Discovering Atypical Flights in Sequences of Discrete Flight Parameters

Suratna Budalakoti, University of California, Santa Cruz, suratna@soe.ucsc.edu,
Ashok N. Srivastava, Ph.D., NASA Ames Research Center, Ashok.N.Srivastava@nasa.gov,
Ram Akella, Ph.D., University of California, Santa Cruz, akella@soe.ucsc.edu

Abstract— This paper describes the results of a novel research and development effort conducted at the NASA Ames Research Center for discovering anomalies in discrete parameter sequences recorded from flight data [1] [2].Many of the discrete parameters that are recorded during the flight of a commercial airliner correspond to binary switches inside the cockpit. The inputs to our system are records from thousands of flights for a given class of aircraft and destination. The system delivers a list of potentially anomalous flights as well as reasons why the flight was tagged as anomalous. This output can be analyzed by safety experts to determine whether or not the anomalies are indicative of a problem that could be addressed with a human factors intervention. The final goal of the system is to help safety experts discover significant human factors issues such as pilot mode confusion, i.e., a flight in which a pilot has lost situational awareness as reflected in atypicality of the sequence of switches that he or she throws during descent compared to a population of similar flights. We view this work as an extension of Integrated System Health Management (ISHM) where the goal is to understand and evaluate the combined health of a class of aircraft at a given destination.

1. Introduction

Previous approaches to the task of anomaly detection focus on continuous sensor data [2], and do not distinguish discrete sensors from continuous, thus disregarding the non-continuous as well as the sequential nature of the discrete sensors. In comparison, we focus on discrete sensors, specifically, sensors recording pilot actions, or switches. We are interested in the sequence in which the values for these sensors change during the course of a flight and finding anomalies in flight behavior based on this information.

Our system performs two tasks, as part of the task of atypical events detection in flights: a) detection of atypical flights, b) finding events during the course of such flights that are anomalous or atypical. Task b) is important as each flight generates large amounts of data during its course, and simply identifying a flight as anomalous still leaves the problem of identifying the problem areas inside the flight unaddressed.

We treat the problem of finding atypical flights as an unsupervised learning problem. We first cluster the flights for each itinerary into groups, and identify the outliers in each cluster as atypical. We use the Longest Common Subsequence, a common measure in bioinformatics and Intrusion Detection systems, as the similarity measure for clustering flight data. We then present two new algorithms that use Bayesian Networks to efficiently identify anomalous events during the course of the flight. We demonstrate the performance of these algorithms using operation information from about 10,000 flights, and developing the base clusters and locating anomalous flights by using these sequences.

Sequence analysis is an active and much-studied area in computer science. Some areas where sequence analysis algorithms are prominent are anomaly-based intrusion detection in computer system/network, and in bioinformatics. The problem of anomaly detection in aircraft data, in fact, has many similarities to the problems of network intrusion detection. Anomaly based network intrusion detection techniques work by forming some sort of model of what constitutes normal activity in a network. Any deviation from this normal behavior is flagged as anomalous. Our approach to the problem of anomaly detection in aircraft data has similarities to the work of Sequeira and Zaki [3], in that both use sequence analysis based methods, though there are significant differences in detail.

2. Data Description and Preprocessing

The flight data the prototype system was implemented over was derived from binary sensors in the aircraft during the landing phase of 6400 flights. All of these flight has the same destination airport. The data was stored as a $T \times N \times F$ matrix, where T is the number of distinct

[1] 0-7803-9546-8/06/\$20.00©2006 IEEE
[2] This work was supported by the NASA Aviation Safety Program, Aviation Systems Monitoring and Modeling element.

observations for the sensors, N is the number of sensors, and F is the number of flights. So, we had a $T \times N$ matrix for each flight. This matrix was reduced to a one-dimensional sequence as follows:

1. The initial value of each sensor was assumed to be zero.
2. At any time-step t, only the sensors that show a transition in value, were recorded in the final sequence.

The above transformation gave us a dataset consisting of 6400 sequences, with sequence lengths varying over a wide range, between 600 and 9000 characters. Following this, we performed another reduction step, where all sensors that changed values an average of thirty or more times were removed from the sequences. This was done with the assumption that sensors that changed values so frequently were not recording pilot actions, but aircraft system response to pilot actions. This reduced the sequence lengths further. Figure 1 contains a histogram describing the distribution of the new sequence lengths. As the figure shows, around 4000 of the sequences are of lengths 0-500, another 1500 have lengths between 500 and 1000, while the rest have lengths between 1000 and 1600.

