2017

Markov Chain analysis of packet sequence for intrusion detection

Chad Bockholt
Iowa State University

Markov Chain analysis of packet sequence for intrusion detection

by

Chad Taylor Bockholt

A thesis submitted to the graduate faculty

in partial fulfilment of the requirements for the degree of

MASTER OF SCIENCE

Co-majors: Computer Engineering (Secure and Reliable Computing) and Information Assurance

Program of Study Committee:
George Amariucai, Co-major Professor
Doug Jacobson, Co-major Professor
Umesh Vaidya

Iowa State University

Ames, Iowa

2017

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

Intrusion Detection is a broad and complex field in cybersecurity. There are varieties of existing methods with varying degrees of success, which attempt to classify various types of traffic as benign, or attacking. A tool that can do this consistently and reliably, and with minimal overhead is ideal, benefiting with respect to analysis overhead, as well as level of information privilege. This paper attempts to provide such a tool through packet sequence analysis.

Packet sequence, as referred to in this paper, is the order and number of the exchange of packets. Sequential probability ratio test (SPRT) analysis is done on the sequence history of each pair of IP addresses in attempt to determine if the flow can be classified as an attack based solely on this. SPRT is performed for single class, two class, and with more specialized attack classes.

Through manipulation of a large variety of parameters and analysis of results indicated that packet sequence can, under the right circumstances provide an indication of an attack. While this is true most of the attacks seen in the data tested, there is a high level of parameter tuning process involved. While likely not all attacks will be identifiable by this method, for those attacks which do not appear readily and obviously useful, there are several which show promise with different configurations of parameters, and could potentially be useful with a higher degree of tuning.

CHAPTER 1. INTRODUCTION

The incentive for an effective IDS with very little overhead or data needed exists in a variety of applications. Oftentimes an ideal IDS is configured as an IPS, that is Intrusion Prevention System. In this case the system needs to be in-line like a firewall, perform the analysis on live data and traffic packets must meet certain criteria in order to pass through the firewall or it will be dropped. Oftentimes even a basic firewall gets overwhelmed in this configuration, especially under large bandwidth situations. Under these circumstances a device might go into failover mode, which either slows network traffic to a crawl, or allows it to continue un-analyzed, bypassing the protection measures [1]. For this reason, in a situation like this it is important to keep the processing overhead to a minimum, and avoid creating a network speed bottleneck, while still maintaining an effective level of detection.

Sometimes other factors eliminate the possibility for deeper network inspection aside from the large volume or processing required, with this as the case, the incentive for such a system is in the need for very little data, and no identifiable information from packets. With encryption as common and useful as it is today, it often allows cyber criminals to take advantage of this by attacking over an encrypted channel such as SSL or SSH [2] [3]. Attackers can for example disguise traffic of planted malware over encrypted channels and remain indistinguishable from legitimate traffic. Even when the payload is encrypted, addresses are still available, being required by routers and network systems for delivery. This information can be used without needing to access the more complex information potentially held.

As the proposed analysis likely requires training on both legitimate, and malicious data, ISPs are potentially a massive source of information regarding both types of network traffic. However there are also many scenarios where data sharing is prohibited by legal consequence, or some other disincentive. The need for a proposed system is high in cases such as this as well, needing virtually none of this protected data. For example HIPPA regulations or an organizations reluctant to allow other access to customer or otherwise sensitive data. In these situations the proposed scheme could still benefit as an intrusion detection system where many or most others might not be able to. A step further, in stripping even more identifying data and obfuscating the IP addresses involved, but leaving the flow intact will still allow the algorithm to function and user privacy may still be preserved.

The motivation and inspiration for this project comes from attacks where an out-of-order, or abnormal pattern of packets, is used to disrupt or attack systems. Noticing and identifying this behavior could be key to minimalistic-ally identifying an attack, especially if it is unique to a certain attack method. In contrast, even classifying attacks as abnormal or unlikely given a set of background data would prove useful to intrusion detection.

CHAPTER 2. RELATED WORK AND OTHER METHODS

Intrusion detection as a field is a large area of security, cyber and otherwise. This paper aims to implement a network based IDS, analyzing network data with the purpose of looking for attacks and anomalies. Network intrusion detection is not a new field, some of its origins seen in 1987 and frequently added to since [4]. But there are ever evolving techniques and strategies, we see everything from open source systems like Snort and Bro, with community developed customizable tools, to highly commercialized systems like FireEye with sophisticated analysis techniques [5] [6] [7]. A common feature of these systems is gathering statistics about the traffic, or subsets of the traffic and examining with various data analysis methods these features often involve potentially protected data, or [8]. Applications of lightweight protocols that use only a subset of data such as SSHCure have been attempted before on network traffic to detect ongoing attacks [3]. A lightweight method such as SSHCure gives promise that the minimal method is possible.

Markov Chains are certainly not new to the broad field of anomaly detection, having been used to examine standard and abnormal video conditions, and are not new to the field of security [9]. Applications of Markov Chains, and Hidden Markov Models, in network analysis with the goal of intrusion and anomaly detection have been attempted, and successful previously, though other methods generally examine phases of an attack as states of a Hidden Markov Model rather than network data points [10] [11]. Columbia University describes the problem of information sharing, with large organizations and internet service providers being reluctant to share data that may prove useful to collaborate in intrusion detection. They do this by attempting to anonymize user data and preventing Markov Chain timing attacks from taking place as an attempt to correlate user identities [12]. Additional advancements in this regard are important to ensure that anonymous sharing of data, along with a continued support of user privacy are essential to computer security. This paper aims to show another method which will allow data sharing for attack and anomaly detection, and still preserve user privacy.

In attempt to reduce the impact of the network bottleneck problem that many intrusion prevention systems face, some turn to high speed, specialized hardware. This hardware acceleration requires a very static algorithm however, and the constantly evolving methods of attackers prevent use of this for every part of the algorithm, though it can still be beneficial. This

type of device may also be useful in the proposed system, though reduction of data required could make it more effective, or even less necessary. While some of these devices have very impressive statistics, they provide little headway to the issue of user anonymity. This paper aims to build on this theory of non-disruption through exploration of data-lightweight analysis algorithms.

CHAPTER 3. EXPERIMENT CONCEPTS

The algorithm this paper intends to build for analysis relies heavily on several mathematical concepts. An introduction to each, along with a general context is included with basic examples. Each is described as it relates to the other concepts relevant to the experiment performed. A discussion of how each will be applied to the context of networks appears in Chapter 4 below.

Markov Chains

A Markov Chain is a mathematical way of representing the states in a system. This system could be the anything from the current base a runner is on to as complex as the state of a computer register [13]. What a Markov Chain does is attempt to represent the probability of a transition from one state to another possible state. From a simple example, the state of a coin flip, it is obvious that a coin can flip either a heads or tails, and at each stage, the probability of transitioning to either heads or tails on the next flip is each 0.5. This can be seen in the diagram below, with the states as circles, and the arrows and numbers representing probability transitions between.

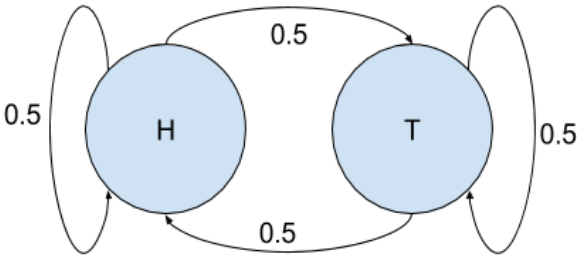

Figure 1 Markov Chain Example Diagram 1, Coin Flip

Though this is a very simple system, more complex cases can be captured with a similar diagram. Note that for each state, the probabilities of transitions exiting the state must add up to 1, including a possibility of remaining in the same state. That is some transition must occur, even a transition which maintains the current state. A more involved example representing the hypothetical states of weather: sunny, rainy, and windy is included below. Note the similar properties to above.

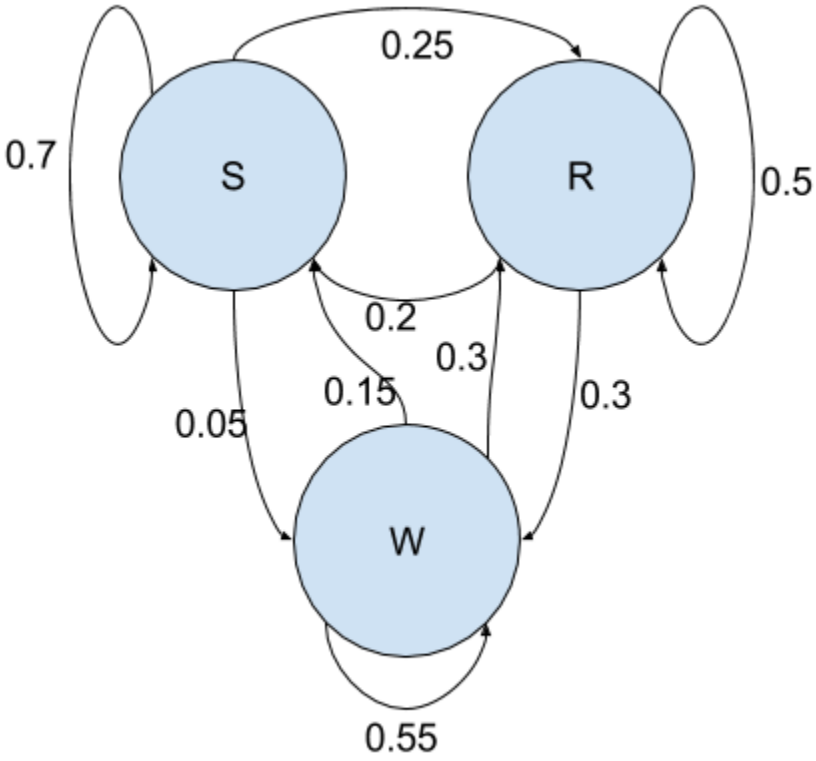

Figure 2 Markov Chain Example Diagram 2, Weather Prediction

This same Markov Chain information can be captured in a matrix as well, and is oftentimes more useful. The current states are listed vertically, and the next potential states are listed horizontally. Note that like the sum of transition probabilities, represented above as arrows must be equal to 1, with a matrix each row must do the same. Both examples previously demonstrated as chains are converted to matrixes and included below.

				Sunny	Rainy	Windy
	H	T	Sunny	0.7	0.25	.05
H	.5	.5	Rainy	0.2	0.5	0.3
T	.5	.5	Windy	0.15	0.3	0.55

Figure 3 Markov Chain Example Matrices

In practice the transition values of a Markov Chain are often not readily known, in these cases a sample of observations is taken in order to estimate the states. The current state is observed, allowed to proceed to the next state, and the transition recorded as a sample. This process is repeated many times until a sufficient sample is taken and the chain is trained. Note that while we can observe it, Markov Chains as represented do not take into consideration any state history. A standard Markov Chain captures is the transition between two states, meaning that there is no allowed dependency on the states leading up to the one represented. In a sufficiently sampled chain, transition probability will be accurate considering all possible histories. In practice there may indeed be other dependencies, which are explored through order, and a pseudo-time dependency later. It may be possible to calculate a better certainty for transition probability if the history leading up to is also considered, taking into account the previous n states encountered.

This is where the concept of a Markov Chain with history is introduced. In the case of a true random variable, such as the coin flip, introducing any history will have no effect, given enough sampling to sufficiently train the Markov chain. This can be seen with the order 2 example included below.

	T	H
H, H	.5	.5
H, T	.5	.5
T, H	.5	.5
T, T	.5	.5

Figure 4 Markov Chain Example Diagram 2nd Order

There are other cases however, where a history may prove useful, such as the weather prediction example. Had history not been considered, the first three states shown, and to be precise: six more that are not, would all be included in one row for a single order chain. This allows better predictions to be made than when considering only a first order chain, showing a dependency on the previous transitions, or the state history.

	S	R	W
S, S, S	0.65	0.15	0.2
R, S, S	0.1	0.05	0.85
W, S, S	0.55	0.25	0.2
…			
W, R, W	0.2	0.6	0.2
W, W, W	0.3	0.5	0.2

Figure 5 Markov Chain Example Diagram 3rd Order

Though increasing the order drastically may seem like an intuitive way to increase the precision when predicting state transitions, it is not without drawbacks. As can be seen by the matrix above, the number of rows is increased from just 3 with a first order to 18 with a third order. Or generally with an n-order, and the k size alphabet of possible states, increasing n by one will increase the number of rows necessary by a factor of k. In an n order Markov Chain, k^n rows are necessary to maintain all possible states. Along with this increase in space this increase comes a reduction in the amount of training done on each row. On average, for every row ending with a particular state, an n order increase will lead to the number of samples used for the training will be reduced by a factor of k^n. Thus a large number of additional samples may be necessary to sufficiently train a matrix to the same degree of confidence.

Once a Markov Chain is trained regardless of order, it is possible to observe a transition and determine the probability of the same transition occurring within a given matrix. Using this method an indication can be determined of how similar the circumstances in which the matrix was trained are to the circumstances under which the transition was observed.

SPRT Algorithm and Log Likelihood ratio

The Log Likelihood ratio is a method of measuring which of two classes an object is more likely to be a member of [14]. Given the probability of a sampled transition belonging to each of the classes, we assign one hypothesis class as the positive direction, and the other as the negative. These hypothesis classes will come from another mechanic, as it relates to the scope of his paper, this sample will come from a trained Markov Chain as described above. The log of the ratio is

then taken, which by logarithm algebraic rules can be written two equivalent ways in Formula 1 below. The log will be positive if the ratio is greater than one, meaning the positive hypothesis is more likely and a negative log indicates the negative hypothesis is more likely.

$$log\ \Lambda(x) = Log\left(\frac{P_1(y|x)}{P_2(y|x)}\right) = Log(P_1(y|x)) - Log(P_2(y|x))$$

Formula 1

Sequential Probability Ratio Test (SPRT) is a technique used for sequences in the classification problem. In this context SPRT is an extension of the likelihood ratio seen above that can be used when events occur in sequence, and are observed a common set of circumstances. For example, it can be used as a sampling algorithm for quality control, samples are taken from a set of product, and through SPRT, the batch can be determined to be either good or bad, the positive or negative hypothesis [15] [16]. Formally, SPRT is the sum of the log likelihood ratio, or the log of the ratio of membership probabilities. S_i below represents the score of the i^{th} state, and $\log \Lambda(x)$ is from Formula 1 above.

$$S_i = S_{i-1} + \log \Lambda(x)$$

Formula 2

This SPRT function can be visualized graphically, by plotting for value with respect to the number in the sequence. This lets the algorithm show how confident the sequence is to belong to either the positive or negative hypothesis. Ideally these graphs will clearly diverge and allow for immediate and obvious classification, an example of a graph that demonstrates this is included below. For the purposes of this paper, a positive event is when a flow would be classified as attack positive. That is, a positive indicates there is an attack present in the flow's SPRT chart.

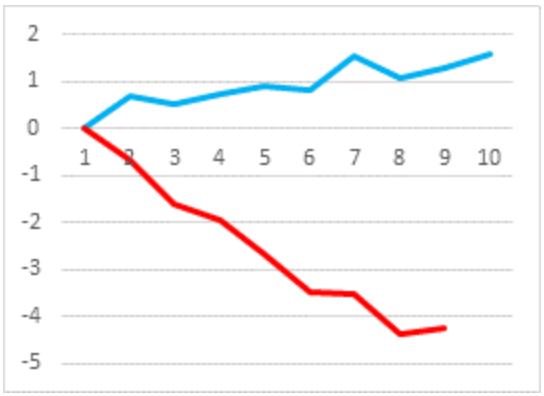

Figure 6 SPRT Two Class Expected

In practice these graphs may not be well grouped, and may not diverge as quickly and cleanly as hoped, this means that false positives or negatives are likely. However because of the charted nature it is easy to define a threshold scheme to select classes. A simple linear fit line can be used, though more complex methods are clearly possible. A simple example is included below for reference.

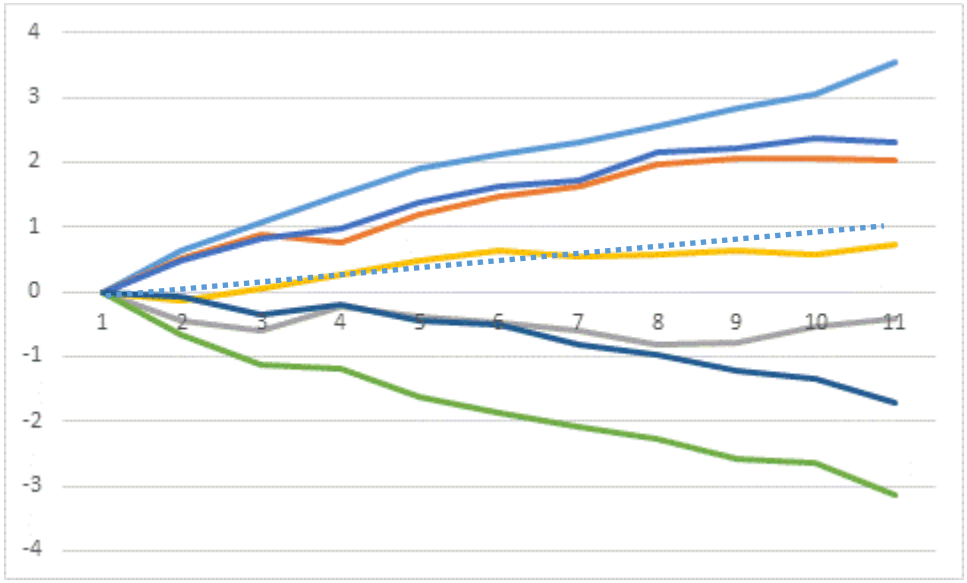

Figure 7 SPRT Threshold Example

With the solid lines as various SPRT series, and the dotted line as the defined linear threshold, we see that most of these series are easily classified as above or below the threshold, and this is the ideal case. It is very possible however to have a sequence which crosses threshold multiple

times as demonstrated by the yellow line above. With only one threshold line each sequence would be determined upon the first observation, this is not what we want. Sone alternatives are to pay attention to the number and severity of the times a sequence crosses, a difference integration of sorts. Another simpler is to have two threshold lines with an ‘unknown zone’, and making a decision based upon this.

