INDUCTIVE SYSTEM HEALTH MONITORING WITH STATISTICAL METRICS

David L. Iverson
NASA Ames Research Center
Moffett Field, CA

ABSTRACT

Model-based reasoning is a powerful method for performing system monitoring and diagnosis. Building models for model-based reasoning is often a difficult and time consuming process. The Inductive Monitoring System (IMS) software was developed to provide a technique to automatically produce health monitoring knowledge bases for systems that are either difficult to model (simulate) with a computer or which require computer models that are too complex to use for real time monitoring. IMS processes nominal data sets collected either directly from the system or from simulations to build a knowledge base that can be used to detect anomalous behavior in the system. Machine learning and data mining techniques are used to characterize typical system behavior by extracting general classes of nominal data from archived data sets. In particular, a clustering algorithm forms groups of nominal values for sets of related parameters. This establishes constraints on those parameter values that should hold during nominal operation. During monitoring, IMS provides a statistically weighted measure of the deviation of current system behavior from the established normal baseline. If the deviation increases beyond the expected level, an anomaly is suspected, prompting further investigation by an operator or automated system.

IMS has shown potential to be an effective, low cost technique to produce system monitoring capability for a variety of applications. We describe the training and system health monitoring techniques of IMS. We also present the application of IMS to a data set from the Space Shuttle Columbia STS-107 flight. IMS was able to detect an anomaly in the launch telemetry shortly after a foam impact damaged Columbia's thermal protection system.

INTRODUCTION

Model-based reasoning is a powerful method for performing system monitoring and diagnosis. Typical model-based reasoning techniques compare a system model or simulation with system sensor data to detect deviations between values predicted by the model and those produced by the actual system[1]. In effect, a model-based reasoner uses the collected system parameter values as input to a simulation and determines if a particular set of input values is consistent with the simulation model. When the values are not consistent with the model a "conflict" occurs, indicating that the system operation is off nominal (when compared to the presumably correct model)[2]. One disadvantage of model-based reasoning is that building models is often a difficult and time consuming process. For many applications, the Inductive Monitoring System (IMS) provides a method that can monitor system health with nearly the same fidelity as a model-based reasoner, but without the need to manually build a model.

IMS automatically defines groups of consistent system parameter data by examining and generalizing from examples of nominal system data. If a system model were available, a set of parameter values selected from one of these groups produced by IMS would typically compute without conflicts when processed by the model. With a sufficiently broad training data set, IMS will produce a knowledge base that contains most or all of the parameter value combinations necessary to effectively characterize and monitor nominal system operation. After learning how the system behaves when operating correctly, IMS can identify off nominal behavior and trigger appropriate alert messages for system operators.

In the following sections, we describe how IMS uses nominal data to characterize normal system behavior. We then describe how this behavior characterization can be exploited to determine if system operation is off-nominal. Finally, we present details of the application of IMS to a Space Shuttle monitoring task.

LEARNING NOMINAL SYSTEM BEHAVIOR

IMS automatically builds monitoring knowledge bases from nominal data sets collected either directly from the system or from high fidelity simulations. Machine learning and data mining techniques are used to characterize typical system behavior by extracting general classes of nominal data from archived data sets. In particular, IMS uses *clustering* to group sets of consistent parameter values found in the training data. Clustering is the unsupervised assignment of elements of a given set to groups or clusters of similar points[3]. The implementation of IMS described here draws from two clustering techniques: K-means clustering[4] and density-based clustering,[5] as we describe below.

DATA VECTORS

The basic data structure of the IMS algorithm is a vector of system parameter values. (Fig. 1) Each vector is an ordered list of parameters collected from the monitored system by a data acquisition process. Additionally, the vectors can contain derived parameters computed from collected data. These vectors define the points in N-dimensional space that will be grouped by the IMS clustering algorithm. The values used in a given vector may be collected simultaneously by the data acquisition system, or collected over a period of time. The user specifies the size and content of the vector structure that is most appropriate for the monitoring application. IMS does not distinguish between raw data values and values derived from the raw data, which allows a wide range of data preprocessing options. More informative vectors will typically contain interrelated parameters that tend to vary together as the system operates. Some time dependent behavior may be captured by including rates of change for one or more parameters, or by combining several consecutive data samples in a single vector structure.