3. Outline of Approach

The main steps performed for detecting anomalies in the flight data were as follows:

1. The sequences were clustered into groups/clusters. The sequences inside each cluster were more similar to each other than to sequences in other clusters. The similarity measure used was the *normalized longest common subsequence*. The next section presents a detailed discussion of the similarity measure.

2. A certain percentage of the sequences in each cluster were identified as atypical, for further investigation. The sequences picked were the sequences which had the lowest similarity score with the most central sequence in each cluster(the most central sequence for each cluster is identified during the clustering step described above).

3. We analyzed each atypical sequence for anomalies. New algorithms were developed as part of this stage, which performed a weighted comparison of the atypical sequence with the other sequences in the cluster, with the aim of discovering the significant differences between the other sequences in the cluster and the atypical sequence being analyzed. This was an important step of the analysis as simply classifying a certain flight as anomalous still leaves a lot of information to be processed by the analyst, to identify why this particular flight was considered anomalous. The aim of this step of the processing was to automate this task as far as possible.

We provide an informal introduction to these algorithms in the following pages. The complete details of these algorithms can be found in [8].

4. The Normalized Longest Common Subsequence Measure

The Longest Common Subsequence(LCS) is a common measure for comparing two sequences. Some common domains of application are bioinformatics, for comparing genome or protein sequences, and in computer system/network intrusion detection systems [3], for comparing user access patterns.

Given two sequences A and B, B is a subsequence of A if removing some characters from removing some characters from A will produce B. For example, suppose sequence A is given by ‘abcdef’, and sequence B is given by ‘bce’. Then removing characters a, d and f from A will produce B. Hence B is a subsequence of A.

A sequence B is described as a common subsequence of two sequences A and C, if removing some characters from both A and C will produce B. For example, if a sequence C is given by ‘gdbefce’. Then B(‘bce’) is a common subsequence of both C and A, as removing the characters at locations 1,2,4,5 from sequence C will give B, and removing characters at locations 1,4,6 from sequence A will give sequence B. The longest such subsequence between two given sequences is called the longest common subsequence.

Other measures used for comparing sequences include the ‘Match Count Polynomial Bound’(MCP), and its variants. A detailed discussion of these measures, and a comparison with LCS, can be found in [1]. However, LCS differs from these measures mainly in that the MCP family of measures does a one-to-one comparison of sequences. For example, given two sequence A=‘abce’ and B=‘bcde’, the MCP similarity would be 1, as MCP counts exact 1-1 location matches, which in the above case is only for ‘e’ in the fourth location. However, the length of the LCS in the above case is 3(‘bce’). Thus, the advantage of using the LCS measure is that it detects similarities between two sequences even if they are out of ‘phase’.

Due to the varying lengths of the flight sequences, we

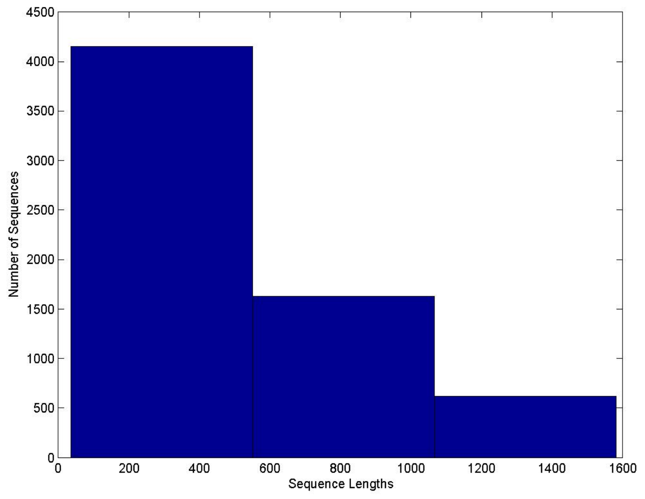

Figure 1. Histogram showing distribution of flight sequence lengths.

