordering lists are able to have influence on each problematic parameter; thereby each problematic problem has the same ordering list only with each different problematic factor as the last one, which implies that all the factors in front of it could be a parent node of the last one. The pseudo code of the ordering algorithm is described as follows.

```
Input: Separate observing parameters from problematic parameters stored in set S= {V1 ...Vn, P1...Pm}.
Output: An ordering list with a certain number of parameters
1) Select a certain observing parameters with high relevance to problematic parameters.
   for each problematic parameter Pj ( j= 1 to m) do
        for i = 1 to n do
        compute information gain Gij = Gain( Vi,Pj) = H(Pj)-H(Vi,Pj) (H() means entropy)
        end for
        rank parameters with Gij from maximum to minimum and save them to list Lj
    end for
Combine all lists Lj (j =1 to m), select observing parameters with the mean information gain exceeds
defined threshold value.
Return the selected parameters and all problematic parameters. S'= {V1...Vk, P1...Pm} (k<n)
2) Make an ordering list for the selected observing parameters in set S'
   Initialize set S''={V1,...,Vk} except for problematic parameters; pair set P={empty}; list L={empty}
   Select two parameters Vx and Vy from the head of set S'' (x!=y)
   Compute Gain(Vx, Vy) and Gain(Vy, Vx) for each pair, put pair(Vx-> Vy) with larger Gain into set P
   stop when there is close loop, run until all parameters in set S'' are considered.
   Sort the pairs in set S'' to an single ordering list L
   Return an ordering list L only with observing parameters
```

Applying an ordering node list into the next step of learning, for score based search method, it can reduce the entire search space when adding link to construct network, as a node can be parent only of node which is behind it according to the ordering node list; for dependency analysis method, it can reduce computing complexity as the number of nodes is decreased and determine the direction between two nodes.

3.4 Structure Learning

In this paper, an ordering node list with certain parameters is used as input to create a fine-grained model by analyzing conditional independency evaluation, which determines dependency relationships between all pairs of nodes. It should be stressed at this point that Bayesian network implies conditional independencies via showing conditional probability tables for leaf nodes having direct parent nodes.

Bayesian network structure learning from data presents an efficient algorithm based on the conditional independence (CI) test to measure dependency relationships [12]. In this paper, one of the structure learning mechanisms, which begin with the definition of Bayesian network, is based on computing mutual information introduced in the Information Theory for pairs of nodes to reflect different degrees of dependency relationships among them. A threshold is given to determine the existence of probabilistic dependency relationship between nodes.

4 Experiment and Evaluation

Following the rapid growing internet systems in the Ubiquitous computing era, violations of service level objectives [13] are related to reliability of system and quality of service. As automated management capability described in self-managing, when there are faults such as bottlenecks, violation of Service Level Objectives occurred, the system should find which factor is directly related to them and affect high level performance of system automatically, by analyzing observed parameters consisting of performances of individual servers or processes, capability of network, hardware and software, dynamic variation resource utilizations by different types of client requests, and temporary traffic situation. Thereby, they can be used to determine which part of the system is responsible for current fault of the system, then it is repaired appropriately; oppositely, the collected information can be used to forecast system potential problems, preventing them in advance.

In our experiment, it collects and filters data of interest that can be used for analysis, including CPU, memory, disk utilization, count of client, package volume, bandwidth logged in a server and detects information such as threshold violation in response time and throughput, on which we rely to analyze and control system management for providing high quality of service and performance, as described in Table 1. Then, after collecting sample data, each parameter should be categorized into corresponding classes according to given criteria, such as High, Medium, Low for performance parameters and Error, warning, normal for problematic parameters.

Table 1. Training parameters

System metrics (Performance parameters)							Bottlenecks, SLO violation	
CPU uti.	RAM uti.	Bandwidth	Filesize	Client count	Disk uti.	...	Response time	Throughput
70%	67%	3.1	35	159	72%	...	3.6	65
65%	58&	2.8	28	132	58%	...	4.1	58
56%	72%	4.6	42	126	69%	...	3.6	43
...	...	...	...	...	...	...	...	...
49%	61%	3.1	38	110	50%	...	4.5	57

$$S = \{V_{cpu}, V_{ram}, V_{disk}, V_{bandwidth}, V_{client}, V_{filesize}; P_{response}, P_{throughput}\}$$

We take above parameters as input to create node ordering with certain number of parameters which are highly related to problematic parameters. After learning on the training data, the result of selecting relative parameters is:

$$S' = \{V_{bandwidth}, V_{client}, V_{ram}, V_{cpu}; P_{response}, P_{throughput}\}$$