Pressure A	*Valve 1 Position*	*Pressure B*	*Valve 2 Position*	*Pressure C*	*Temperature 1*	*Temperature 2*
2857.2	86.4%	1218.4	96.2%	1104.1	49.8	37.6

Figure 1. Sample IMS vector

BUILDING CLUSTERS FROM NOMINAL DATA

IMS processes the training data by formatting input data into the predefined vector format and building a knowledge base containing clusters of related value ranges for the vector parameters. (Fig. 2) Each cluster defines a range of allowable values for each parameter in a given input vector. The vector of high values and the vector of low values in a cluster can be thought of as corners defining a minimum bounding rectangle in N-space. Since the rectangles are defined by nominal data, points that fall inside or very near the rectangles are considered to be within the system's nominal operating range. The high and low ranges for each element in the cluster can also be considered as allowable ranges for the corresponding parameter, provided the other parameters are within their respective ranges specified in that cluster. This view is similar to model-based reasoning with interval arithmetic where simulations are performed using a range of possible values for each parameter, rather than a single value[6].

	Pressure A	*Valve 1 Position*	*Pressure B*	*Valve 2 Position*	*Pressure C*	*Temperature 1*	*Temperature 2*
High	2857.6	86.8%	1219.2	96.3%	1105.0	50.1	38.2
Low	2855.8	86.2%	1215.7	95.5%	1103.2	49.6	37.5

Figure 2. Sample IMS cluster structure

The IMS training process consists of the following steps:

1. IMS starts the training process with an empty cluster database.
2. It reads the nominal training data and formats it into vectors.
   It can be useful to scale or normalize the data values before they are inserted in the vectors. For instance, each parameter can be scaled to represent a percentage of the maximum expected range for that parameter. This eliminates a tendency for changes in parameters with larger ranges to overshadow changes in those parameters with smaller ranges. Conversely, data values can also be relatively scaled to provide weighting of more significant parameters. For instance, scaling a parameter to have a larger possible range relative to other parameters in the vector will amplify deviations in that parameter.
3. The first vector is inserted in the database as the initial cluster.
4. Each subsequent input vector is compared to the contents of the cluster database to find the cluster that is closest to the vector.
   To calculate the distance between a given cluster and a vector, a point contained in the cluster is selected and the distance between that point and the vector of interest is computed with a distance formula. Various methods can be used to select the point in the cluster, depending on desired results. One option, based on the K-means clustering method[4], measures the distance from the centroid of the cluster found by forming a vector from the average of the high and low values for each cluster parameter. Various methods can also be used to compute the distance between a vector and a point in a cluster, e.g., the Euclidean distance metric.
5. IMS then determines if the input vector is contained in the bounding rectangle defined by the closest cluster. If so, the database remains unchanged.
6. If the input vector falls outside of the cluster, IMS determines if it is close enough to be incorporated into the cluster.
   As in density-based clustering[5], a threshold value, ε, specified by the user defines the maximum allowable distance between a cluster and vector to determine if the vector should be incorporated into the cluster.
   a. If the vector is close enough (distance less than or equal to ε), the cluster parameter intervals are expanded as necessary to include the new vector.
   b. If the distance between the training vector and the closest cluster in the database is greater than ε, a new cluster containing the vector is formed and inserted into the database.

This learning process repeats until all of the training data has been processed and incorporated into clusters in the knowledge base. Note that the value of ε can be adjusted to balance knowledge base size and monitoring speed versus monitoring tolerance. A larger ε value will tend to produce a smaller knowledge base that allows for real-time monitoring at a higher data rate. A smaller ε value will result in smaller clusters that provide tighter monitoring tolerance, but will sometimes produce a larger than desired monitoring knowledge base.

After IMS processes the training data, the result is a database of clusters that characterize system performance in the operating regimes covered by the training data. In essence, each cluster defines constraints on the values allowed for each parameter in any particular monitoring input vector. The parameter values in a vector must meet these constraints for that vector to be considered nominal data. If there is no cluster in the database that contains a given input vector or is "near" that input vector, then the system is behaving in an unexpected manner, indicating a possible system anomaly.