normalize the LCS value. We call the normalized value the normalized Longest Common Subsequence score, or nLCS. Given two sequences S_1 and S_2, the formula to calculate the nLCS is given by:

$$nLCS = \frac{length(LCS)}{\sqrt{length(S_1) \cdot length(S_2)}}$$

For example, let A = 'abcdefg'. Let B = 'fbdfeacg'. Then the LCS is given by 'bdeg', and the length of the LCS = 4. Then

$$nLCS = \frac{4}{\sqrt{6 \cdot 8}}$$

$$= 0.58$$

However, the nLCS is a difficult and computationally expensive measure to compute. This is because there is no fast method to find the LCS between two sequences. The classical approach is to use the optimal substructure property of the Longest Common Subsequence. This property is given as follows:

Let X = $< x_1, x_2, \ldots, x_m >$ and Y = $< y_1, y_2, \ldots, y_n >$ be two sequences. Let the LCS be given by Z = $< z_1, z_2, \ldots, z_k >$. Then

1. If $x_m = y_n$ $z_k = x_m = y_n$ and Z_{k-1} is the LCS of X_{m-1} and y_{n-1}.
2. If $x_m \neq y_n$ then $z_k \neq x_m$ implies that Z is an LCS of X_{m-1} and Y.
3. If $x_m \neq y_n$ then $z_k \neq y_n$ implies that Z is an LCS of X and Y_{n-1}.

This property is used to construct a dynamic programming algorithm to find the LCS. More information on this algorithm can be found in [4]. An alternate approach is the Hunt-Szymanski algorithm [5]. However, both algorithms are expensive and take a long time to compute the LCS, especially as the sequence length increases, a common reason due to which the LCS is not used as a similarity measure in practice. As part of the project, we developed a new hybrid algorithm that used ideas from both the approaches mentioned above, to calculate the LCS. Our new LCS algorithm was upto five times faster than the current algorithms. This enabled us to cluster the sequences data many times faster than current algorithms are capable of. More information on our new hybrid algorithm for calculating the LCS can be found in [8].

5. Clustering and Outlier Detection

The flight sequences were clustered using a k-medoids algorithm called CLARA (Clustering LARge Applications) [6], [7]. CLARA is a modified version of the PAM (Partitioning Around Medoids) [6] algorithm. Given a value of k for a dataset, PAM finds k clusters in the dataset. It finds the clusters by finding a representative data point for each cluster. For each cluster, the representative point, called the medoid, is the most central point in the cluster, or, the point which has the highest average similarity compared to all other points in the cluster. Given the medoids, the clusters can be found as follows: assign each data point to the medoid with which it gives the highest similarity score. Hence, to calculate the clusters, it is sufficient to calculate the medoids.

PAM finds the medoids that would maximize the quality of a clustering, that is, medoids for which the average similarity of each medoid with its respective cluster is maximum overall. However, PAM is a computationally expensive algorithm. CLARA tries to cut down on the time taken for computation by randomly picking a data subsample from the dataset, and finding the medoids for this subsample. The medoids selected for this subsample are then treated as medoids for the entire cluster. The intuition behind CLARA is that if the subsample is picked with sufficient randomness, it will mimic the original dataset in its distribution, and the medoids for the subsample shall be sufficiently close to the original dataset.

Table 1 summarizes the results of clustering on the test

Table 1. Results of clustering on the test dataset of 6400 flights. Clustering yielded one large and one medium-sized cluster(clusters 1 and 2) with high average similarity, and one small cluster with low average similarity(cluster 3).

Cluster Count	Cluster Size	Percentage of total cluster	Mean Similarity(nLCS)	Median Similarity(nLCS)
1	3301	52%	0.72	0.75
2	2253	35%	0.71	0.71
3	846	13%	0.55	0.55

dataset of 6400 flights. Three clusters were discovered in the data. The largest cluster contained around 52% of the flights, and had a high average similarity of 0.72. The second cluster contained around 35% of the flights and had a high average similarity of 0.71. The above information suggests that at least 70% of the switches are flipped in the same order in most flights, as the nLCS measures the degree to which two sequences follow the same order. This is interesting, and suggests that sequence analysis techniques are suitable for the task of comparing and discovering anomalies in flight switch data.