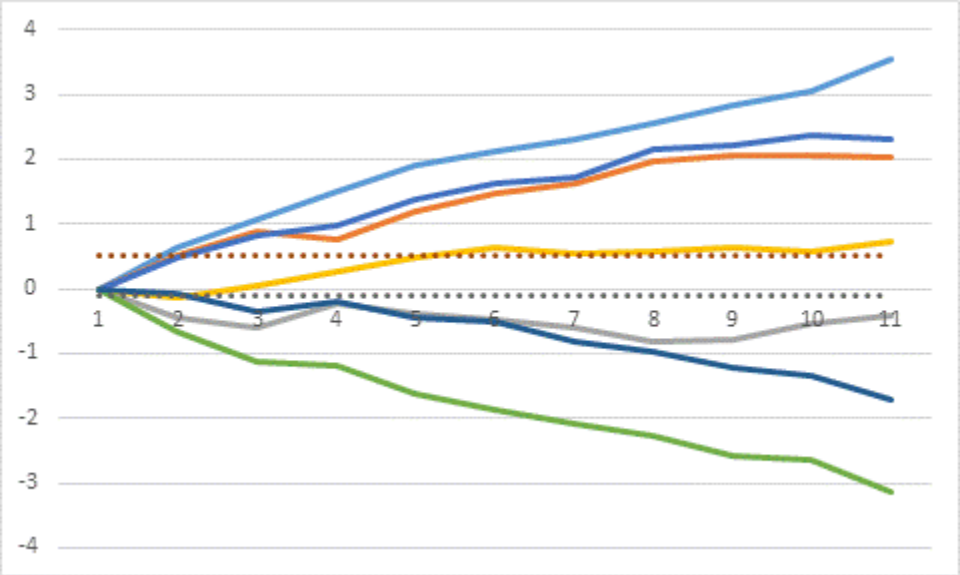

Figure 8 SPRT Threshold Example 2

With the two horizontal lines we more easily see an unknown zone where blue and yellow originate, then are soon after classified. It is also possible however, that an attacker may purposefully or be forced to exhibit normal user behavior for a period of time. This could be an unintentional part of attacking, perhaps involved in establishing a channel, or perhaps intentionally attempting to beat the classification system. In this case the flow of an attacker will still have large periods of decrease as the SPRT is charted. An example of what this may look like is included below. A good threshold idea for the classes should be able to detect and account for this behavior, and still classify as an attack. This could be seen, and accounted for by a sloped threshold, requiring that a flow behave well throughout the entirety of the attack in order to continue to be classified as legitimate.

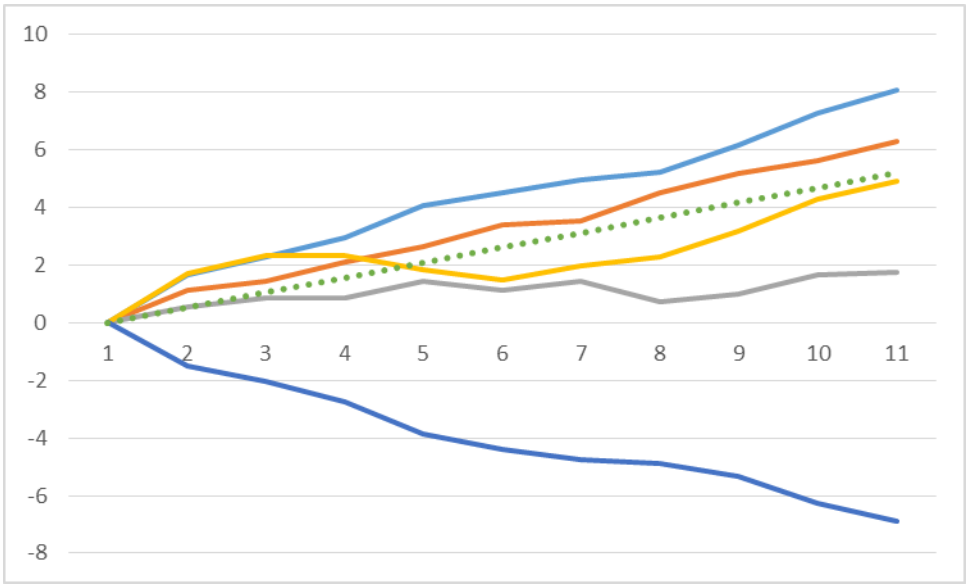

Figure 9SPRT Threshold Example 3

It can easily be imagined that a significant variety of possible schemes for which to classify the SPRT flows. Specific parameters will depend on the nature of the scenario, as well as the desired false positive and negative rates.

SPRT Variants

In some cases it may be beneficial to adjust the SPRT algorithm in attempt to more quickly or more accurately classify a sequence. The adjustment this paper examines is assigning a weight to the training of a particular scenario. The classes to be examined in this analysis are for membership of an estimated Markov Chain. Because these chains are estimated from training, a confidence multiplier can be assigned corresponding to how well the distribution for a transition from a given state has been trained. The general theory is that state transitions which have seen more, and thus are better trained, will be given a higher weight and allowed to affect the SPRT chart more. Because the value is a ratio, this confidence multiplier should be assigned corresponding to the more minimally trained Markov Chain. This modifies Formula 2 as seen above as follows. With the number of times the row 'x' in a matrix is trained, as $P(x)_{i\text{-}train}$, and 'trained' is an integer value where the matrix is then considered trained, and additional data points should not increase the confidence For the purposes of this experiment the trained value is set to consider 1000 packets sufficiently trained.

$$confidence: \ C = \min\left(\frac{\min(P(x)_{1-train}, P(x)_{2-train})}{trained}, 1\right)$$

Formula 3

$$S_i = S_{i-1} + \mathrm{C} * \log \Lambda(x)$$

Formula 4

SPRT can be extended to consider cases where there is only one, and where there are n classes. For one class extension, the algorithm only needs to track the numerator of the LLR. Note that in this case, because a logarithm is negative on the (0-1) interval, the SPRT graph will be always decreasing. It would expect that a member of the class will have high probability and remain close to the horizontal axis, while a non-member will have low probability and decrease greatly. Threshold schemes for this scenario can be applied similarly to those examined above.

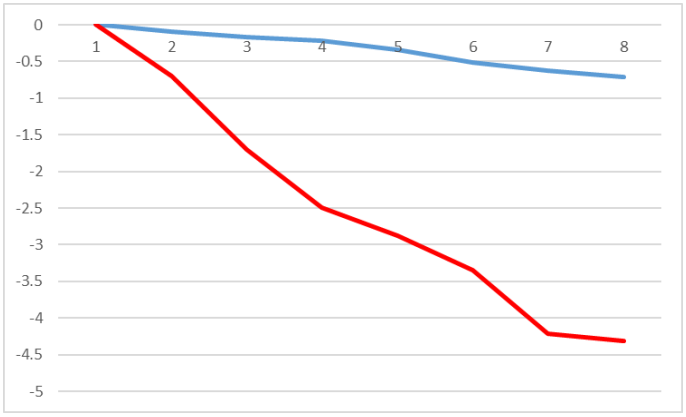

Figure 10 SPRT One Class Expected

If SPRT is extended to more than two classes, the easiest solution is to perform one class analysis on each, and determine which class fits most optimally. Alternatively, and perhaps more effectively each can be compared to a reference class, consisting of background 'normal' noise as this paper later demonstrates. This requires scaling in the number of tests, and class training required to increase by a factor of the number of classes. In a cybersecurity context a class for each attack type would grow extremely quickly with the large variety of attacks which already exist, increasing the necessary computation of the system proposed.

CHAPTER 4. NETWORK APPLICATION OF SPRT

In this paper, an attempt is made to classify network traffic through flow sequence patterns. Chapter 3 above discussed the mathematical concepts used in general to facilitate this process. This chapter is a description of how these tools relate to the network classification scenario, as well as how the algorithm of the experiment is performed.

Markov Chain Formation

For the scope of this paper a Markov state involves the sequence of packets between two hosts, we will refer to this sequence of events and the IPs associated a Flow. Regardless of the protocol, an exchange of packets takes place and the sequence contained is observable. The sequence can be gathered through sniffing, a custom network tap, pcap files, or a variety of other sources. Even with a variety of sources possible, in applications where information is sensitive, the appropriate transitions can be extracted and used to train a matrix. Sharing this matrix will not violate privacy of any sort, as once trained, each transition is indistinguishable from others and any context outside of any transition history included. Pseudocode of this transition extraction process is included in this sub section.

When observing a flow sequence from any packet source, we use the following algorithm. Note that a flow, will have defining keys corresponding to the pair of IP addresses associated. A packet will as well, and thus the corresponding flow can be reference from a packet.

For each packet encountered:
    If the flow corresponding to the packet exists in the dictionary:
        Check the direction, if it is forwards:
            Increase a counter kept for each flow, to reflect the number of packets seen in that direction
        If it is backwards:
            Append the current counter value to a list representing the history
            Reset the counter back to one
    If there is no flow that corresponds to the packet:
        Initialize the flow in the dictionary
        Initialize the counter of the flow to 0

Figure 11 Pseudo Code For Flow Extraction

This algorithm builds the flows that the defined states of the Markov chain can then come from. In practice we need to put a practical limit on the maximum number of packets recorded in one direction in a row, or the number of states in the Markov Chain will scale to be infinite in size. This value is set to 10 for the experiment and results discussed following. These flows have a history of the following form, where a, b … are positive integers between zero and the practical limit.

[a, b, c, d] -> e

When examining a packet for the Markov chain sample the n order history must be considered, in order to attempt to take into account the beginning and ending of a sequence, leading and trailing zeros are added to the sequence, so for a 3 order Markov Chain, this is modified as follows.

[0, 0, 0, a, b, c, d, e, 0]

This allows the algorithm to use an n-wide sliding window approach to retrieve the state transitions. Continuing with the current example, the state transitions this flow gives us for an order 3 history are:

[0, 0, 0] -> a; [0, 0, a] -> b; [0, a, b] -> c; [a, b, c] -> d; [b, c, d] -> e; [c, d, e] -> 0

The algorithm then can use these transitions to train a matrix with the x -> y pairs by increasing counts in the corresponding position of the matrix. A sufficient amount of this data from benign traffic will train a matrix, and allow the algorithm to test the probability of membership with the SPRT algorithm. This test is performed whenever a transition is observed in the flow, the probability of the current counter value is measured given the corresponding flow history in the matrix. For the example flow above, a subset of the transitions would be determined as follows.

$$P_1(a|[0,0,0]), \quad P_1(c|[0,a,b]), \quad P_1(0|[c,d,e])$$

For visualization purposes, the process would be to take the given Markov Matrix: P_1, find the given row: [a, b, c], and locate the appropriate column: e. The probability at this location is what is used in LLR and SPRT as described below.

LLR and SPRT

LLR and SPRT can be performed according to the several variants above. Given the trained matrices, SPRT can easily be plotted relative to one or two classes, and results examined for a threshold possibility. One Class, Two Class and Multi Class are all performed in the scope of this paper. For the purposes of this paper, in the multi-class case of this experiment, attack types are known and in order to save computational time, SPRT is only performed for each attack flow relative to that corresponding attack matrix. However in practice this is not possible, as an unknown flow cannot be evaluated against a single known attack if the attack is unknown. This will lead to a much more complex analysis process, scaled by a factor of the number of attack matrixes used.

Pseudo Time Dependency

A common behavior of attacks is to attempt to ignore protocol when establishing a channel, this might be done by sending out of order packets in attempt to exploit a vulnerable machine like the TCP split handshake attack, or sending a large number of packets initially to try to overwhelm a host like they SYN Flood attack [17] [18]. If this behavior occurs, it is likely to happen in the first several packet exchanges of a flow. In order to more fully account for this behavior, an additional dimension of the Markov Chain is taken into account, a pseudo timing of sorts.

The number of the sequence in the flow will be the extra dimension that is also used to train, and to test the probability of membership to a class. The idea being that the initial sequence is not drowned in the Markov Chain by the volume of packets that are not part of this initial sequence. If a distinction is noticeable in an attack very early in the flow, this method should make the divergence more obvious, and the difference apparent early in the SPRT Graph.

In practice the first 10 steps of the flow sequence are recorded in a corresponding matrix and all remaining are recorded in the same steady state matrix. So the transitions seen in a flow to be tested are evaluated against the same sequence number in the reference matrixes.

DARPA Dataset

MIT and DARPA released datasets for the evaluation of intrusion detection systems, it consists of 7 weeks of training data, with a number of labeled attacks, and background data, as well as 2 weeks of testing data in the same format, though consisting of a higher portion of attack data [19]. For this project a subset of the 1998 set is used for testing and training. Network traffic files were separated into normal or background data, and the distinct labeled attacks. These attacks are used for training and testing both as an all encompassing group, and individually according the different class scenarios described above.

Appendix A contains data counts, and histograms for the Markov Matrix associated with the background, and attack data sets used in training of the following experiments. Note that for the attack data, Appendix A only contains information about the jointly trained attack chain used in the Two Class case. For the Multi Class and individual attack chains, see the corresponding appendix containing more information about the attack type.

There were 25 distinct attack types sampled from the DARPA data. All were used to train the attack Markov Chain in the Two Class case described and evaluated in Chapter 5.

CHAPTER 5. RESULTS

Because of the large number of attack types included in the DARPA 1998 Data, and a large number of variables examined, this section has selected and focuses mainly on 4 interesting attacks. Portscan, Multihop, Satan, and Warez, as labeled by DARPA. For completed results of these attacks see Appendices B – E including descriptions, histograms, and SPRT graphs of results. Relevant graphs are also included for discussion within this chapter when relevant. SPRT Charts are generated by plotting different parameter scenarios of attack data in red, against the same subset of background data, evaluated with the same parameters as the attack data, in blue.

Effect of Order

Intuitively, if a chain is able to be sufficiently trained, increasing the order will only increase the accuracy of predictions, as described above in section two. For example if an attacker makes an unlikely move within a low order chain, it will negatively affect the probability, and SPRT at few steps. But the same unlikely move will be present for the n steps, and impact the ratio for all n. For demonstration, below the order 1, 2, and 3 Two Class SPRT graphs for the Warez attack are included as a reference.