SYSTEM HEALTH MONITORING

Because IMS generalizes training data to characterize nominal system operation, the resulting cluster knowledge base can be used for system health monitoring. For this task, IMS simply formats the real time data into vectors and queries the knowledge base to locate the cluster that is closest to each input vector. Two techniques are available for monitoring: strict cluster matching without a distance indication, or fuzzy matching using a distance measurement. The strict IMS monitoring scheme requires that the input data vectors be contained inside at least one of the knowledge base clusters (all parameter values must be within the ranges specified by the cluster limits) to be considered nominal. This scheme is fastest since it eliminates the need to perform distance calculations. The fuzzy scheme, on the other hand, will locate the cluster in the monitoring knowledge base that is closest to the input vector, and report the distance of that vector from the cluster. This more informative, but slower, scheme will give the operator an idea of how far the system behavior is deviating from nominal operation as represented by the training data.

In many cases the training data set does not completely cover the nominal operating space. In these cases it may be desirable to consider a monitored data vector as nominal when it falls *near* an established cluster. Suspicion that there is an anomaly would then grow as the vectors move further away from the nominal clusters. The distance value at which a data point becomes suspect will vary between different monitoring applications, and will further depend on the number of parameters monitored, scaling applied to those parameters, etc. It is beneficial to standardize the distance/deviation values reported by the monitoring system to ease interpretation of the results. One approach for this is to use statistical distance metrics.

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-x^2/(2\sigma^2)}$$

Equation 1. Gaussian Distribution with Mean of zero and Standard Deviation σ

STATISTICAL DISTANCE METRICS

If the training data set provides adequate coverage of nominal system behavior, it is reasonable to assume that the set of distances that monitored nominal data points (vectors) fall from their nearest cluster will fit a common statistical distribution. For example, most nominal points that are not contained in their nearest cluster will fall very close to the cluster, with fewer nominal points falling in areas further away from the cluster. This situation can be characterized by the Normal (Gaussian) statistical distribution presented in equation 1, where *P(x)* gives the probability that a nominal data point will fall a distance *x* from the cluster. This quality makes it convenient to view the proximity of a point to the nearest cluster as a measure of the probability that the point is a member of the set of nominal values represented by that cluster. Applying a bell-shaped fuzzy set membership function similar to those described by Yin[7] allows us to quantify and normalize the probability that a given data point is nominal. This fuzzy membership function is implemented by using equation 1 to calculate the probability that a point a distance x away from the nearest cluster represents nominal behavior. Since IMS is monitoring for system failures, it is more intuitive to report the likelihood that a data point is anomalous than that it is normal. This can be accomplished by calculating 100 * $(P(0) - P(x))$ to give a value between 0 and 100 that the data represents an anomaly. $P(0)$ returns the maximum value for equation 1, signifying that the data point lies within the cluster, i.e., the distance between the point and the cluster is zero.

The shape of the fuzzy membership function will depend on the value of the standard deviation, σ, selected for equation 1. A larger σ will increase the width of the bell curve, decreasing the reported likelihood that points nearby the cluster are anomalous. An effective method for determining σ for the membership function consists of these three steps:

1. Run additional representative nominal system data (not used in the training process) through the IMS monitoring routine and collect the raw distance data for those points.
2. Find the standard deviation, σ_0, for that data set.
   Raw distance values less than or equal to $2\sigma_0$ represent about 96% of the points that should be encountered during nominal system monitoring, so it is reasonable to assume that raw distance excursions much beyond $2\sigma_0$ from nominal could be cause for concern.
3. Select an acceptable value as a monitoring caution (yellow) limit, e.g. 20%, and set the value of σ in equation 1 so 100 * $(P(0) - P(2\sigma_0))$ equals approximately 20.

These values are presented as an example and should be adjusted to match each particular monitoring application. During system monitoring, raw distance values for each data sample are processed through equation 1 and reported to the user. Since equation 1 has been tailored to return values near or below an established caution threshold during normal operation, output below that threshold can be considered nominal. An increasing level of alert can be raised as values climb above this threshold.

It may also be useful to track the distribution of raw distances over time to discover trends that may indicate behavior that differs from the baseline distribution described by the currently active fuzzy membership function. For example, find the standard deviation of raw IMS distances collected over a period of time and compare that value to the σ_0 that was previously calculated to set the monitoring caution level. Significant changes in raw distance standard deviation could indicate declining system health.