The third cluster was very small in comparison to the first two clusters, containing only around 13% of the total flights, and had a low overall similarity score, of 0.55. There is a strong possibility that the small cluster consists largely of anomalous flights, though more investigation into the operational significance of these clusters needs to be done to establish this.

Following the clustering step, a certain percentage of the flights from each cluster, that were farthest from the cluster centre(medoid), were classified as atypical for further investigation. The next section describes methods for detecting anomalies inside these atypical flights.

6. Detecting Anomalous Events in Atypical Flights

In the previous section, we described our approach towards finding flights that are atypical and contained events that did not follow established patterns. However, it is not sufficient to detect flights with anomalies, as this still leaves a lot of information to be analysed, to identify what the exact events were. In this section we discuss the anomaly detection algorithms we designed, that are able to identify anomalous events inside a flight sequence, thereby automating this step to a great extent. These algorithms identify any unusual event as anomalous. No operational information is currently used for this step.

The type of anomalous events we expect the algorithms to detect can be divided into three categories:

1. A sequence of switches are normally flipped at the current stage in flight, but were not flipped.
2. A sequence of switches are normally *not* flipped at the current stage in flight, but were flipped.
3. A sequence of switches were flipped in the wrong order.

The algorithms understand these events in terms of insertions and deletions. For example, if a sequence of switches is normally not flipped at the given stage of flight, but was flipped for a particular anomalous flight, the algorithms suggest that these switches should be *deleted* from the flight to make it more normal. Similarly, if some switches should have been flipped, but were not flipped, the algorithms suggest that these switches be *inserted* into the flight to make it more normal. Note that Case 3, where switches are flipped in the wrong order, is simply a combination of insertions and deletions. For example, if we have a switch sequence ABC, when the normal sequence for pressing these switches is ACB, the algorithms will suggest that the switch C be deleted from its current location, and be instered in front of switch B. Combined together, these two suggestions give us the information that the switches were pressed in the wrong order.

We developed two algorithms for anomaly detection: a) an 'insertion algorithm' that predicts desirable insertions in the atypical sequence, covering Type 1 in the three events described above, and b) a 'deletion algorithm' that predicts desirable insertions into the atypical sequence, covering Type 2 of events. Type 3 of anomalous events are covered by the insertion and deletion algorithms in tandem.

We again utilize the Longest Common Subsequence to find the desirable insertions and deletions into the atypical sequence. This is because, the common subsequence between two sequences gives us the areas of the two sequences which follow the same order. If there are regions inside a sequence that are not part of the longest common subsequence with most or all of the sequences in the cluster, it can only be because, a) they are in the wrong location compared to the rest of the flight, or b) they are in the wrong order.

These ideas are formalized by constructing an objective function to maximize for the atypical sequence, and then identifying insertions and deletions to the sequence that will maximize this function. Intuitively, the objective function is a measure of how similar the atypical sequence is to the cluster it occurs in, and one simple objective function could be the average nLCS similarity score of the atypical sequence with all the sequences in the cluster. We use a more sophisticated objective function, where we model each cluster as a Bayesian Network. The objective function, in that case, is the probability of generation of the atypical sequence from this Network. After the objective function has been defined, the next step is to identify the insertions and deletions which would maximize the objective function. That is, intuitively, we identify the insertions and deletions which, if made to the atypical sequence, would increase the similarity of the atypical sequence with the cluster it belongs to, or, in other words, make it less anomalous. These changes are identified using a greedy algorithm which, at each step, identifies the modification to the sequence that would improve the score by the greatest margin.

We now provide some examples of the type of anomalies that can be discovered by the algorithms. In these examples we compare how the algorithms react given a 'usual' flight pattern and a new sequence that does not follow that pattern. However, it must be remembered that the algorithms do *not* identify/generate any single flight from the cluster as a 'usual' flight. Instead, they make a probabilistic comparison of the atypical sequence with all the flights in the cluster. The 'usual' flight sequence, in the discussion below, is just a concept constructed to provide a more intuitive understanding of how the algorithms operate.

Example 1: Switches normally flipped at the stage of flight were not flipped.