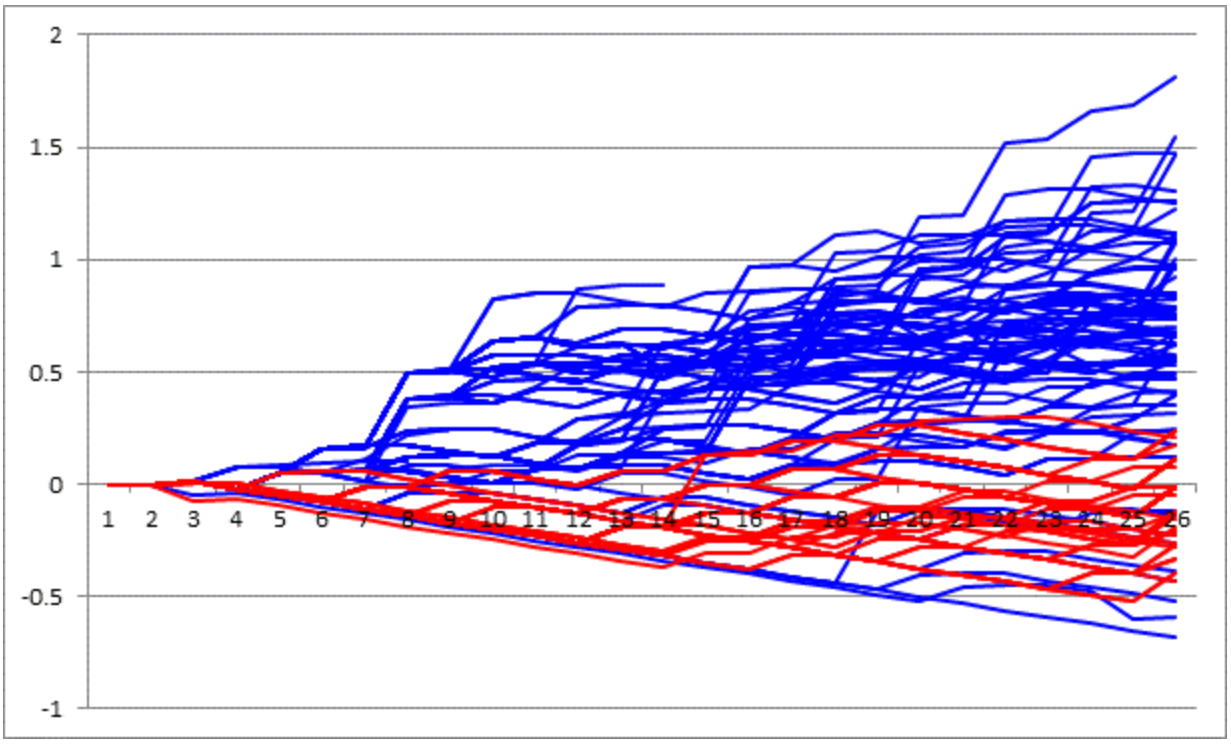

Figure 12 Warez, Order 1

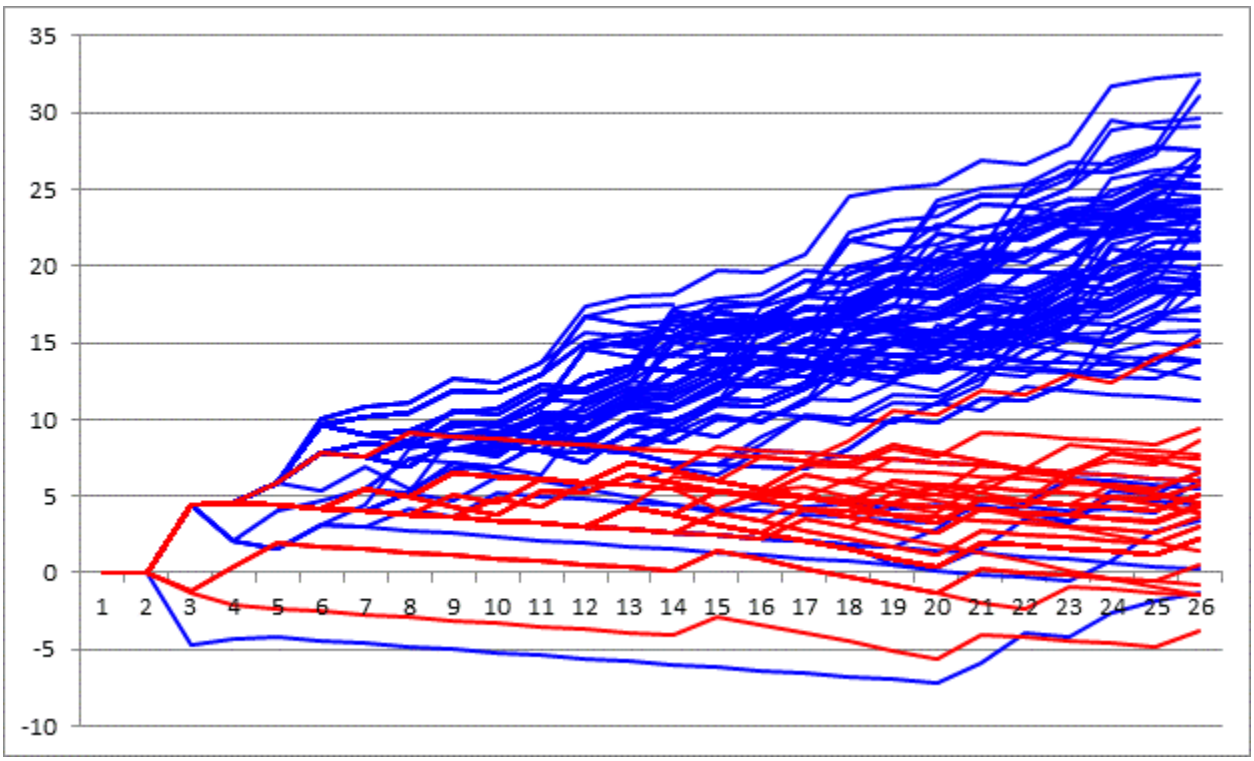

Figure 13 Warez, Order 2

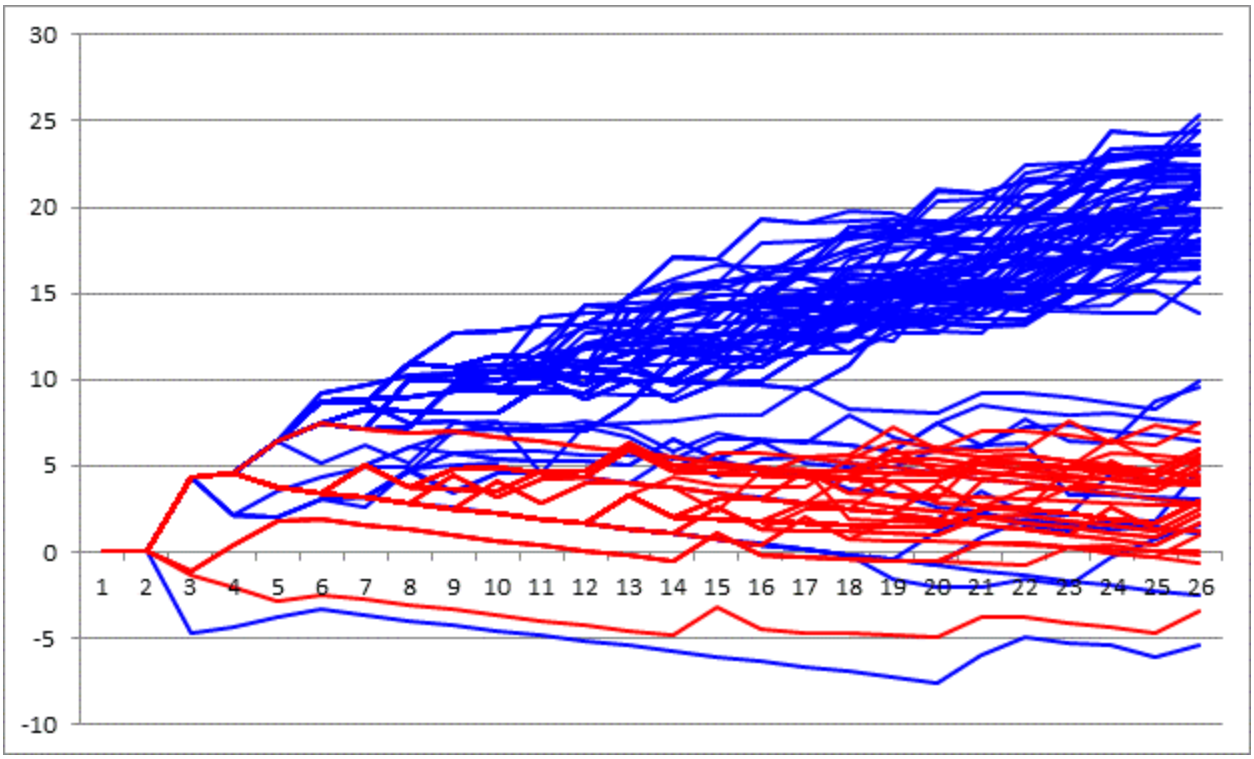

Figure 14 Warez, Order 3

As seen by the above charts, order has the effect of narrowing the grouping of both the attack and the background data, with the exception of outliers. This is a very useful effect, as it would allow for tighter thresholds and ideally fewer false positives and negatives. Though the graphs in general look promising, it is important to note the presence of several good reference flows still overlaid, and some even lower with the general grouping of attack flows, which would surely violate any thresholds set under this scenario.

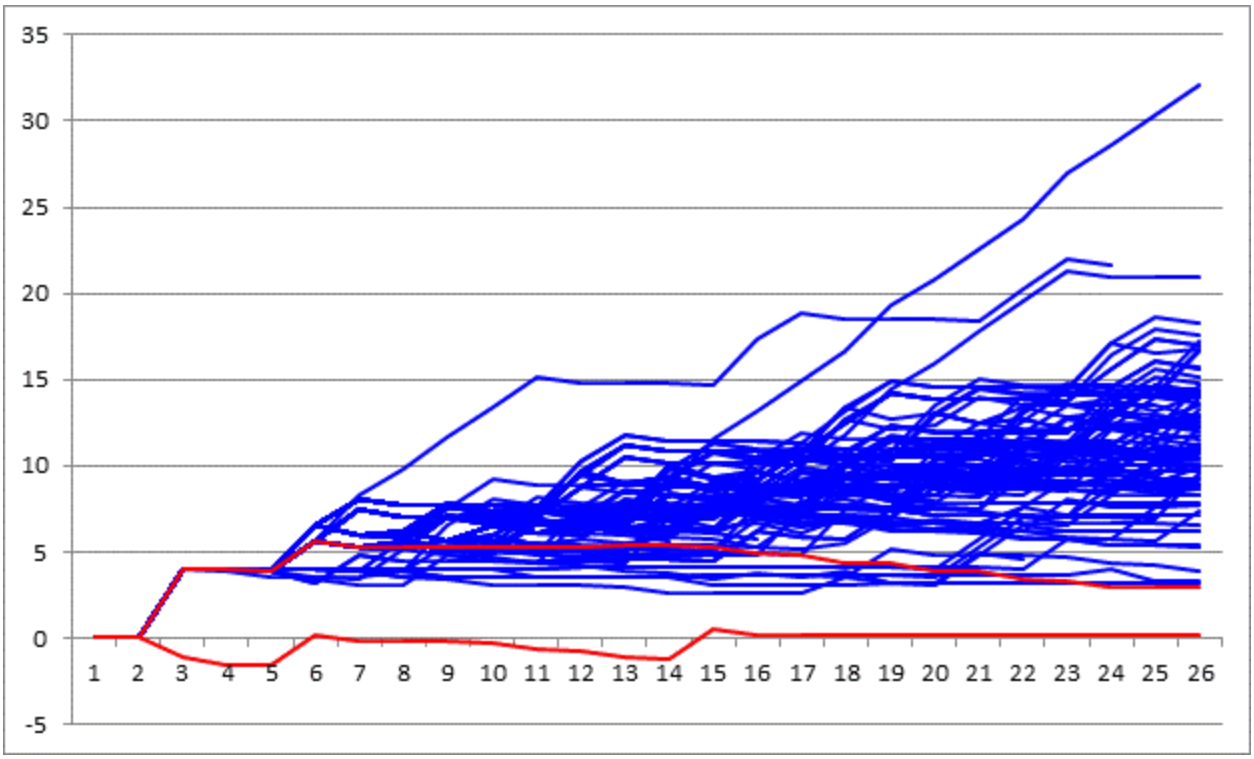

Figure 15 Multihop Order 1

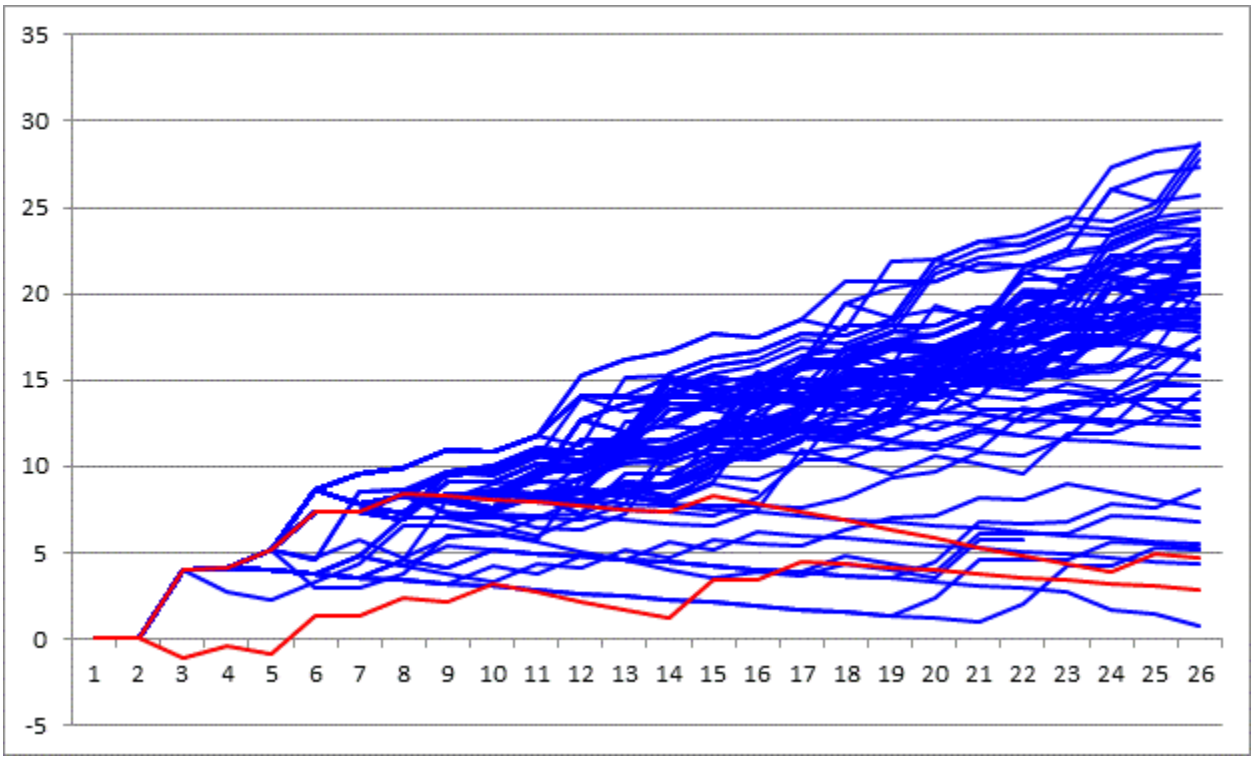

Figure 16 Multihop, Order 3

As can be seen, the observations noted above hold true for other types of attacks as well, over the same number of sequence, the grouping of the Multihop attack flows grow closer with an increase from order 1 to order 3.

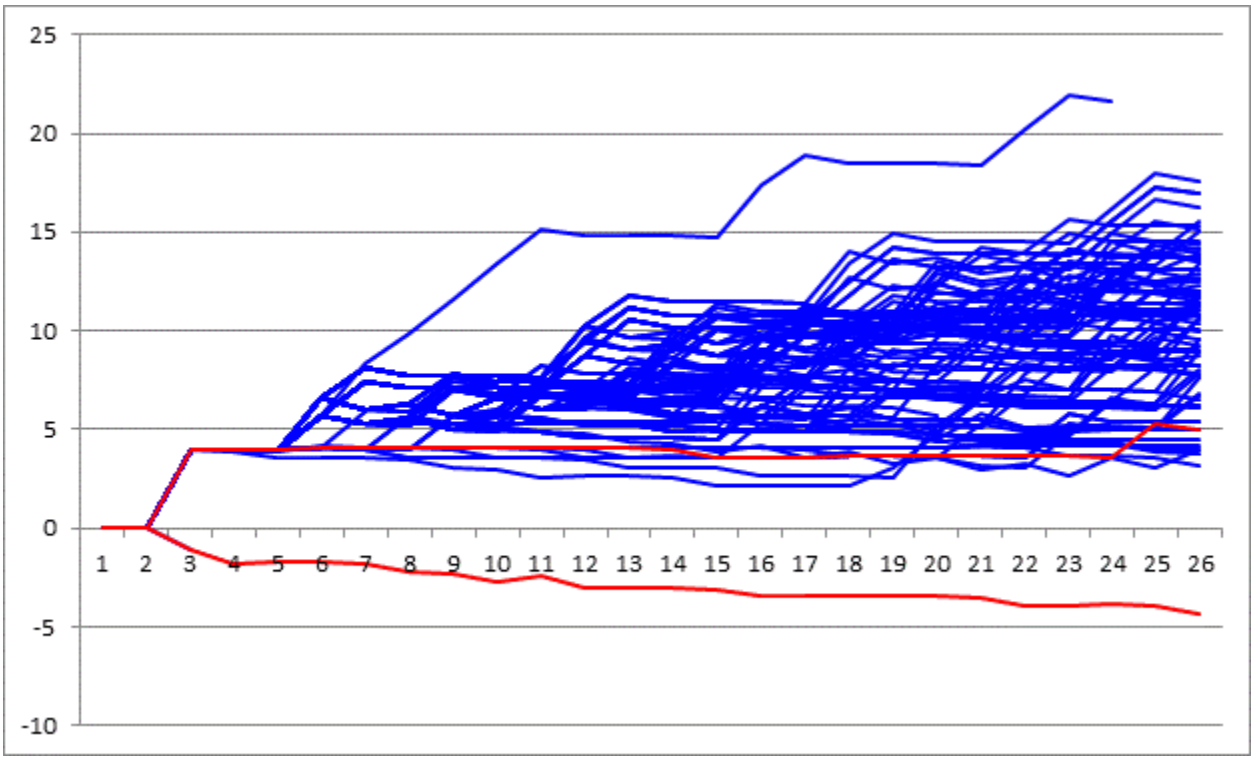

Figure 17 Satan, Order 1

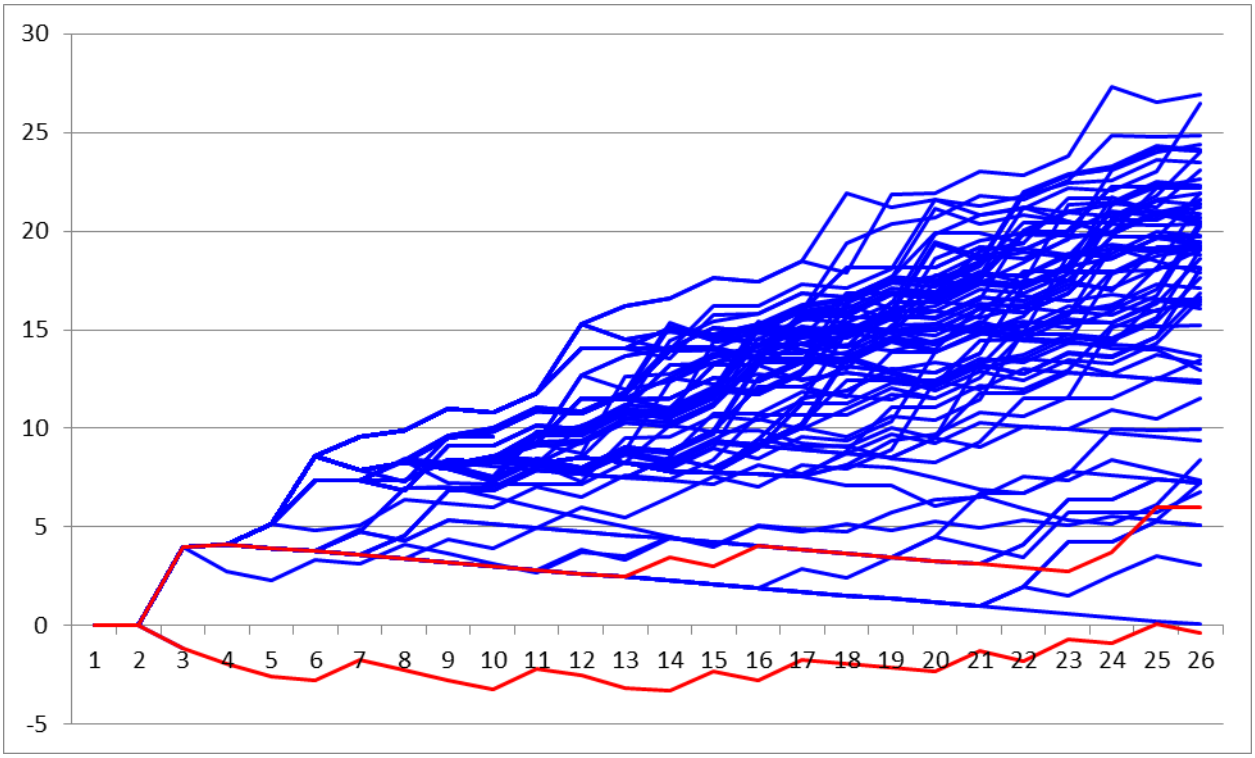

Figure 18 Satan, Order 2

Once again, a similar effect on the increase of order is observed. While not all attack traffic could be correctly classified without false positives, a threshold which will capture all attack flows will include a small subset as false positives.

Effect of Weight

In attempt to compensate for a lack of attack training data, the weighted SPRT function Formula 4 above is applied. The background reference matrix is significantly better trained than any of the attack matrixes, likely sufficiently falling above the trained threshold from Formula 3 in all transitions. For the attack reference matrix however this is less likely, both due to the lower availability of data, as well as a more narrow slice of traffic, which will likely populate a fewer variety of transitions. We again investigate the two class scenario for weighted and unweighted versions and explore the effect on captured data. In addition to two class we fix the order of the Markov history chain to 2.