REAL TIME PERFORMANCE

In order to use the IMS generated cluster database for real time or near real time system monitoring, an efficient cluster indexing and retrieval scheme may be required. Several applicable schemes have been developed in the area of nearest neighbor searching[8]. In order to allow searching the database for the closest cluster, the scheme must include a distance metric and the ability to return the record that is nearest to the query point, not just those that contain the query point. The search and retrieval speed must also be sufficiently fast to keep up with the expected data acquisition rate. An efficient indexing and retrieval scheme can also help to speed up the initial IMS training process since training requires similar closest cluster queries. IMS applications to date have successfully monitored 1 KHz data rates without distance calculations and 50 Hz data rates with distance calculations on computers running with clock speeds under 1 GHz. When run on a 295 MHz Ultra-SPARC processor, monitoring rates in excess of 5 KHz were obtained on the Space Shuttle data analysis presented in the next section. This Shuttle analysis included distance calculations. In the current implementation, IMS monitoring speed will typically decrease as the size of the monitored vector increases, since additional calculations are required for each extra parameter. Larger cluster knowledge bases can also decrease average monitoring speed.

IMS APPLICATION EXAMPLE

While some domain knowledge is usually required to define effective IMS vectors, the IMS methodology is domain independent and can be used in a variety of system monitoring situations. Some possible application domains include aerospace, transportation, manufacturing, power generation and transmission, medicine, or process monitoring. Of particular interest to NASA is the application to Integrated Systems Health Management (ISHM), either on board a vehicle or in a mission control room. To demonstrate the utility of IMS in a mission control setting, we recently developed an IMS knowledge base to monitor temperature sensors in the wings of a Space Shuttle Orbiter, and used that knowledge base to analyze archived telemetry data collected from the ill-fated STS-107 Columbia Space Shuttle mission. This flight came to a disastrous end when the Columbia orbiter was destroyed during reentry, claiming the lives of all seven crew members. The ultimate cause of the accident was determined to be a breach in the Thermal Protection System on the leading edge of the left wing, caused by a piece of insulating foam that struck the wing approximately 82 seconds after launch. The first indication of the damage that was noticed by mission controllers monitoring telemetry data was not seen until Orbiter re-entry, 17 days after launch. That anomaly was a slight increase in a left main landing gear brake line temperature that occurred about seven minutes before the destruction of the vehicle[9].

IMS STS-107 ANALYSIS

The post mission IMS analysis of the STS-107 Columbia flight concentrated on telemetered data from temperature sensors in the wings of the orbiter. Analyses of telemetry data from two flight phases, launch/ascent and on-orbit, were conducted. In both analyses, IMS detected anomalies much earlier in the data than monitoring systems available in mission control. This example will focus on the STS-107 launch and ascent analysis.

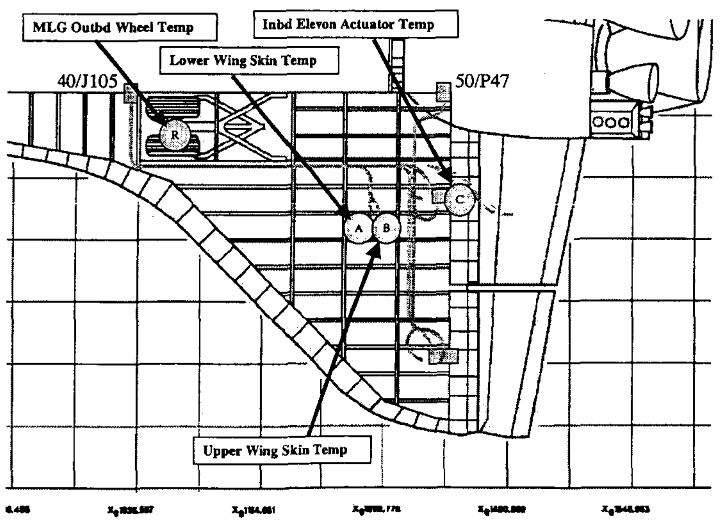

Figure 3. Left Wing Temperature Sensors used in Columbia Ascent Analysis

IMS knowledge bases for the launch and ascent analysis were generated from training data collected during 10 previous Columbia flights. Separate knowledge bases were generated for each wing. Training vectors were formed from four corresponding temperature sensors in each wing of the Orbiter. (Fig. 3) Since ambient temperatures differed on each flight, the data vectors were normalized to a reference sensor (Main Gear Outboard Wheel Temperature) by expressing the other sensor values relative to the value of that sensor in each telemetry time slice. Thus, the resulting vectors contained three parameters each. The data sets used for training and analysis covered the time period from launch through ascent to just before Main