Suppose, at a given stage of flight, the usual flight switch pattern is given as follows(here the numbers represent different switches):

101 105 102 105 103 107 106

Now, suppose we are analyzing an atypical sequence, where the flight switch pattern at the same stage is given by:

101 105 102 105 106

Here, the first four switches match identically, but switches 103 and 107 are missing from the atypical sequence.

In this case, the insertion algorithm will suggest that switches 103 and 107 be inserted after switch 105 at location 4, in the atypical sequence. Importantly, it will not suggest that switch 102 be inserted after switch 105, even though 102 also occurs after switch number 105, right after location 2. It is able to avoid that confusion because it established a longest common subsequence between the two sequences, which matches the 105s at location 2 in the two sequences.

Example 2: Switches normally not flipped at the stage of flight were flipped. Suppose we have the following switch sequence pattern as usual at a given stage:

101 105 102 105 106

And the atypical sequence follows the following pattern:

101 105 102 105 107 106

In this case, we have a 1-1 correspondence between the first four switches in both sequences, and between the switch at location 5 in the first sequence, and the switch at location 6 in the second sequence. However, switch number 107 does not match. In this case, the deletion algorithm will suggest we delete the switch 107 from the atypical(second) sequence.

Example 3: Switches were flipped in the wrong order.

Let the usual pattern for switches be

101 103 105 * 107

Let the atypical sequence be

101 103 107 * 105

Here the asterix(*) represents wildcards. That is, there may be switches that are usually pressed between 105 and 107, but the fact that 107 is pressed after 105 remains constant. In this case, as the atypical sequence has switch 105 pressed after switch 107, the deletion algorithm will suggest that switch 105 be removed from its location in the atypical flight sequence. The insertion algorithm will suggest that switch 105 be inserted before switch 107. Combining these recommendations, we are able to deduce that the switches 107 and 105 were pressed in the wrong order.

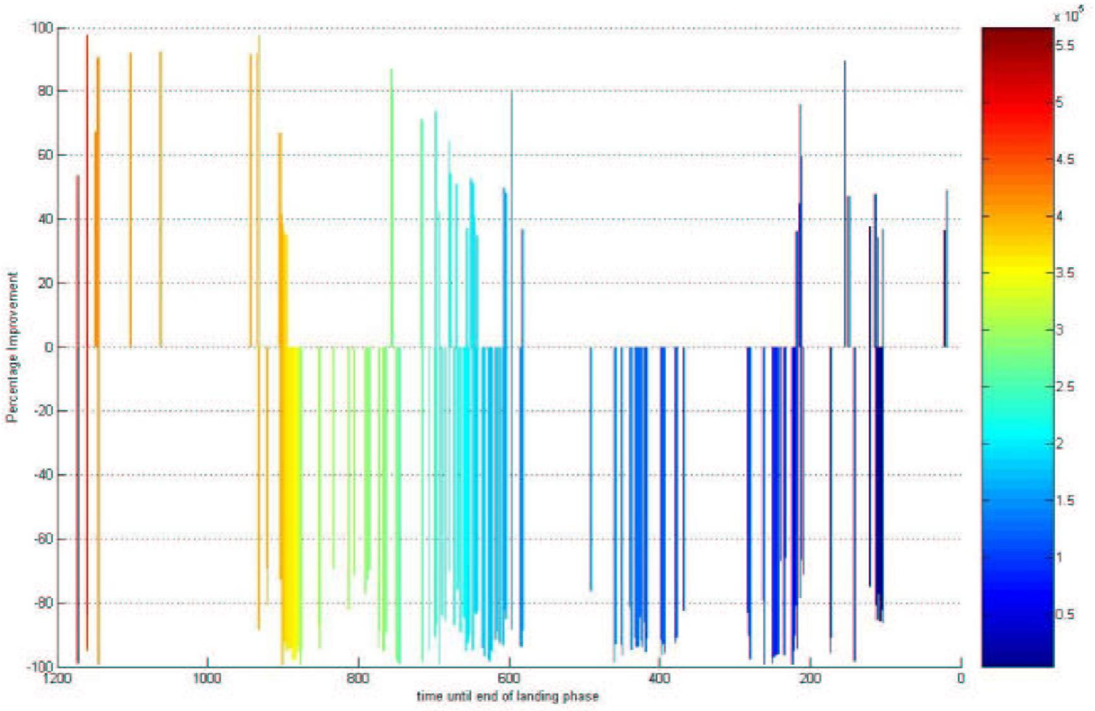

Figure 2. Sample graph for landing stage of a flight.The x-axis represents the time till touchdown. The bars along the positive y-axis represent the switches usually pressed at that stage of flight, but not pressed in this flight. The bars along the negative y-axis represents the switches usually not pressed at that stage, but pressed for this flight. The height of the bars gives the algorithm's confidence in its prediction. The color represents the altitude.