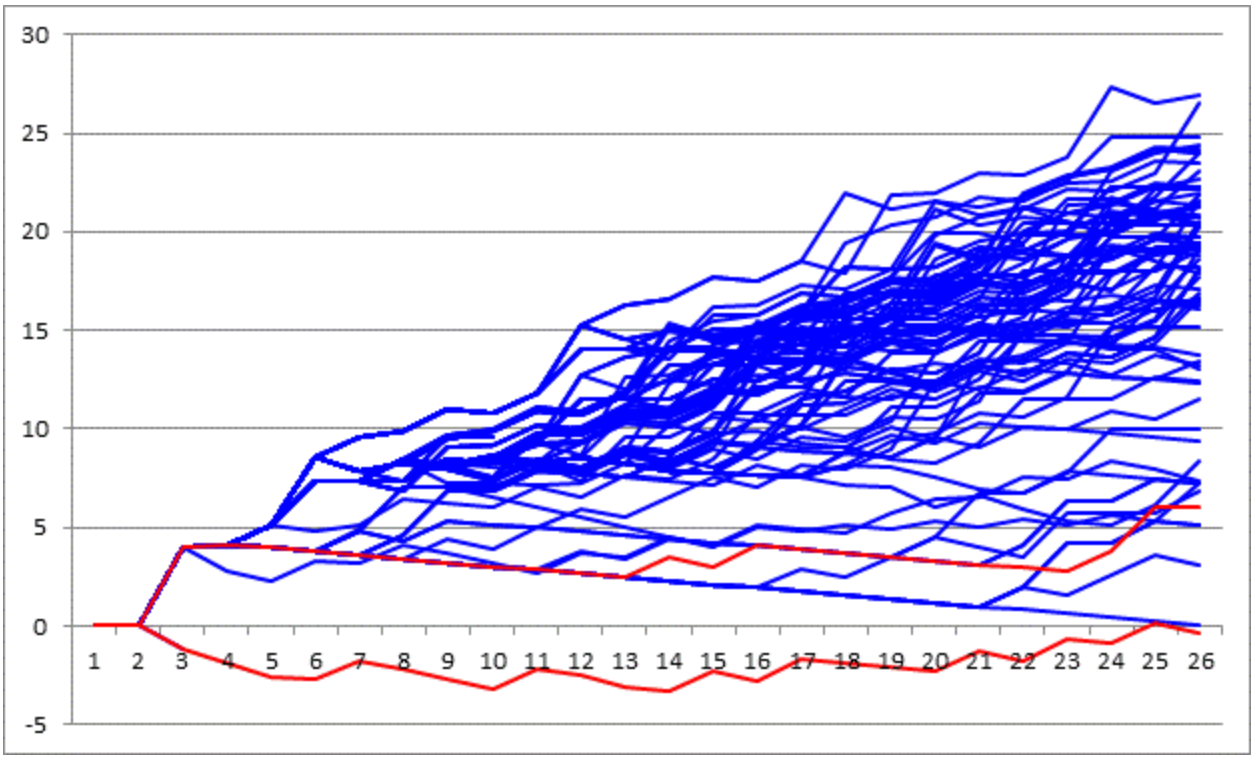

Figure 19 Satan, Order 2, Unweighted

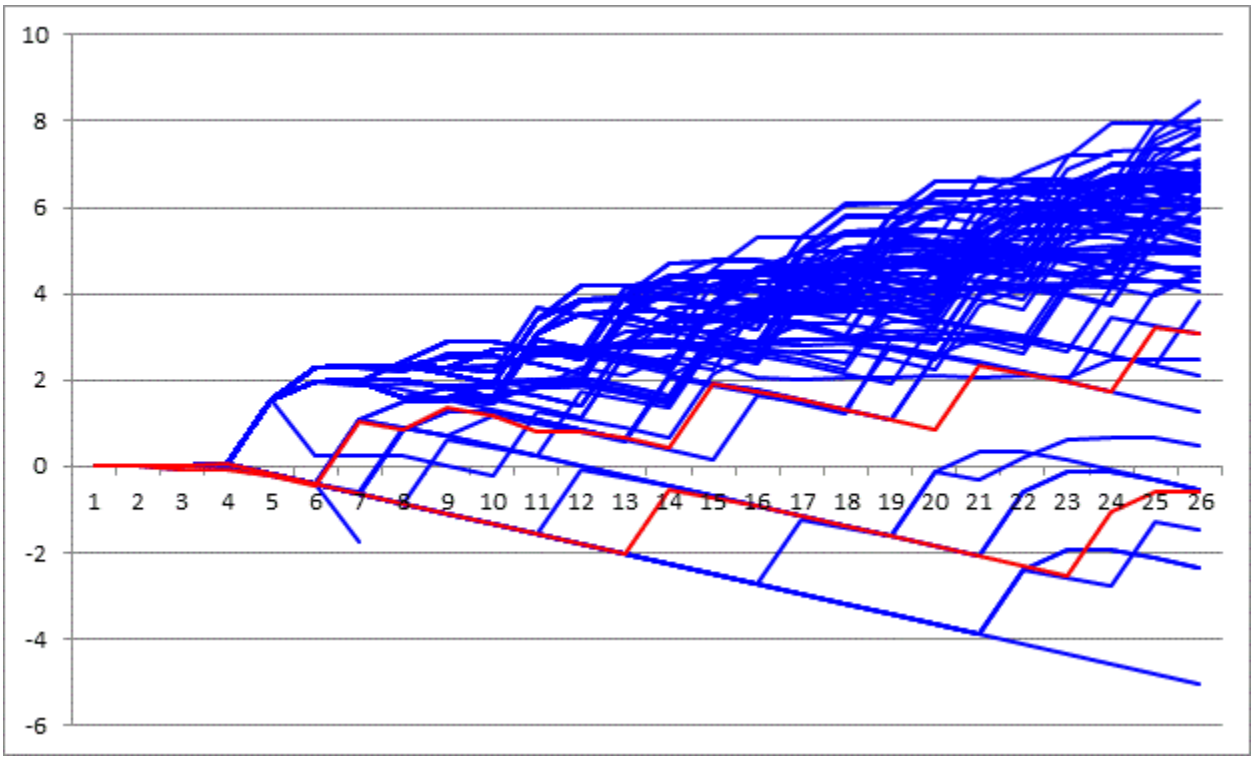

Figure 20 Satan, Order 2, Weighted

In this scenario, it appears that the weighting formula actually had a negative effect on the SPRT graphs. As stated above, it is set to 1000 for the duration of experiments performed in this paper. Though of interesting note, every flow appears to have a very similarly probable initial sequence, noted by the first four graphed values having very minimal change. This introduces a new area for exploration, if a deviation in this initial control sequence is noticed, should the SPRT values be affected more significantly. This is later explored through the pseudo timing algorithm.

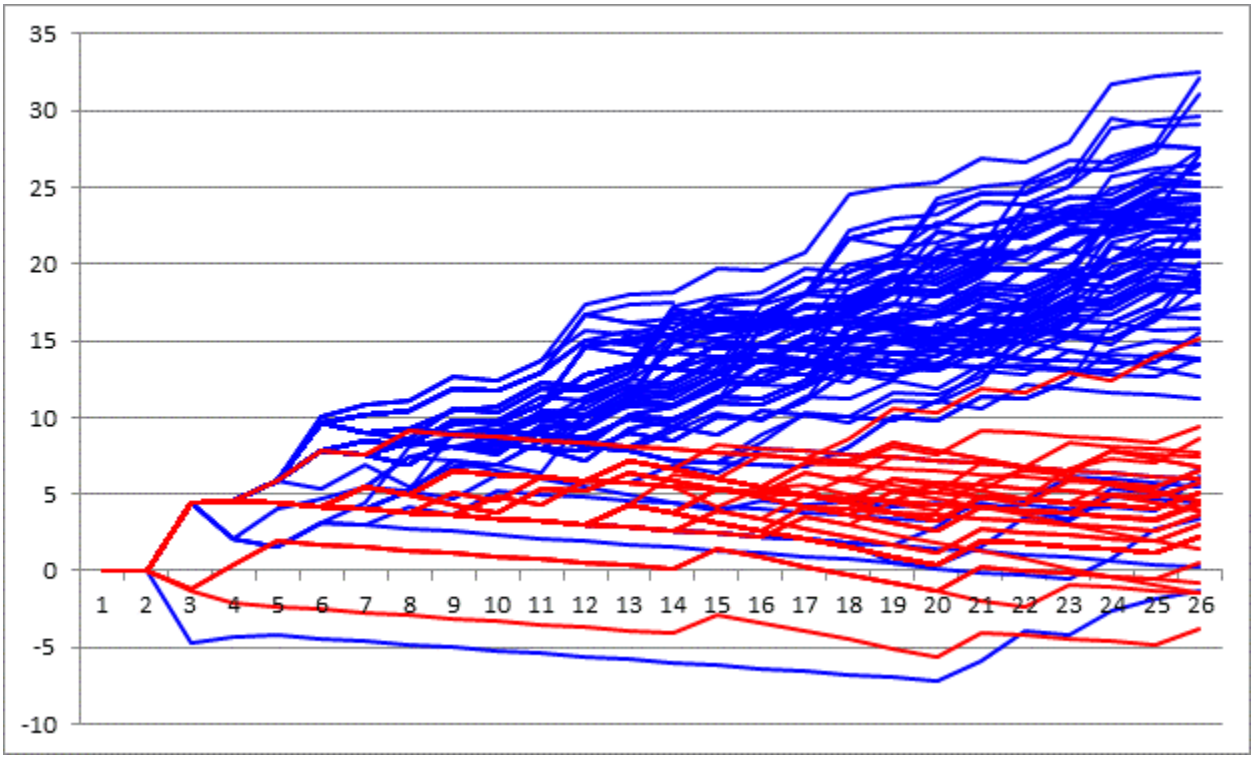

Figure 21 Warez, Order 2, Unweighted

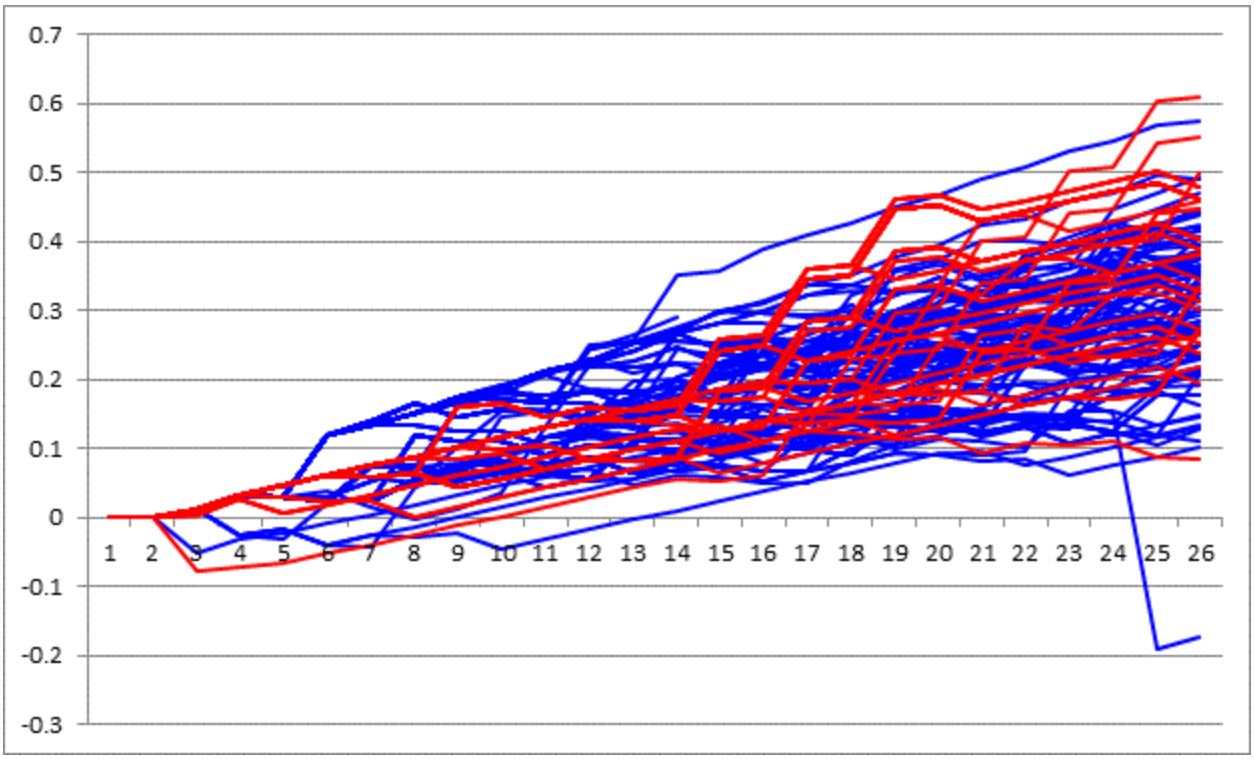

Figure 22 Warez, Order 2, Weighted

Contrary to what is shown in Satan above, Warez produces truly unreliable results. It appears that because matrices are overwhelmingly better trained on background data, that attack traffic continues almost indistinguishable from background traffic because of a proportionately low degree of confidence compared to the background matrix.

Unfortunately it appears that weighting the SPRT formula has not helped to compensated for a lack of training. It is possible that this is partially to do with an interesting finding related to the training distributions discussed in Chapter 5, which indicates that the matrix may be sufficiently well trained with fewer data points than expected.

Effect of Classifiers

Because SPRT is ultimately a sum of ratios, in the most general case we must have two reference points or hypotheses for which to compare effectively, though the one class SPRT variant discussed in Chapter 4 above is a possibility. The other possibility is the multi class case also introduced in Chapter 4 above. In this instance the flows in question are compared against a more precise matrix, trained only on a specific type of attack. Up until this point the results this

paper have discussed have been what we refer to as the two class case for consistency. That is, all available attack data is used to train the reference classifier. While this give us a balanced efficiency scenario between a potentially large number of Markov Matrixes required, and failing to be too specific when testing for membership to an attack class, other options should also be explored for their effect on the SPRT. In attempt to classify an unknown attack the single class method is also included. In an attempt to classify an attack as belonging to a Markov Chain consisting of all attacks, two class is used. In practice single or two class may be used as a sort of pre processor classification to classify most normal traffic, and reduce the flows for which multi class is performed. Multi class is an attempt to narrow the scope of the attack classification to just one type. By training the attack Markov Chain on a single attack type it may be more obvious a flow is a member of that particular class rather than when only compared to the general attack class. For the scope of this section we again fix order 2 and examine unweighted ratios.