Then the observing parameters are ranked by using the proposed approach to output a node ordering list without problematic parameters, as follows:

$$Orderinglist = \{V_{cpu} -> V_{ram} -> V_{bandwidth} -> V_{client} -> P_{response}, P_{throughput}\}$$

With the predefined assumption, the problematic parameters response time and throughput are independent of each other. From the above ordering list, it implies that the node orderings can be used when constructing a Bayesian network.

$$P = \{V_{cpu} -> V_{ram}, V_{ram} -> V_{bandwidth}, V_{bandwidth} -> V_{client}, V_{client} -> V_{response}, V_{client} -> V_{throughput}\}$$

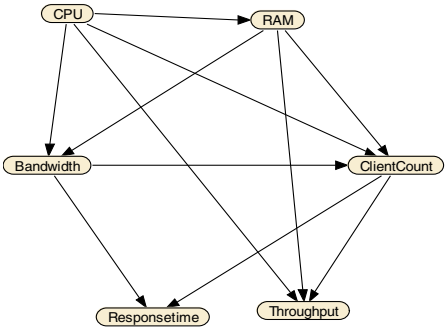

Fig. 3. Structure Learning

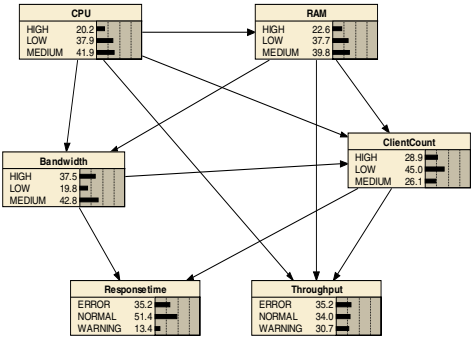

Fig. 4. Parameter Learning

From Fig. 3 we can see that the created structure is a compact hierarchy model after learning from certain parameters and ordering list. In contrast to the simple structure of Naïve Bayesian network, it discovers and represents internal dependency relationships between each pair of causal parameters in the network structure, which makes the results of inferences more accurate. The next learning phase is parameter learning given structure and training data i.e. fixing conditional probabilities for each node. Fig.4 describes the complete Bayesian network after parameter learning.

The inference courses based on probabilistic dependency analysis can be carried out given the created model, and including inductive and deductive reasoning. Given the convinced states of several parameters, it makes the known state with 100% belief, which operation can change beliefs of all nodes that related to such one after probability being propagated throughout the whole network. For instance, a violation of response time is occurred that makes response time be of error state, the most probable impact factor can be found that low class of bandwidth with the highest probability; on the other hand, when the utilization of CPU resource arrives at 95% utilization which belongs to high class with 100% evidence, the probability of error state of throughput can get up to be the highest, which means that there will be a fault of throughput appeared in coming time. These results derived from diagnosis and prognosis are very helpful for system to take correct repairs to figure out faults or to avoid potential faults in advance. From the probabilistic network, it's easy for us to understand how the factors affect each other by changing the evidences of nodes with dynamic representation.

However, in order to estimate the effect of inference following our approach, we apply testing data into the built model then compare the results with the actual outcome. At first, we evaluate the time consumption of structure learning and error rate given different numbers of parameters, showing the obvious effect of using a certain number of parameters that highly correlate with the domain. From Fig. 5, we can find that as the number of parameters grows, the time consumption mounts up but the error rate of detecting faults drops and the number of parameters corresponding to the crossing of two curves can be chosen as an appropriate quantity for considering the parameters in such a domain.

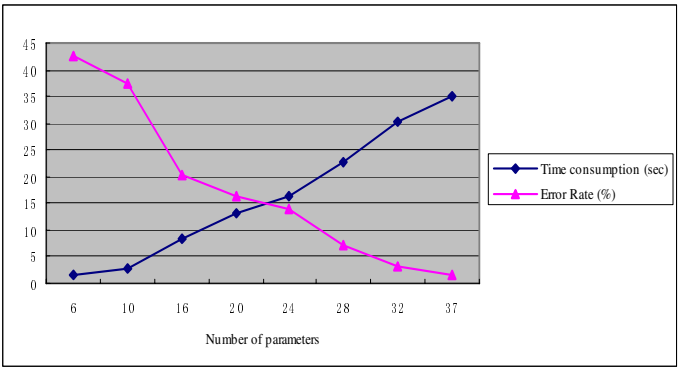

Fig. 5. Evaluation with different number of nodes

Furthermore, comparison of time consumption and accuracy is evaluated in the case of selecting 24 parameters regardless of being provided with a node ordering list. Table 2 can tell us that with the node ordering list, there are improvements both in time consumption and accuracy of inferring results.