7. Description of Algorithm Output

The anomaly detection algorithms present the following output:

A graph showing the anomalous areas inside an outlier sequence.

The graph provides a simple visual interface that will allow the analyst to focus his interest on the areas which seem most suspicious. For detailed information on these anomalies he/she can access at a table generated by the algorithms in parallel.

Figure 2 presents the graph output for a sample atypical flight taken from the data. The horizontal axis represents the time remaining till touchdown. The positive direction of the vertical axis represents the desirable insertions, that is, switches that are usually pressed at that stage of the flight, but were not pressed for this particular flight. The negative direction represents the desirable deletions, that is, the switches that are usually not pressed at that stage of the flight, but were pressed on this flight. The height of the columns indicates the confidence of the prediction. The confidence is calculated as proportional to the improvement in the score of the objective function defined for the atypical sequence, if that particular addition/deletion was made to the sequence. A graph with no bars would mean that the algorithm cannot suggest any desirable insertion/deletions with non-zero confidence, and would represent a completely normal flight. The color of the columns is representative of the altitude. The colors change from darker to lighter, as the altitude decreases (for the graph in Figure 2, the altitude information was synthetically generated, and does not represent the actual altitude information for the given flight).

Besides the test runs over real data, a number of test simulations were run over synthetically generated and modified data, to test the effectiveness of the algorithms under various situation. Figure 3 show the resulting graph from one of the test simulations run over synthetic data. As part of this simulation, a sample dataset of 5000 sequences, each with a length of 1000 symbols,

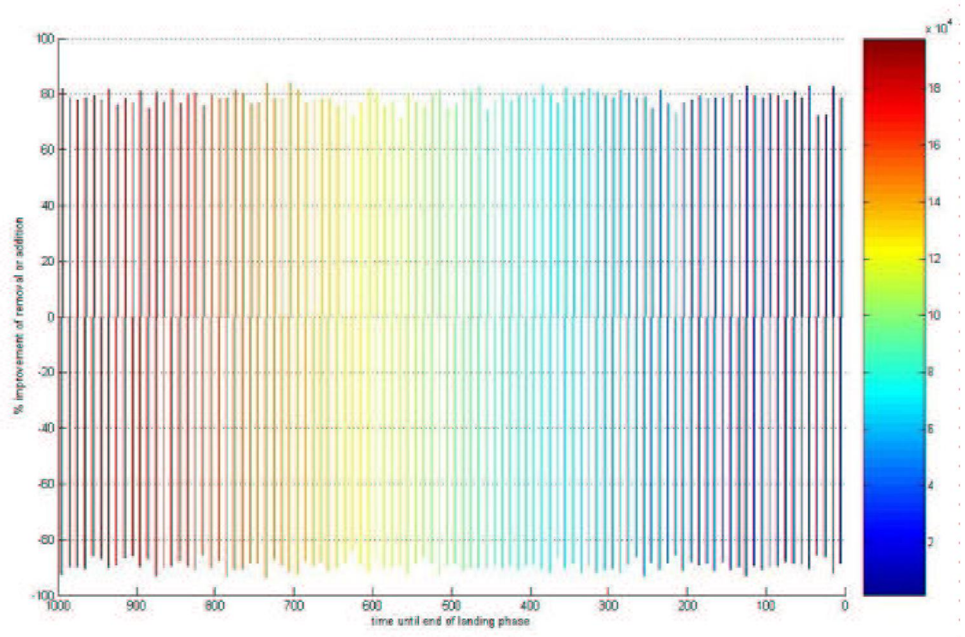

Figure 3. A graph showing test results over a syntetically generated dataset of sequences. The x-axis and y-axis are the same as in Figure 2. A test dataset of similar sequences was first synthetically generated. A sequence from the dataset was randomly picked, and every tenth symbol in the sequence, starting with the symbol at location 5, was swapped with the immediately following symbol. The algorithms were then run over the modified sequence, to test if they could catch these modifications. As the graph shows, the algorithms suggest a deletion, followed by an insertion, at every swapped pair location. That is, the algorithms could successfully identify all the swap locations.

and with the same number of unique symbols as the original flights dataset, was generated. These sequences were generated so, that they demonstrated an approximate similarity of 75% to each other.