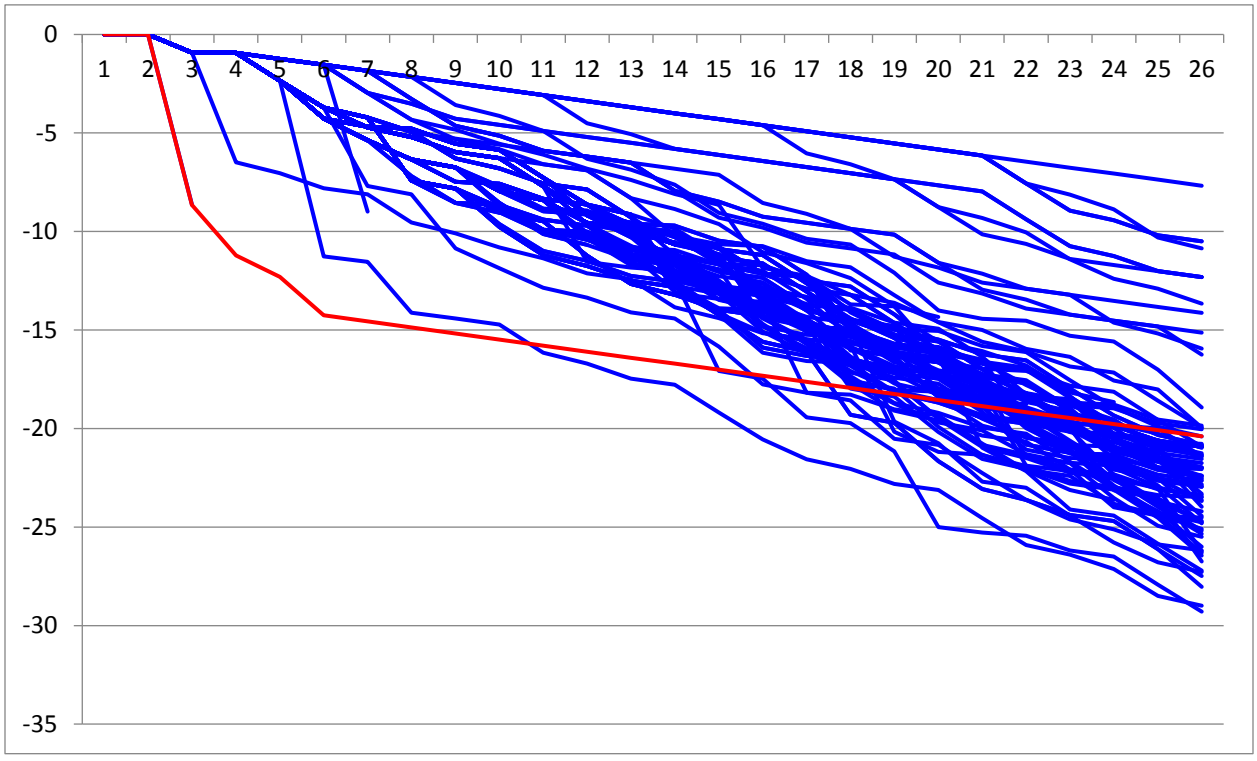

Figure 23 Portsweep, Order 2, Unweighted, Single Class

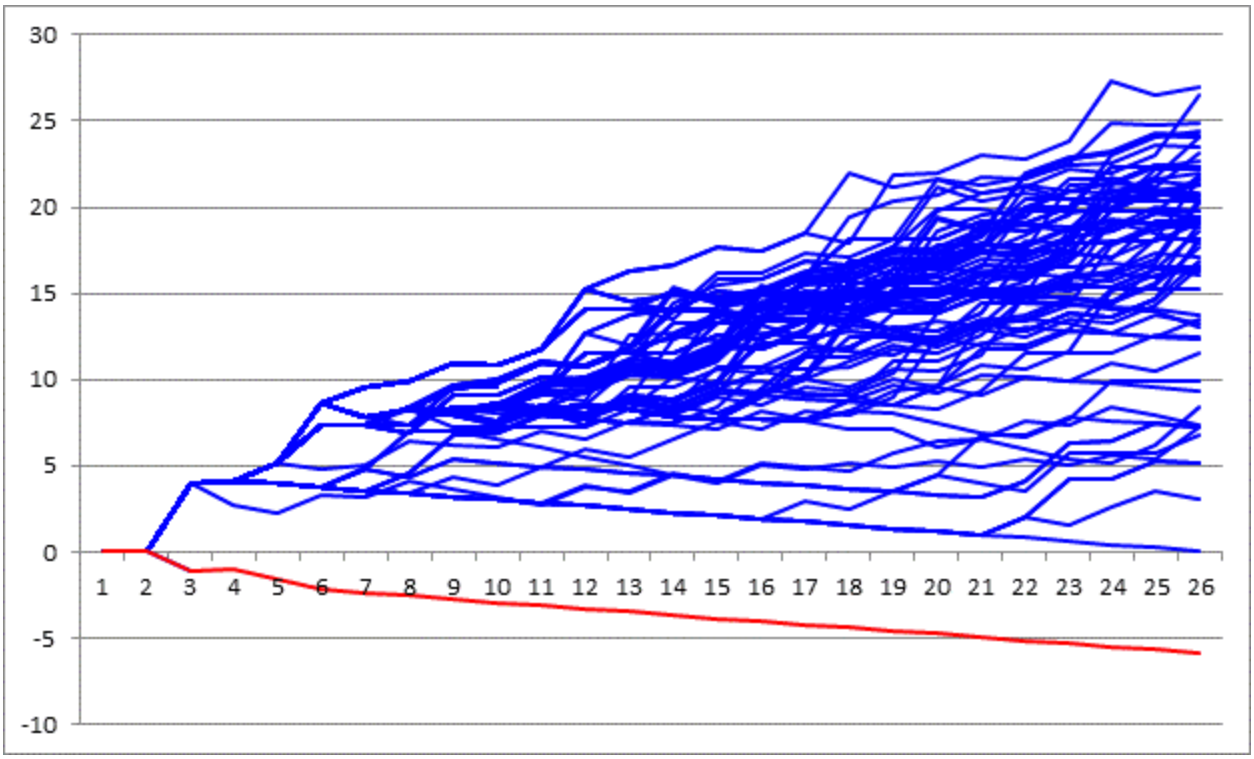

Figure 24 Portsweep, Order 2, Unweighted, Two Class

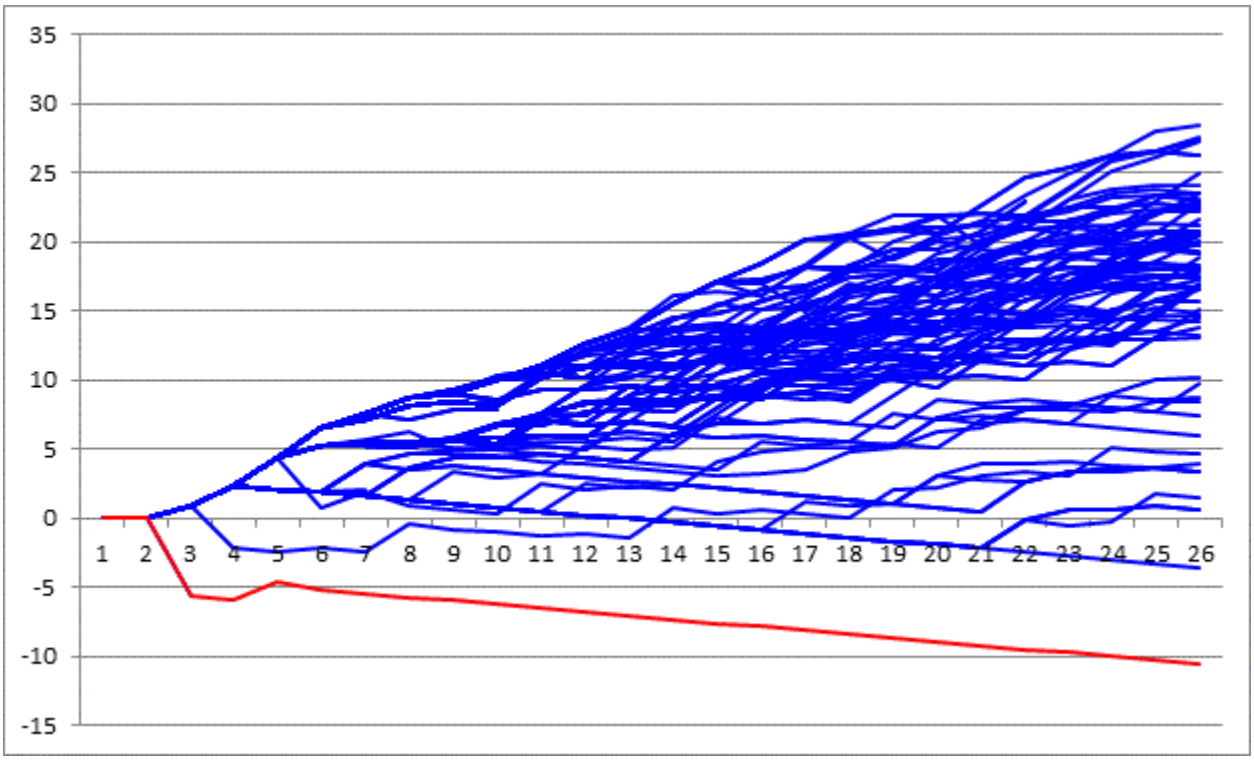

Figure 25 Portsweep, Order 2, Unweighted, Multi Class

Initially Portsweep should be noted as one of the most obvious forms of attack to identify with this classification scheme, as it is a scan against one address. There are possibly multiple concurrent connections, leading to higher numbers for each stream of packets, or step of the flow's history as seen by our context. This should appear unusual compared to other background traffic, which should have low counts for each stream of packets. Even under this observation, we see that the one flow of packets classified under this scenario becomes increasingly more obvious through an increase in Markov Chain precision. While Portsweep appears obvious even in the single class, the two class is evermore clear. The attack flow is the only line that even crosses the horizontal 0-axis within the first 25 steps. In the Multi class the difference is even more apparent. The closest flow at step 26 to the two class was around a score of 5 away. In the multi class the end score is decreased further still, with a score almost doubling from -5 to -10.

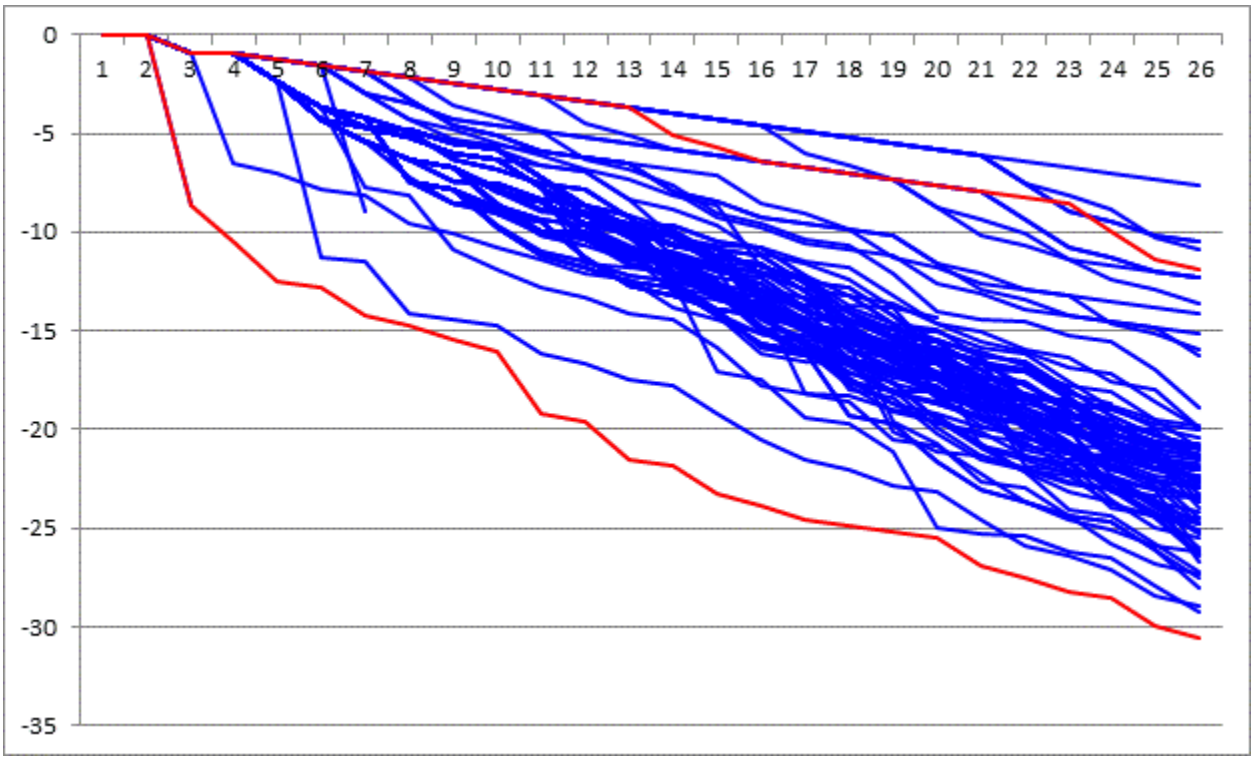

Figure 26 Satan, Order 2, Unweighted, Single Class

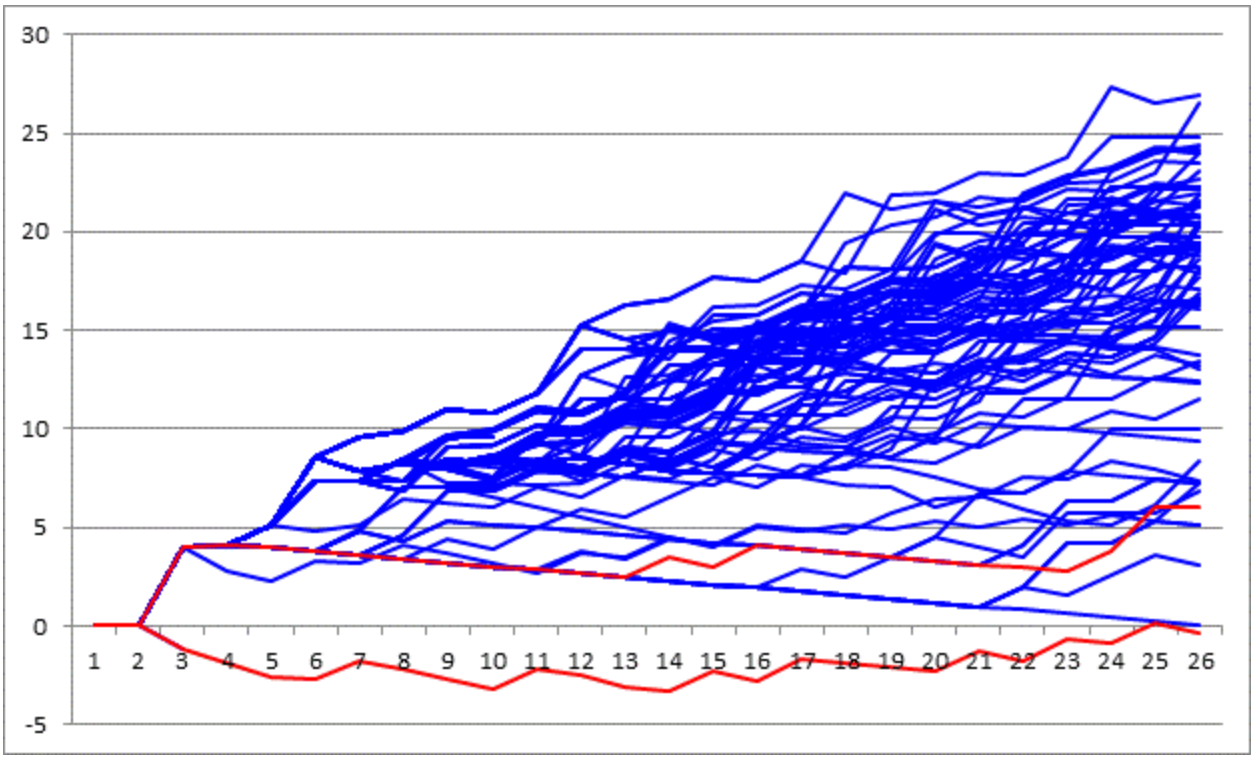

Figure 27 Satan, Order 2, Unweighted, Two Class

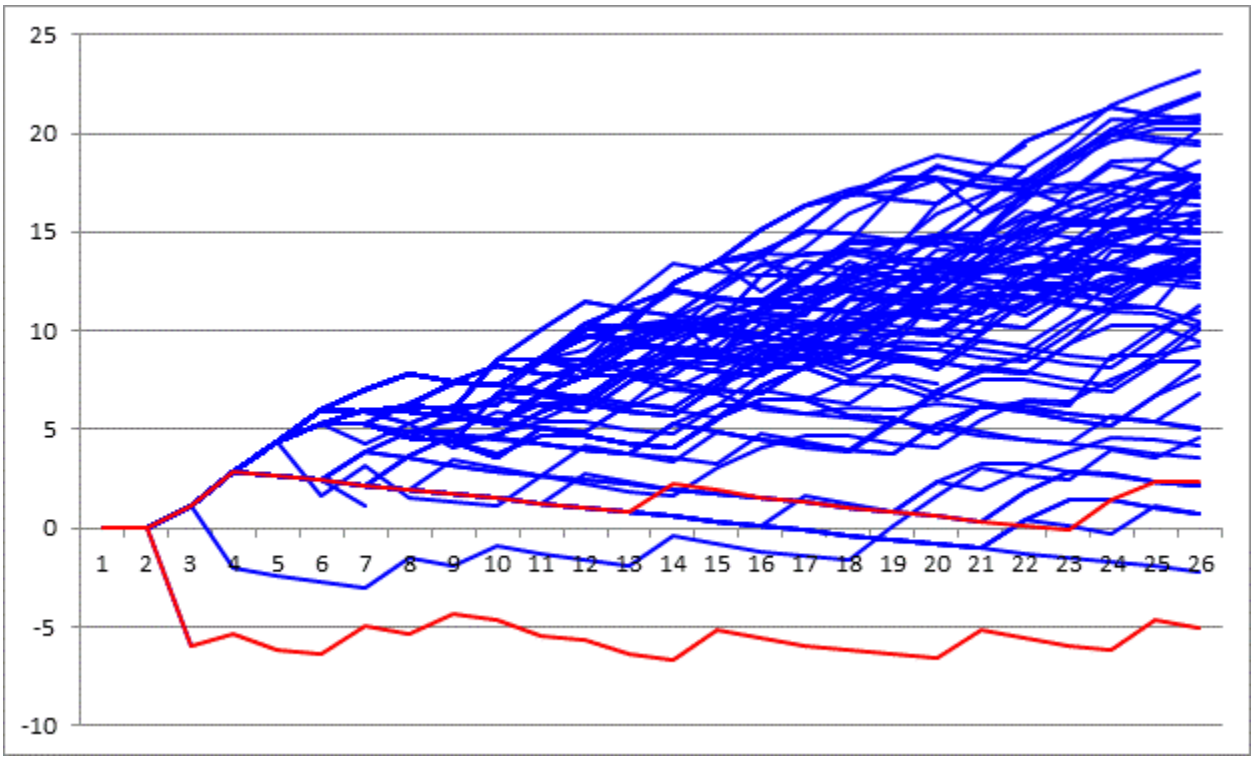

Figure 28 Satan, Order 2, Unweighted, Multi Class

Observing Satan attack data, it is the most difficult of the selected attacks to classify, the single class SPRT is essentially worst case, with one representation of the attack going in a completely opposite end of the background reference data. Observation based solely on this would lead an analyst to almost immediately seek alternate methods for detecting the attack. When we continue and examine the two and multi class cases, the results look much better. In each case only 3 to 5 background flows are seen between the two, this is much more optimal than the wildly encompassing Satan flows seen with one class SPRT. The significant improvements in this regard demonstrate the possibility of the method to identify the attack traffic through increased precision.

Effect of Pseudo Time Dependency

Because the effect of timing requires the equivalent of an increase in order of the size of the Markov Chain, the experiment is only performed for first and second order, though all other parameters are still manipulated as seen above. A complete listing of the SPRT graphs for these selected attacks is included in the appropriate appended section.