Engine cut off. The data was sampled at a 4 Hz rate. The resulting left wing knowledge base contained 1212 clusters and the right wing knowledge base contained 840 clusters. The difference in knowledge base size between the two wings may be due to larger fluctuations in the nominal sensor data collected from the left wing. Since IMS monitors data produced by the same sensors that generated the training data, distinctive characteristics of the sensors are automatically captured and accounted for in the monitoring knowledge base.

IMS STS-107 ANALYSIS RESULTS

The results of the IMS analysis of the STS-107 Columbia launch are graphed in Figure 4. The horizontal axis represents time, beginning at the moment of lift off. The vertical axis represents the IMS measure of deviation from nominal behavior, that is, the distance of the input

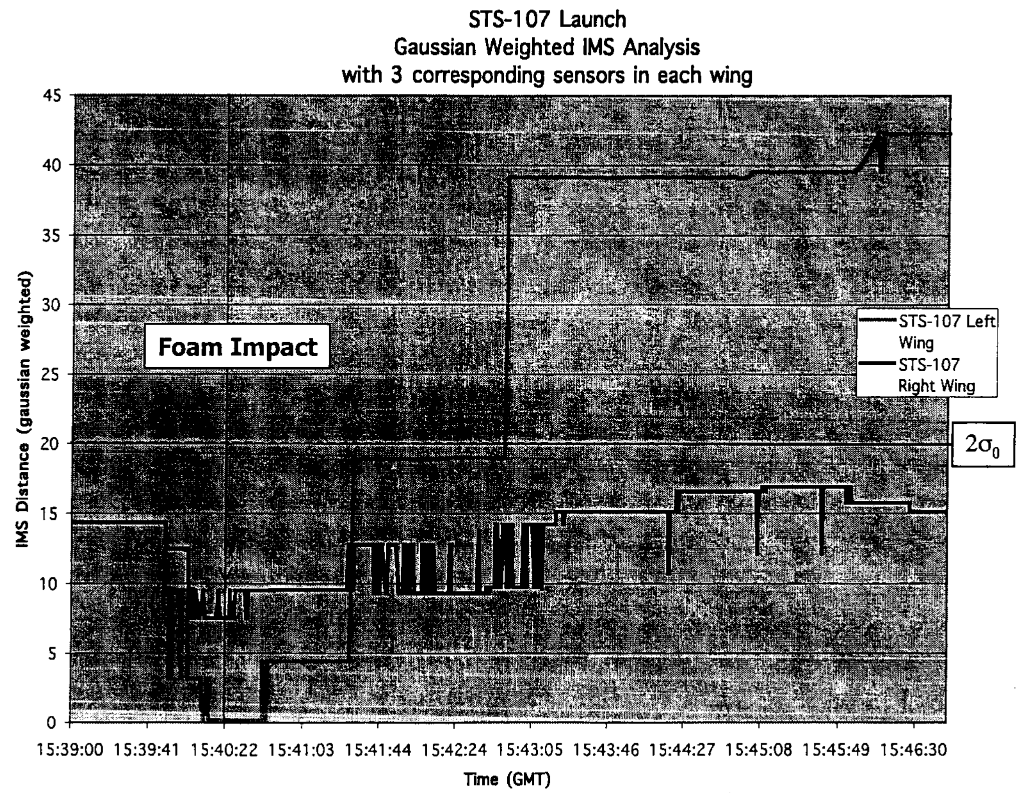

Figure 4 – Results of IMS analysis of STS-107 Columbia launch

vector from the closest nominal cluster. The distance measure has been scaled with the Gaussian membership function described above. IMS monitoring results for the left wing are represented by the lighter colored line. Right wing results using corresponding sensors are shown as a darker line for comparison. The vertical line near time 15:40:22 shows the moment of the foam impact event that breached the Thermal Protection System on the left wing. Notice that the IMS results for the left and right wings remain fairly low until shortly after the foam impact, at which point the values for the left wing begin to diverge sharply. The IMS deviation values for the right wing continue to show results within a reasonable range of nominal, while the left wing deviation values increase to nearly three times those of the right wing. The standard deviation (σ) value for the membership function was set to return a value near 20 as the IMS distances approached two standard deviations ($2\sigma_0$) of the raw distances found in nominal data, as described previously. Due to limited training data availability, the raw distances used to determine σ_0 were gathered by running data from each nominal flight through separate IMS knowledge bases generated using the other nominal data sets rather than running additional nominal data through the full STS-107 monitoring knowledge base. During STS-107 launch monitoring the right wing values remained well below the caution threshold of 20, while the left wing values significantly exceeded that threshold soon after the foam impact. Although this analysis was performed off-line using archived data, the techniques used could be implemented for real time monitoring. Significant deviations in a group of sensors or asymmetrical results for identical sensor sets, especially of large magnitude such as shown in this analysis, may provide indication of anomalies earlier in a mission than current telemetry monitoring tools.

SUMMARY AND CONCLUSIONS