Following this, a subsample of the sequences was picked at random from the dataset, and modified in the following fashion: for each sequence, starting with the symbol at location 5, every tenth symbol was exchanged with the one immediately following it. For example, symbols at locations 5 and 6 were swapped, also symbols at location 15 and 16, 25 and 26, and so on. The algorithms described in Section 6 were then run on the dataset. The aim of the run was to check whether the algorithms could detect these swaps or not, as, compared to the rest of the sequences in the dataset

Figure 3 shows the graph generated by a test run over one such sequence. As can be seen from the graph, the algorithms suggest a deletion at every tenth location, starting with location 5, followed by an insertion at the immediately following location. Combined together, these two actions represent the swap originally made during the test modifications. In other words, the algorithm successfully discovered all of the modifications made to this sequence, without a single false alarm.

A report on the anomalous areas. A report is generated, giving detailed information about each anomalous event in the sequence. The report tells the analyst about the switches anomalously pressed/not pressed during the flight, along with the confidence.

8. Conclusions

This paper shows a novel algorithm to detect anomalies in discrete sequences that record the switch positions in the cockpits of commercial airliners. The results so far are promising in that they indicate that we are able to identify anomalies in very large data sets of aircraft sequence data. Moreover, we are able to indicate time steps at which certain switches should have been either depressed or left unchanged.

The algorithm is fast and scalable and has several advantages over standard methods, because it takes advantage of the sequential nature of the data. We plan to apply these techniques to general ISHM problems for a wide variety of aerospace platforms.

In the next stage of the project, we also plan to build more sophisticated models of the data sequences, such as by grouping common sets of contiguous symbol sequences under a single 'super-symbol', and by training a Hidden Markov Model over these 'super-symbol' groups. We also plan to reduce the number of false alarms raised by the anomaly detection algorithms. These false alarms occur because all detected anomalies are not equally important. For example, an anomaly involving a pilot talking to the control tower at an occasion different from usual, is not as important as an anomaly where some significant step of the landing phase was not executed. This can be taken into account by building probabilistic models of the sequence 'gaps', that is areas within sequences that do not form a part of the longest common subsequences. These probabilistic models could be simple maximum likelihood estimate models, or more sophsiticated models, such as modeling the gap as a Hidden Markov Model, or as an even following a Poisson distribution.

REFERENCES

[1] T. Lane, "Machine learning techniques for the computer security domain of anomaly detection", Ph.D. Thesis, CERIAS TR 2000-12, Purdue University, August 2000.

[2] B. Amidan, and T. Ferryman, "Atypical Event and Typical Pattern Detection within Complex Systems", IEEE Aerospace Conference, 2005.

[3] K. Sequeira and M. Zaki, "ADMIT: Anomaly based Data Mining for Intrusions", Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(SIGKDD), 2002.

[4] T. Cormen, C. Leiserson, R. Rivest and C. Stein, "Introduction to algorithms", The MIT Press; 2nd edition.

[5] James W. Hunt and Thomas G. Szymanski, "A Fast Algorithm for computing Longest Common Subsequences", Communications of the ACM, Volume 20, Issue 5 (May 1977), Pages: 350 - 353.

[6] L. Kaufman and P.J. Rousseeuw, "Finding Groups in Data: An Introduction to Cluster Analysis", John Wiley and Sons, Inc., New York (1990).

[7] R. T. Ng and Jiawei Han, "CLARANS: a method for clustering objects for spatial data mining", IEEE Transactions on Knowledge and Data Engineering, Volume 14, Issue 5 (Sep/Oct 2002), Pages: 1003 - 1016.

[8] S. Budalakoti, A. N. Srivastava, R. Akella, E. Turkov, "Anomaly Detection in Large Sets of High-Dimensional Symbol Sequences", Submitted for evaluation, 2005.