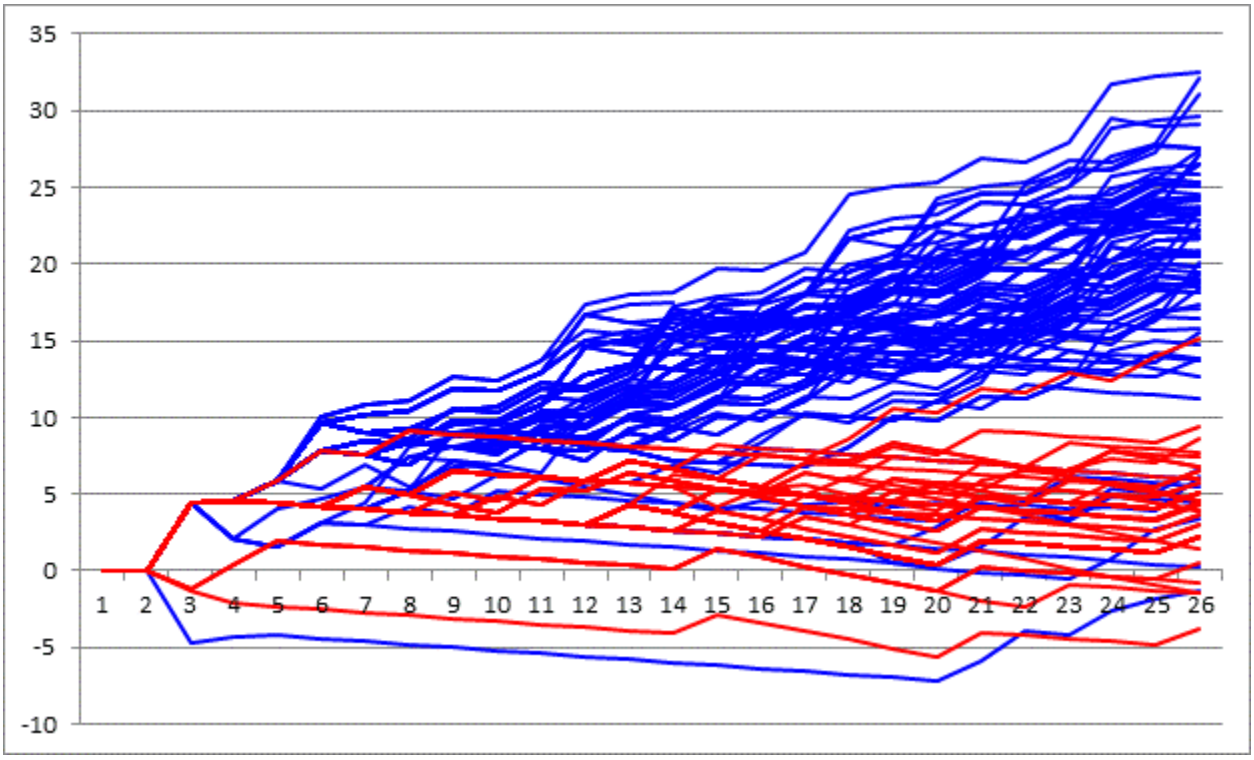

Figure 29 Warez, Order Two, Unweighted, Two Class, No Time Dependency

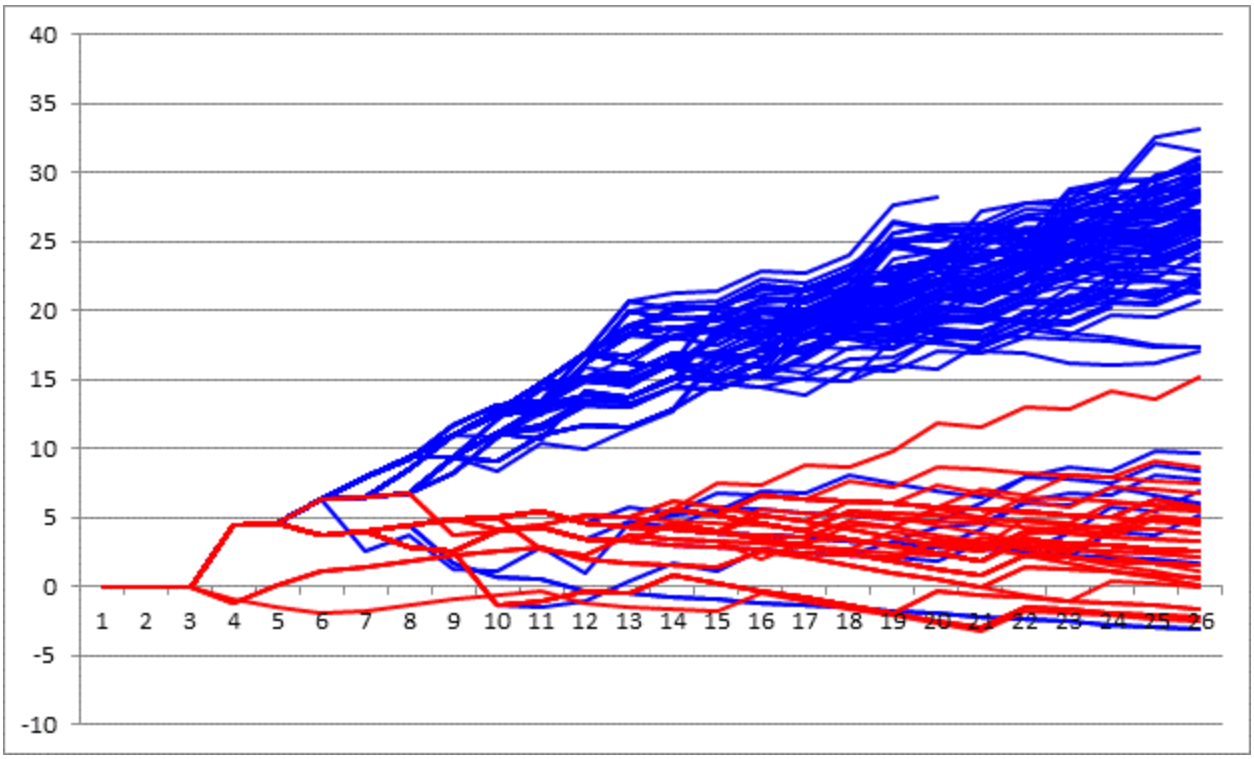

Figure 30 Warez, Order Two, Two Class, Unweighted, Pseudo Time Dependency

Comparing the above two graphs, when the pseudo timing scheme is introduced the grouping significantly improves for both background and attack data. While it does not appear that the classification would be decided any sooner by a threshold, within the first 10 steps affected, it does give opportunity for better threshold scheme later in the sequence. In addition, an outlier background flow that previously scored below all of the attack data has scored higher, significantly more in line with other reference data, with the introduction of this time dependency, indicating a potential reduction in the false positive rate as well.

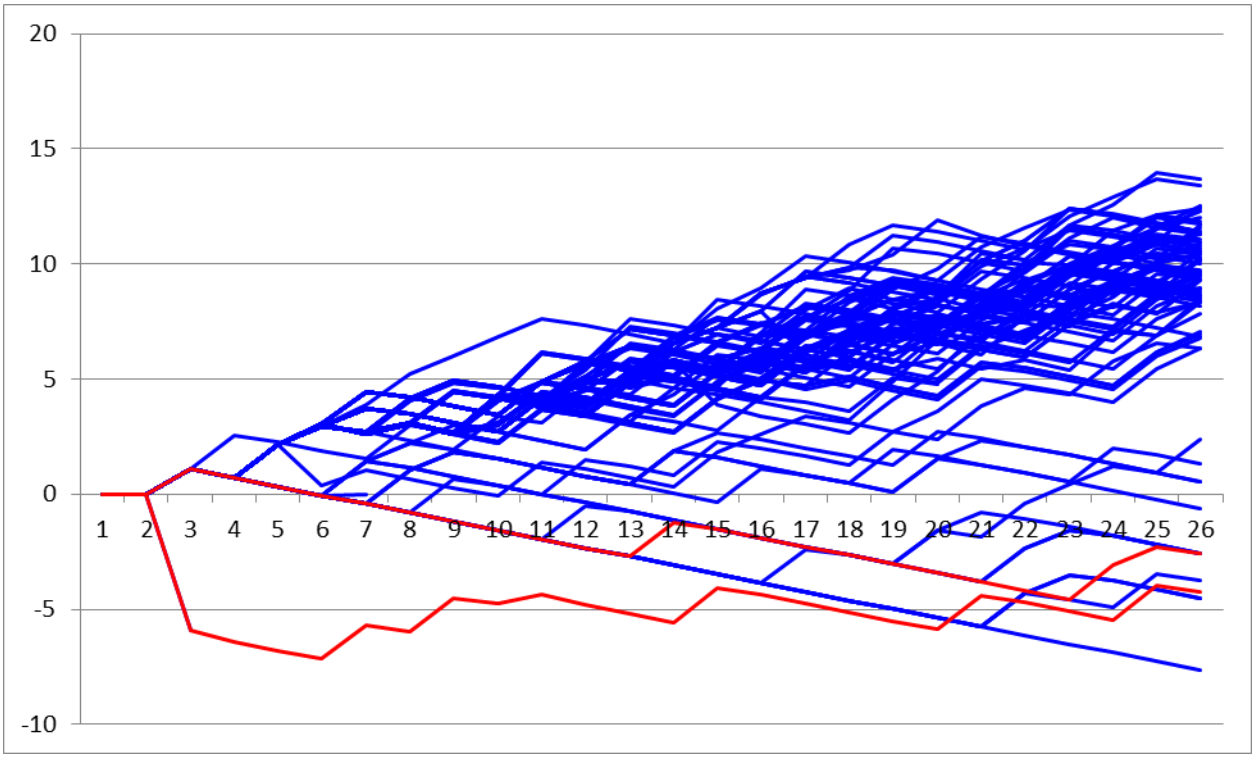

Figure 31 Satan, Order One, Unweighted, Multi Class, No Time Dependency

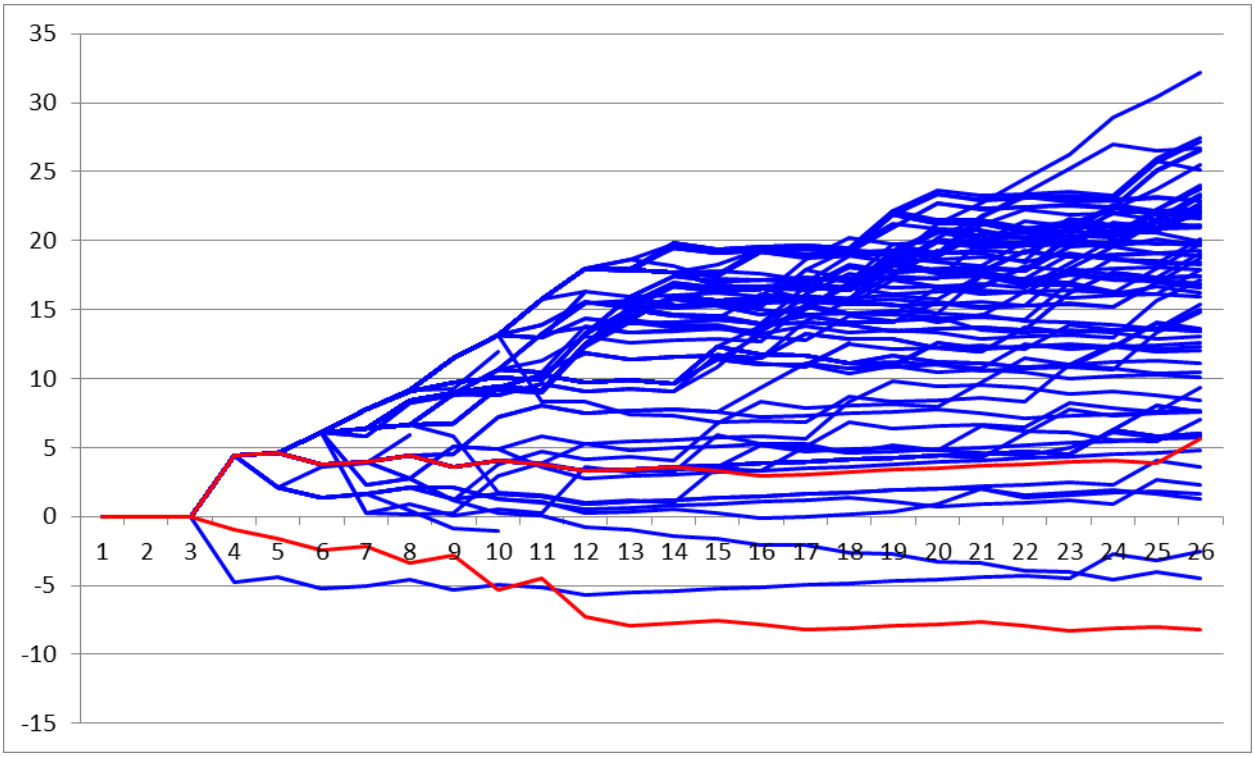

Figure 32 Satan, Order One, Unweighted, Multi Class, Pseudo Time Dependency

While the previous Warez example indeed looked promising, it appears that in other cases, such as Satan above the timing scheme has the opposite effect. The Two attack flows actually become less grouped, and a threshold scheme which selects both attack flows will incur significantly more false positives than it would have without this dependency. It is very likely that this is due to a training issue, as training the attack matrix with so few flows gives almost no data points for which to reference the first several transitions, and deviation of an attack even slightly in the several time-dependent steps, from previously seen similar attacks will give poor performance.

Other Interesting Results

Initially the practical limit on packet sequence was decided to be 10, that is, anything above 10 packets from a flow in order, in one direction would be counted as the same thing for the Markov Chain's purposes. The intent was to for this number to be sufficient enough to distinguish the majority of flows. This limit was established somewhat arbitrarily, but it appears that in practice it could have been much smaller, as seen below by a sample of the first order histograms from the appendixes. In these graphs the X axis corresponds to the different histories possible, or in other terms the rows of the matrices described in Chapter 3. The Y axis is a logarithmically scaled integer of the amount of times the Markov Chain was trained for that particular history, or as discussed above, each possible history representing a distribution.

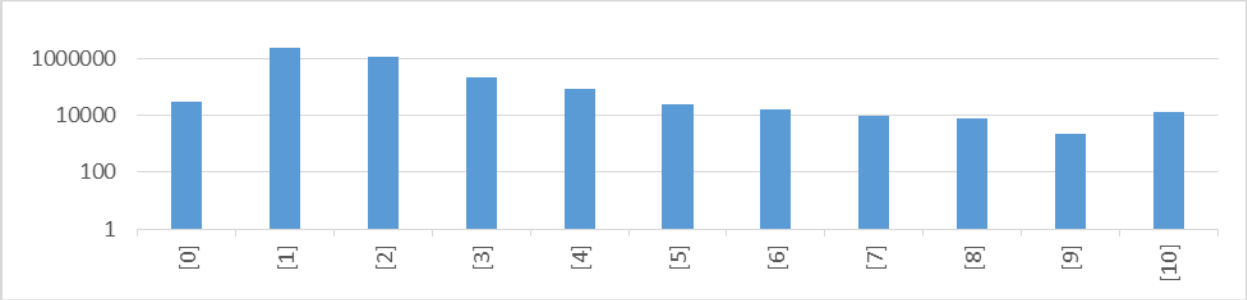

Figure 33 Good/Background Data Histogram

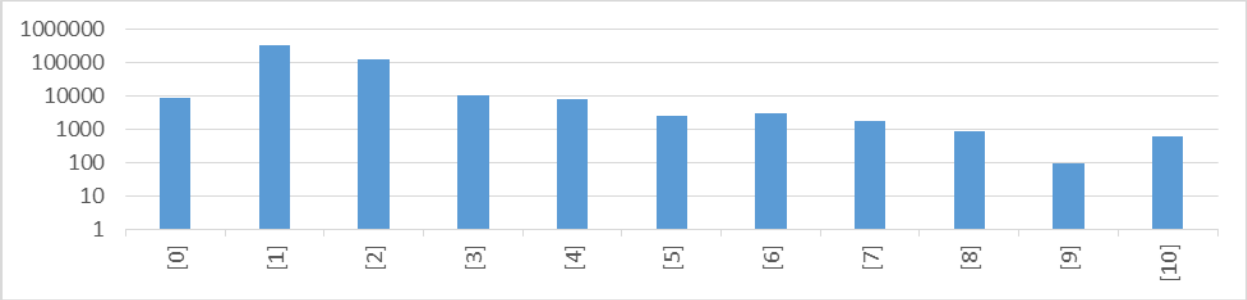

Figure 34 Bad Training Data Histogram

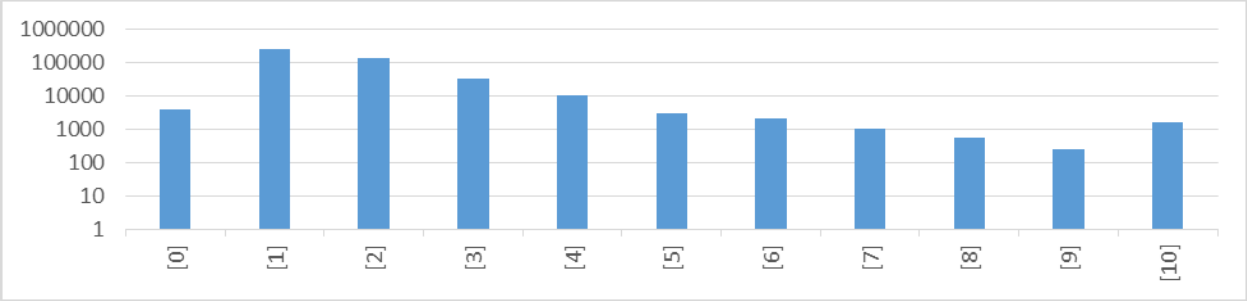

Figure 35 Reference Good Background Histogram

As can be seen the overwhelming majority in all cases comes from one and two counts, as would likely be expected for network traffic in general. Under this scheme the limit likely could have just as easily been set lower, to say 5, and similar results been yielded, while reducing the size of the matrix to an n order fraction of the original size.

The above observations of an unsuccessful weighting scheme are supported by similar training histograms below. The training and testing datasets look very similar, even with one consisting of approximately 100 times the number of transitions

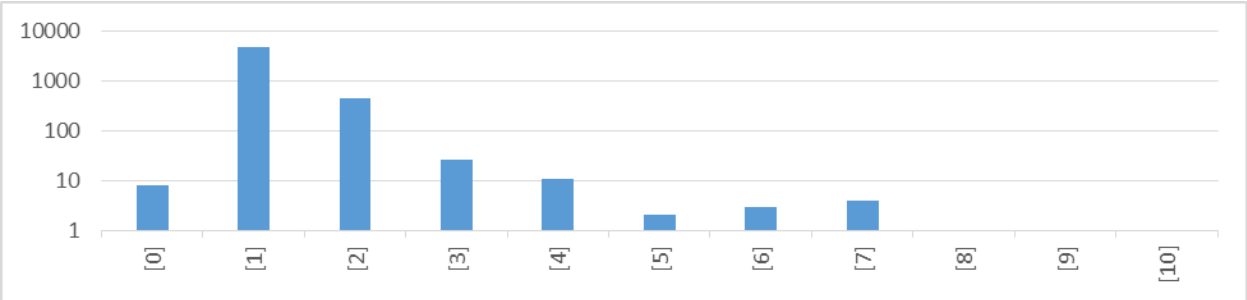

Figure 36 Portweep Testing Data Histogram

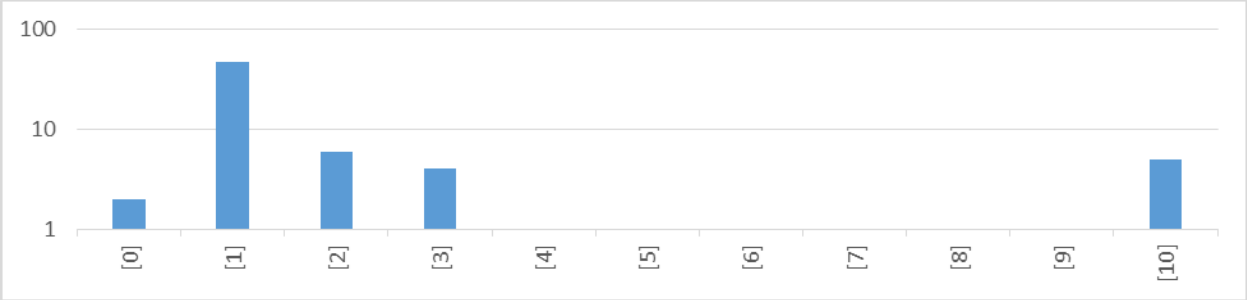

Figure 37 Portsweep Training Data Histogram

While the value of the graph for the background data is significantly higher, it is very proportionate to the values seen in the histogram for the testing data, and would tend to give very similar probabilities when tested for membership.