These early results from the Inductive Monitoring System (IMS) show that it is feasible to automatically construct a useful system monitoring knowledge base from archived system data using clustering techniques. These knowledge bases could provide system monitoring capability comparable to that obtained by model-based reasoning techniques, without requiring the cost and effort of manually building detailed system models. In addition, the IMS monitoring routine may be used for remote or onboard, real time or near real time system monitoring. As a mission control tool, IMS could help augment controller awareness of vehicle health and provide early detection of possible anomalies. As shown by the STS-107 analysis, IMS revealed evidence of the Thermal Protection System breach within minutes of the foam strike while current mission control tools did not detect symptoms of the problem until Orbiter re-entry, 17 days later.

FUTURE WORK

In addition to applying IMS to various monitoring applications, future work will address parameter selection, efficiency, system health metrics, and integration with other ISHM components. Various analytical techniques, including statistical variance analysis and Principal Components Analysis, will be tested for their ability to provide insight into informative parameters for inclusion in IMS vectors. Modifications will be made to the cluster search algorithm in an attempt to increase the speed of both the training and monitoring processes. Techniques will be explored to decrease the size of the cluster knowledge base by combining similar clusters. We will also investigate alternate system health indicators, in addition to the current distance measurements, e.g., the tracking of raw distance distributions mentioned previously. Finally, we plan to explore diagnostic applications of IMS, including extracting diagnostic information from the IMS cluster matching routines, and integrating IMS with diagnostic reasoning systems.

REFERENCES

1. Dvorak, D. and Kuipers, B., ***Model-Based Monitoring of Dynamic Systems,*** Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89), Morgan Kaufman, Los Altos, CA., 1989.

2. Reiter, R., ***A Theory of Diagnosis from First Principles***, Artificial Intelligence, 32(1):57-96, Elsevier Science, 1987.

3. Bradley, P.S., Mangasarian, O.L., and Street, W.N., ***Clustering via Concave Minimization***, Advances in Neural Information Processing Systems 9, Mozer, M.C., Jordon, M.I., and Petsche, T. (Eds.), pp 368-374, MIT Press, 1997.

4. Bradley, P.S. and Fayyad, U. M., ***Refining initial points for K-means clustering***, in Proceedings of the International Conference on Machine Learning (ICML-98), pp 91--99, July 1998.

5. Ester, M., Kreigel, H-P, Sander, J., and Xu, X., ***A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise***, Proceedings of the 2nd ACM SIGKDD, pp 226-231, Portland, OR, 1996.

6. Hamscher, W.C., ***ACP: Reason maintenance and inference control for constraint propagation over intervals***, Proceedings of the 9th National Conference on Artificial Intelligence, pp 506-511, Anaheim, CA, July, 1991.

7. Yin, T-K, ***A Characteristic-Point-Based Fuzzy Inference System Aimed to Minimize the Number of Fuzzy Rules***, IEEE Trans. Fuzzy Systems, vol. 12, no. 2, pp. 250-273, April 2004.

8. Kleinberg, J.M., ***Two Algorithms for Nearest-Neighbor Search in High Dimensions***, Proceedings of the 29th Annual ACM Symposium on Theory of Computing, pp 599-608, El Paso, TX, May, 1997.

9. Gehman, H.W., et al., ***Columbia Accident Investigation Board Report***, U.S. Government Printing Office, Washington, D.C., August 2003.