Table 2. Comparison on cases with and without node ordering

Dimensions	Time consumption (sec)	Accuracy (%)
with node ordering	15.48	90.3
without node ordering	16.37	85.2

In conclusion, the comparison of structure learning under a given certain quantity of parameters and ordering list, points out taking the ordering list as input of structure learning accounts for that preconditioning of parameters in the process is much more effective, regardless of whatever learning method is used.

5 Conclusion

In this paper, a Bayesian Network based mechanism of problem localization for automated system management in Ubiquitous computing is proposed. In order to improve the performance of learning with domain knowledge, a preprocessing step which reduces the size of parameters is added to improve the whole process of Bayesian network modeling. Using the proposed methods, we can transform a complex system into a compact model with high efficiency and accuracy, on which we depend to make inference via probabilistic dependency analysis. In contrast to other existing researches on using Bayesian network, it can process input data in advance, which is implemented with high accuracy to improve the efficiency of structure learning. In order to prove availability of the proposed approach, we perform it in the system performance domain to achieve automated system management and make various comparisons under different conditions.

Our future work will continue to research on machine learning, which is considered as an Artificial Intelligent approach for self-managing system to learn real-life streams of events that expresses health states and faults. There are many algorithms[14] used in various fields for machine learning, including time-series, decision Tree, case-based reasoning, rule based reasoning and so on. Following these methods, it can provide multiple functions in fields of diagnosis, prognosis, fault isolation and root cause analysis.

References

1. Sahoo, R.K., Oliner, A.J., Rish, I., Gupta, M., Moreira, J.E., Ma, S., Vilalta, R., Sivasubramaniam, A.: Critical event prediction for proactive management in large-scale computer clusters. In: Proceedings of the ACM SIGKDD, Intl. Conf. on Knowledge Discovery and Data Mining, August, pp. 426–435 (2003)
2. Kephart, J.O., Chess, D.M.: IBM Thomas J. Watson Research Center: The Vision of Autonomic Computing. IEEE Computer Society, Los Alamitos (2003)
3. Steinder, M., Sethi, A.S.: A Survey of Fault Localization Techniques in Computer Networks. Science of Computer Programming. Special Edition on Topics in System Administration 53(2), 165–194 (2004)
4. Yoo, G., Park, J., Lee, E.: Hybrid Prediction Model for improving Reliability in Self-Healing System. In: SERA 2006. ACIS International Conference on Software Engineering Research, Management & Application, pp. 108–115. IEEE Computer Society, Los Alamitos (2006)
5. Rish, I., Brodie, M., Ma, S., Odintsova, N., Beygelzimer, A., Grabarnik, G., Hernandez, K.: Adaptive Diagnosis in Distributed Systems. IEEE Transactions on Neural Networks (March 2005)
6. Dai, Y.-S.: Autonomic Computing and Reliability Improvement. In: ISORC 2005. Proceedings of Eighth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, pp. 204–206 (2005)
7. IBM Self-Aware Distributed Systems: http://domino.watson.ibm.com/comm/research.nsf/pages/r.ai.innovation.2.html
8. Sun Microsystems: Predictive Self-Healing in the Solaris 10 Operating System: http://www.sun.com/bigadmin/content/selfheal
9. Alpaydm, E.: Introduction of Machine Learning. © 2004, Massachusetts Institute of Technology
10. Charles River Analytics Inc: About Bayesian Belief Networks. © Copyright, Charles River Analytics, Inc. (2004)
11. Ding, J., Kramer, B., Bai, Y., Chen, h.: Backward Inference in Bayesian Networks for Distributed Systems Management. Journal of Network and Systems Management 13(4) (2005)
12. Cheng, J., Bell, D.A., Liu, W.: An Algorithm for Bayesian Belief Network Construction from Data. In: Proceedings of AI &STAT, pp. 83–90 (1997)
13. http://www.risi.com/services/sla.html
14. Vilalta, R., Apte, C.V., Hellerstein, J.L., Ma, S., Weiss, S.M.: Predictive algorithms in the management of computer systems. IBM Systems Journal issue 41-3, Artificial Intelligence 41(3) (2002)