Drawbacks

While this method does prove effective with certain attack scenarios, and generally manipulations made increases the amount of grouping observed, or separation of attack and background traffic. This does not come without a price however. One of the goals of the project was to remain a saleable and fast algorithm. Each extra insight or dimension to a transition introduced increases the processing overhead significantly, often by a factor of the number of states. While this is necessary to a point, and it is not immediately an issue, the more this is increased, the more complex the algorithm becomes, and the less useful as an in-line method of intrusion prevention it will become. The multi class method in particular, the scaling required with the number of attacks becomes very large. While this does allow for additional performance tuning options, simply by turning on and off the checks against a particular attack Markov Chain, to test for all such attacks would be a computationally intensive ordeal.

Another potential issue is network architecture. While testing for this issue was not done, Markov Chains trained on different network parameters may look very different. For example if payload size is limited, then the same channel between two hosts may look very different on another network. This could likely be, and should be circumvented by using a training matrix derived from traffic seen on the network to be tested, but this solution prevents data sharing between areas with different architecture. In this matter the monitoring algorithms could even be

distributed on a subnet or host based system for increased granularity and more precise background Markov Chains.

This highlights another issue that reference data may be difficult to provide. Attack Markov chains could be given as a sort of indicator, and compared against carefully gathered and analyzed reference data in an enterprise or collaborative system. But because background data needs to be trained on an individual setup, background data without any attacks could be difficult to provide. Another tool will be necessary to remove attacks, this potentially being an existing solution, and be able to train the matrix on the cleaned data. Without an existing method of evaluating for clean data, the attack free background Markov Chain could prove difficult to train.

CHAPTER 6. FUTURE WORK

As a general rule it is of note that the more finely grained analysis that can be performed, the more likely it is to yield useful results. There are additional data points from network traffic that could be used to further examine flows and potentially produce notable results. However this should be proceeded with caution, as any additional data from specific packets will stray from one of the primary goals of the project, that is to maintain a low level of information so that the tool can be used in data and privacy sensitive scenarios. In addition, more complex data processing can be explored, extra dimensions added, but is once again subject to violation of a primary goal of the project, increasing the computational overhead and reducing the usefulness for an in-line system.

A next possible step in further examining flow sequences is to look more closely at the direction of the first packet. This will separate cases where a sub sequence might be likely to be seen as the initial part of a connection, an in for example inbound VPN, whereas an outbound VPN may be more suspicious. Exploring this method may make some attacks more apparent in this regard.

There are also a large number of possible ways in which the traffics could be separated for analysis ahead of time, through the protocol, distributing the algorithm for each host, or segmented networks. However it is important to consider when exploring these options that additional complexity may violate one of the goals of the project, as each time traffic is separated in any form, at least an additional training matrices are necessary.

Additional evaluation of the system is likely necessary, a direct comparison to existing intrusion detection systems for example. The testing and implementation of a threshold system would also need to be applied for this evaluation to take place. While very possible, there are several steps before the project can reach that goal.

CHAPTER 7. CONCLUSIONS

This project explored a new avenue for the field of traffic classification. Using a minimal amount of data from network traffic, this paper was able to implement a method for characterizing a flow as a sample of a Markov Chain. These Markov Chains were then used as classifiers, and SPRT analysis performed. Using standard and defined variants to SPRT, grouping and separation of classes was shown, indicating the possibility of a classification scheme to identify attacks or abnormalities. Parameters explored for varied SPRT methods included all combinations of history degree, weighting, number of classifiers, and the introduction of a pseudo time dependency. All of these evaluations were done for four of the attacks seen in the DARPA 1998 intrusion detection dataset and compared to a subset of reference normal data.

While there are many possible avenues left to explore for this approach, this form of classification test has shown to indicate potential success at classifying network attacks with fairly high degrees of accuracy. The system can likely be utilized as both an online and offline analysis system, to prove useful in detecting attacks or anomalies. Some issues do potentially exist with the setup and deployment process, but the method shows sincere potential as an effective method for classifying attack traffic, and should be included in future research into the intrusion detection field.

REFERENCES

[1] R. Chang, "Defending against flooding-based distributed denial-of-service attacks: a tutorial," *IEEE Communications Magazine*, vol. 40, no. 10, pp. 42-51, 2002.

[2] "Protecting from a growing Attack Vector: Encrypted Attacks," *Gartner: Radware.*

[3] L. Hellemons, "SSHCure: A Flow-Based SSH Intrusion Detection System," *ependable Networks and Services. AIMS 2012. Lecture Notes in Computer Science,* vol. 7279, 2012.

[4] D. Denning, "An Intrusion-Detection Model," *IEEE Transactions on Software Engineering* , Vols. Se-13, no. 2, pp. 222-232, 1987.

[5] R. Sommer, "Bro: An Open Source Network Intrusion Detection System," *DFN-Arbeitstagung über Kommunikationsnetze*, pp. 273-288, 2003.

[6] J. Carr, "Snort: Open Source Network Intrusion Prevention," *eSecurity Planet*, 2007.

[7] J. Oltsik, "FireEye Myth and Reality," *NetworkWorld*, 2015.

[8] R. Hofsteade, "Flow Monitoring Explained: From Packet Capture to Data Analysis with NetFlow an dIPFIX," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 2037-2064, 2014.

[9] N. Vaswani, ""Shape Activity": A Continuous-State HMM for Moving/Deforming Shapes With Application to Abnormal Activity Detection," *IEEE Transactions on Image Processing*, vol. 14, no. 10, pp. 1603-1616, 2005.

[10] D. Ariu, "Sensing attacks in Computer Networks with Hidden Markov Models," *Machine Learning and Data Mining in Pattern Recognition. MLDM. Lecture Notes in Computer Science,* vol. 4571, 2007.

[11] A. Sperotto, "Hidden Markov Model modeling of SSH brute-force attacks," *International Workshop on Distributed Systems: Operations and Management*, 2009.

[12] Y. Song, "Markov Models for Network-Behavior Modeling and Anonymization," *Tech Rep*, vol. 29, no. 11, 2011.

[13] S. Asmusen, Applied Probability and Queues, Pringer Science and Business Media, 3002.

[14] G. Casella, Statistical Inference, Thomson Learning, 2002.

[15] A. Wald, "Sequential Tests of Statistical Hypotheses," *The Annals of Mathematical Statistics*, vol. 16, no. 2, pp. 117-186, 1945.

[16] Y. Ou, "An improved SPRT control chart for monitoring process mean," *The International Journal of Advanced Manufacturing Technology,* vol. 51, no. 9, pp. 1045-1054, 2010.

[17] T. A. Beardsley, "The TCP split handshake: Practical effects on modern network equipment," *Network Protocols and Algorithms* , vol. 2, no. 1, pp. 197-217, 2010.

[18] W. M. Eddy, "SYN Flood Attack," *Encylopedia of Cryptography and Security,* pp. 1273-1274, 2011.

[19] *DARPA Intrusion Detection Evaluation Dataset,* Cambridge: Massachusetts Institue of Technology, 1998.

APPENDIX A. TRAINING AND REFERENCE DATASET HISTOGRAMS

Background / Good is the data used in all experiments performed as a reference to normal.

All Attack is the data used in all two class experiments. This is the chain trained on each attack

Reference is the standard blue data used to evaluate attack classification against. A subset of the blue lines are plotted against each attack to better compare success of each attempt

Table A1. Data Counts

	Good/Background	All Attack	Reference Good
Flows	14205	4464	1970
Transitions	3876442	508720	453722
Packets	6239514	1047127	1484289

Background / Good Training Data Histograms

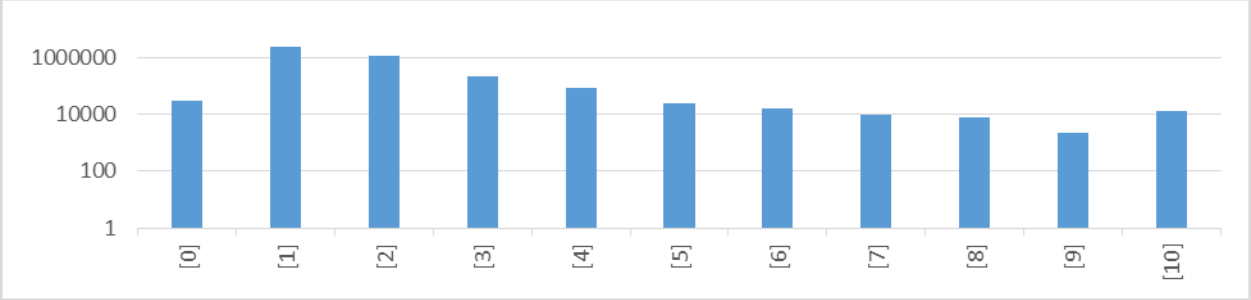

Figure A1. Order 1 Background / Good Training Data Histograms

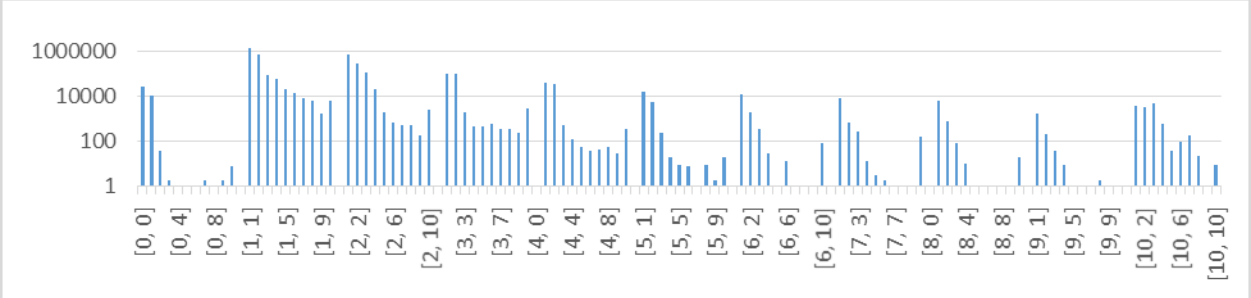

Figure A2. Order 2 Background / Good Training Data Histograms

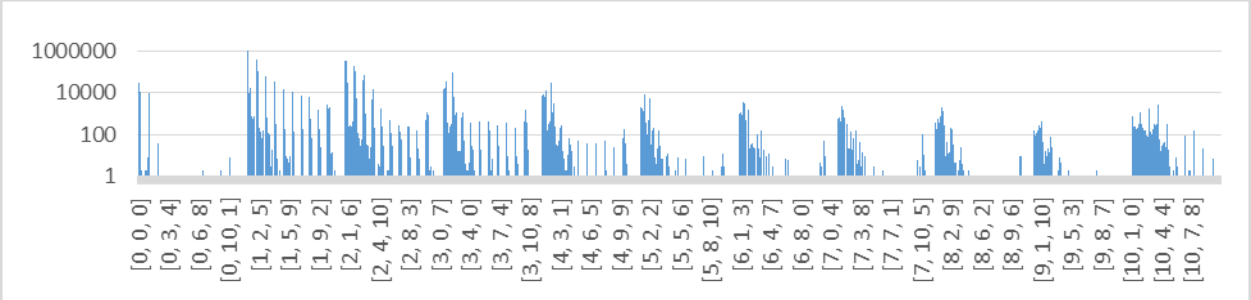

Figure A3. Order 3 Background / Good Training Data Histograms

All Attack Data Histograms

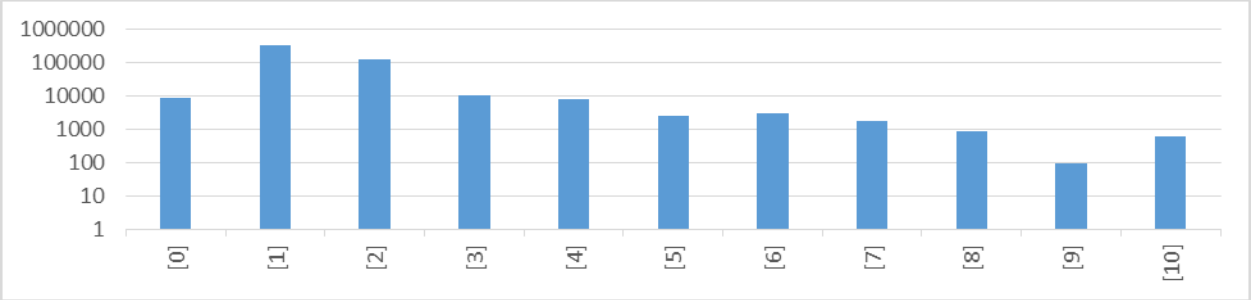

Figure A4. Order 1 All Attack Training Data Histograms

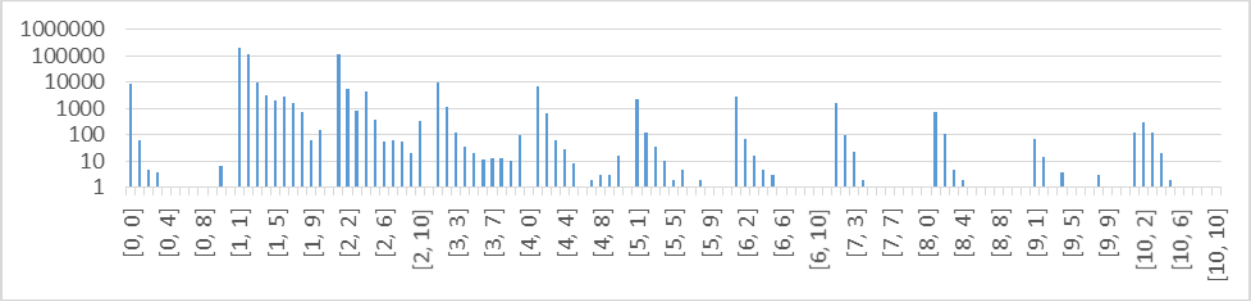

Figure A5. Order 2 All Attack Training Data Histograms

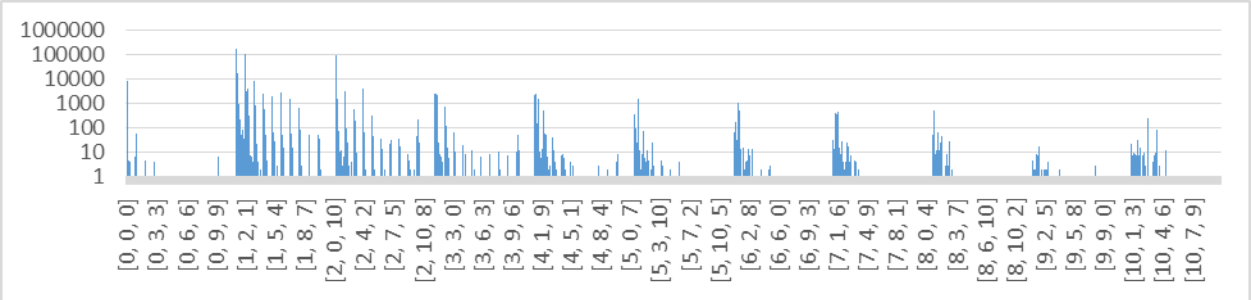

Figure A6. Order 3 All Attack Training Data Histograms

Reference Background Data Histograms

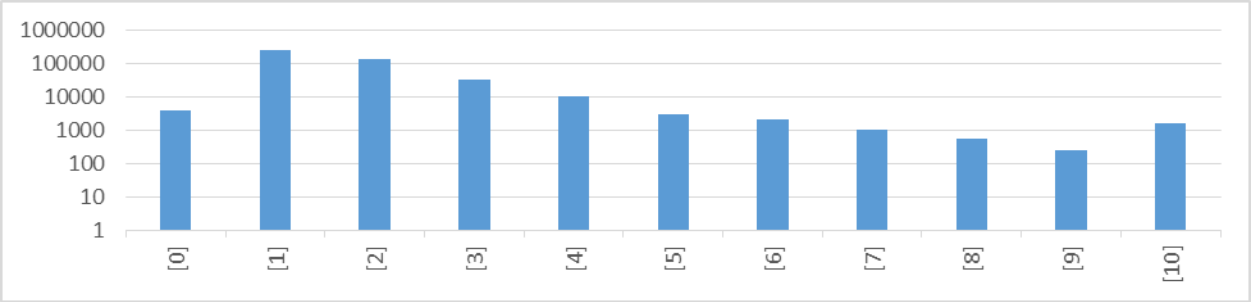

Figure A7. Order 1 Reference Good Training Data Histograms

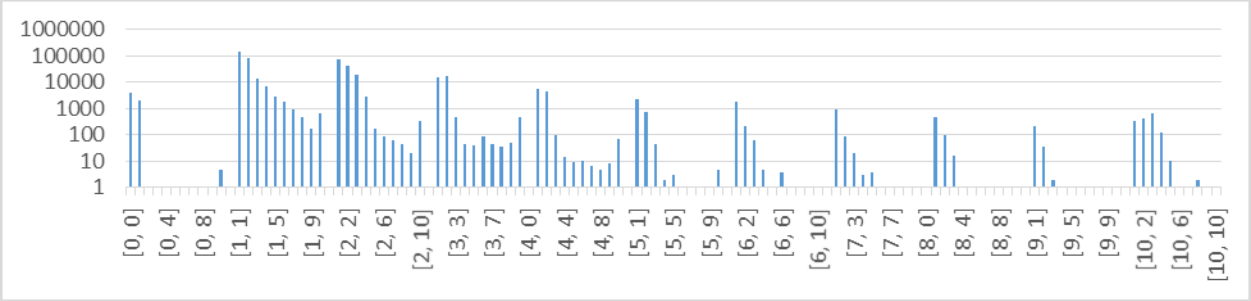

Figure A8. Order 2 Reference Good Training Data Histograms

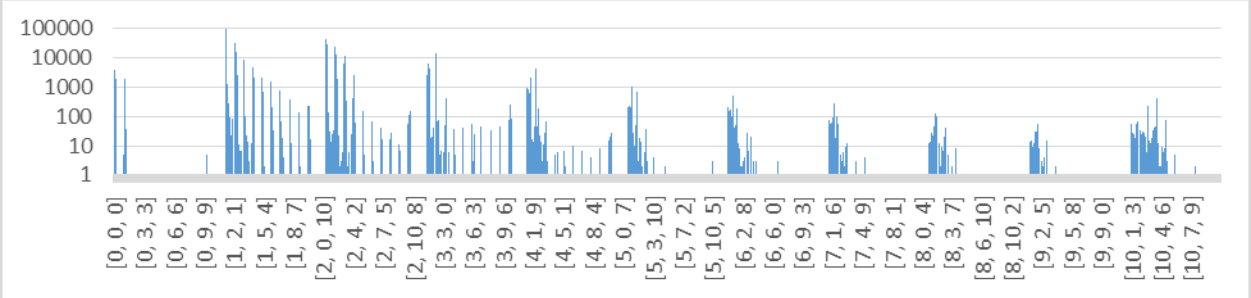

Figure A9. Order 3 Reference Good Training Data Histograms

APPENDIX B. MULTIHOP ATTACK DATASET SPRT GRAPHS

Multhop is a set of attacks from the 1998 DARPA intrusion detection evaluation dataset. Included for reference is an organized listing of SPRT graphs calculated under explained Conditions.

DARPA Description: Multi-day scenario in which a user first breaks into one machine

Table B1. Data Counts

	Training	Testing
Flows	2	2
Transitions	4480	7137
Packets	7275	9407

Training Dataset

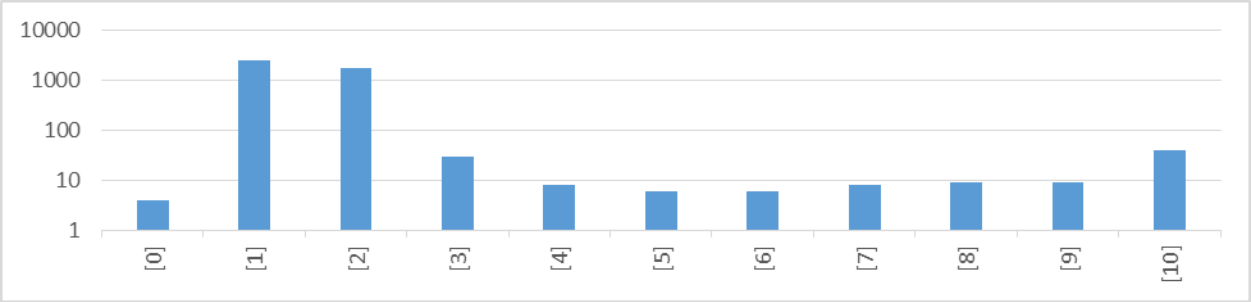

Figure B1. Multihop Order 1 Training Dataset Histogram

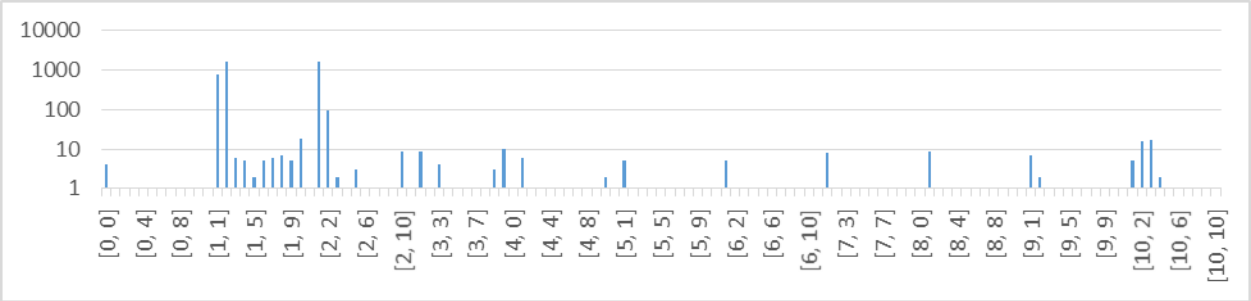

Figure B2. Multihop Order 3 Training Dataset Histogram

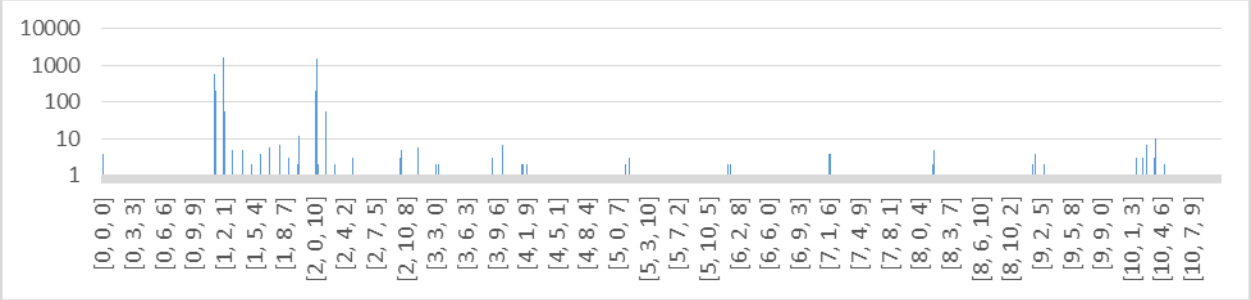

Figure B3. Multihop Order 3 Training Dataset Histogram

Testing Data Histograms

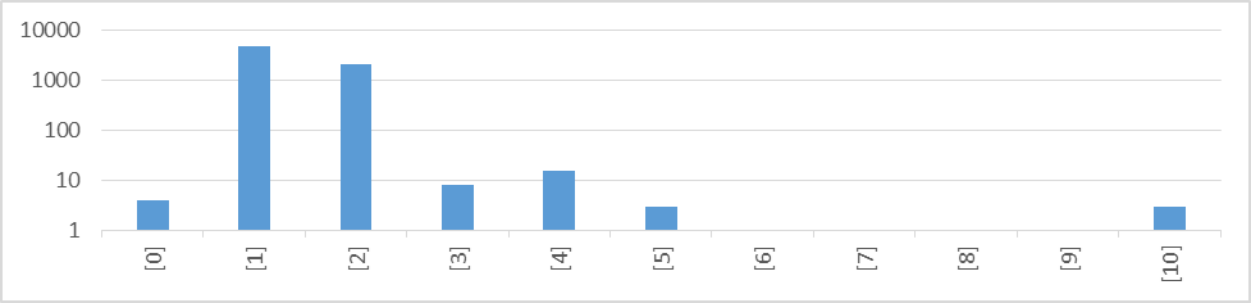

Figure B4. Multihop Order 1 Testing Dataset Histogram

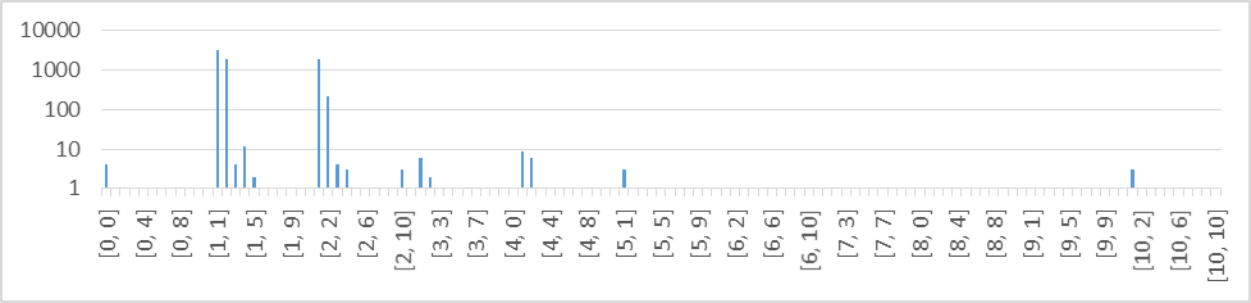

Figure B5. Multihop Order 2 Testing Dataset Histogram

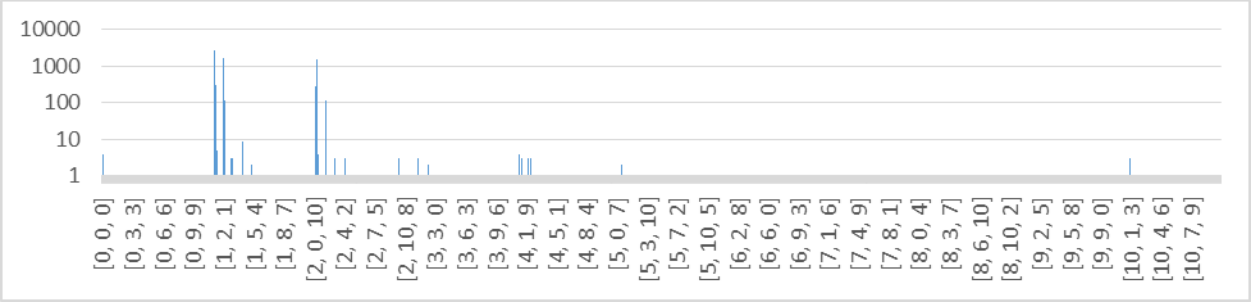

Figure B6. Multihop Order 3 Testing Dataset Histogram

SPRT Graphs

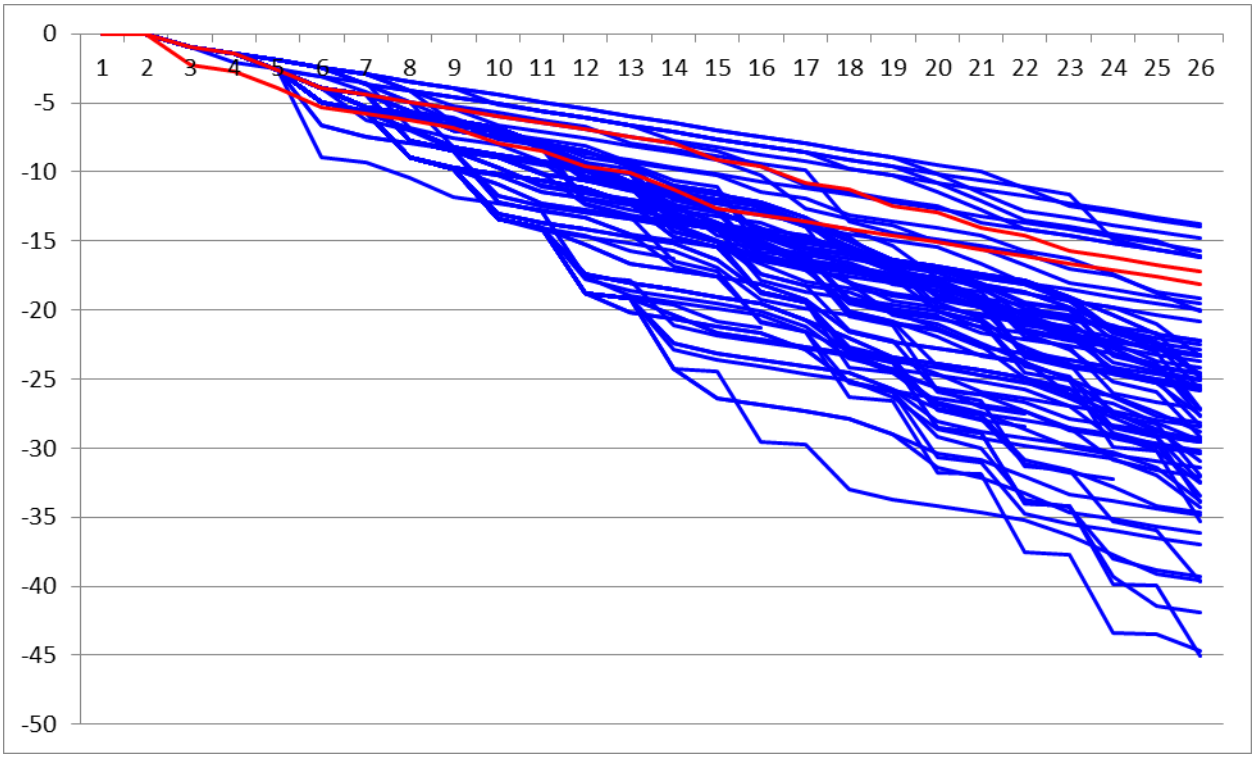

Figure B7. Order One, One class, Unweighted

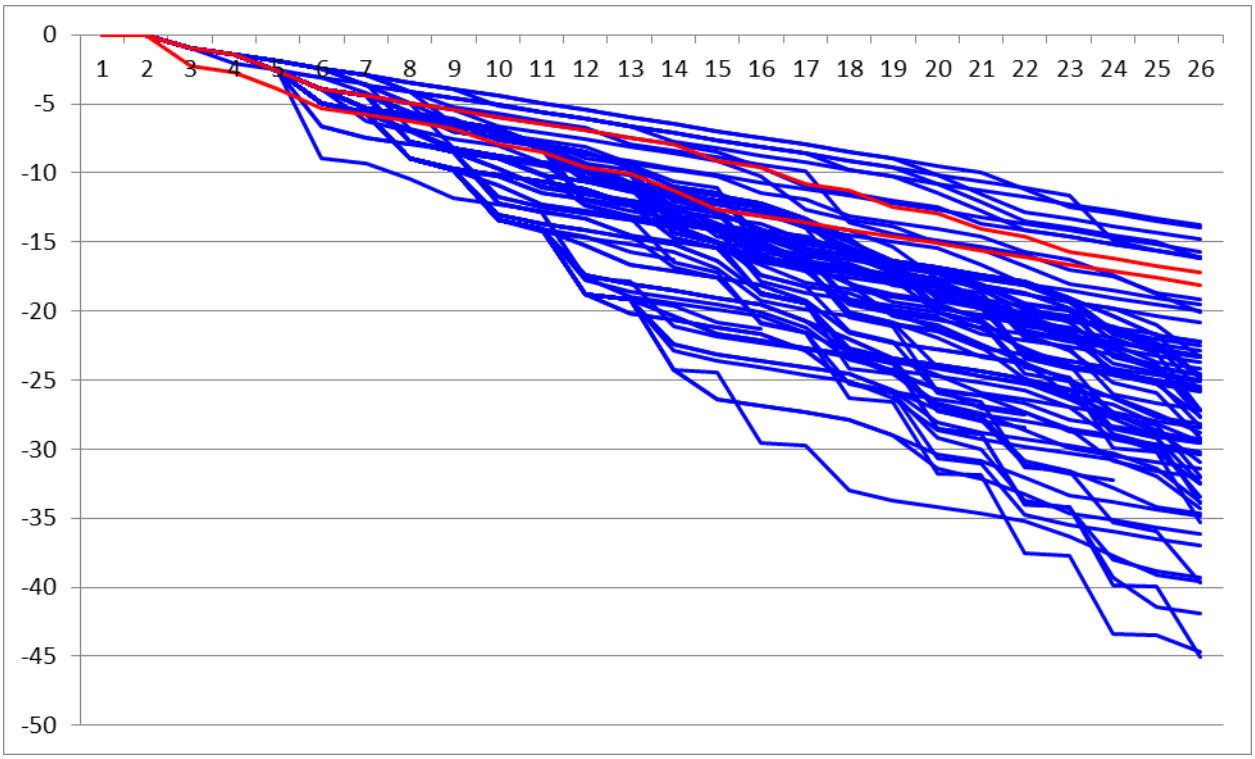

Figure B8.Order One, One class, Weighted

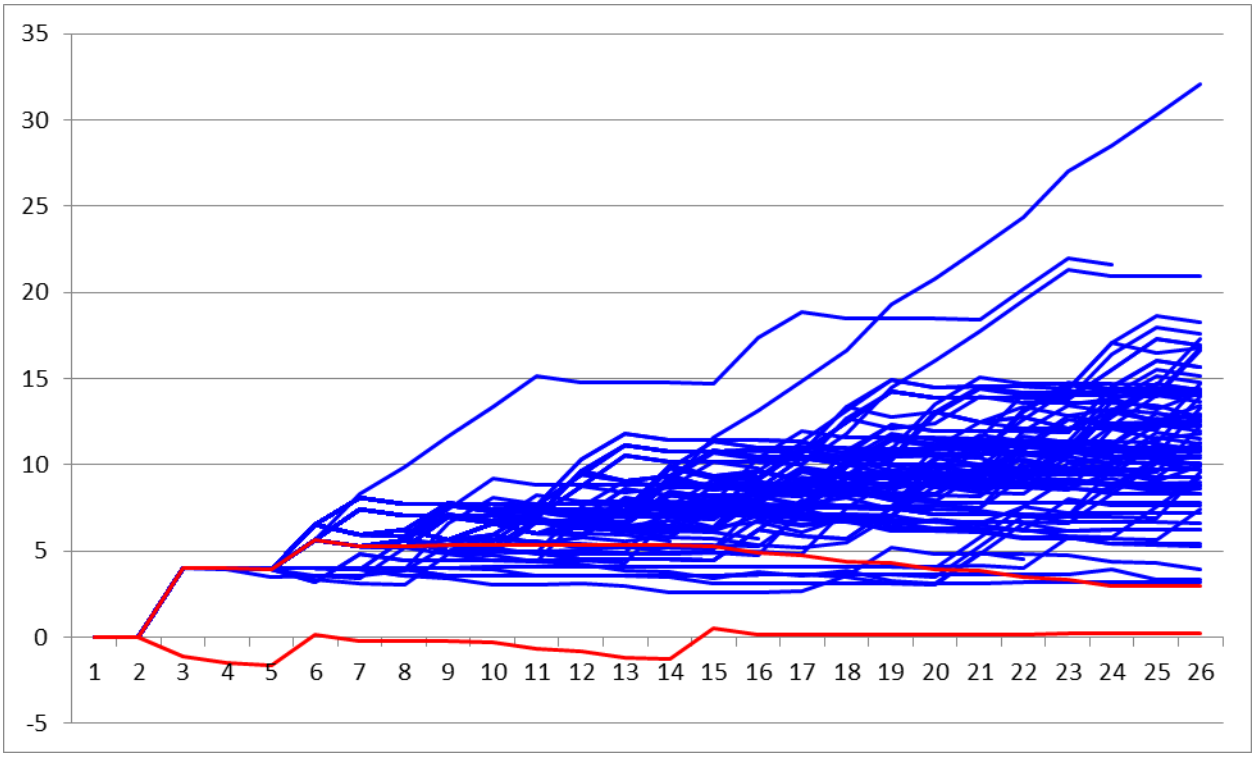

Figure B9. Order One, Two class, Unweighted

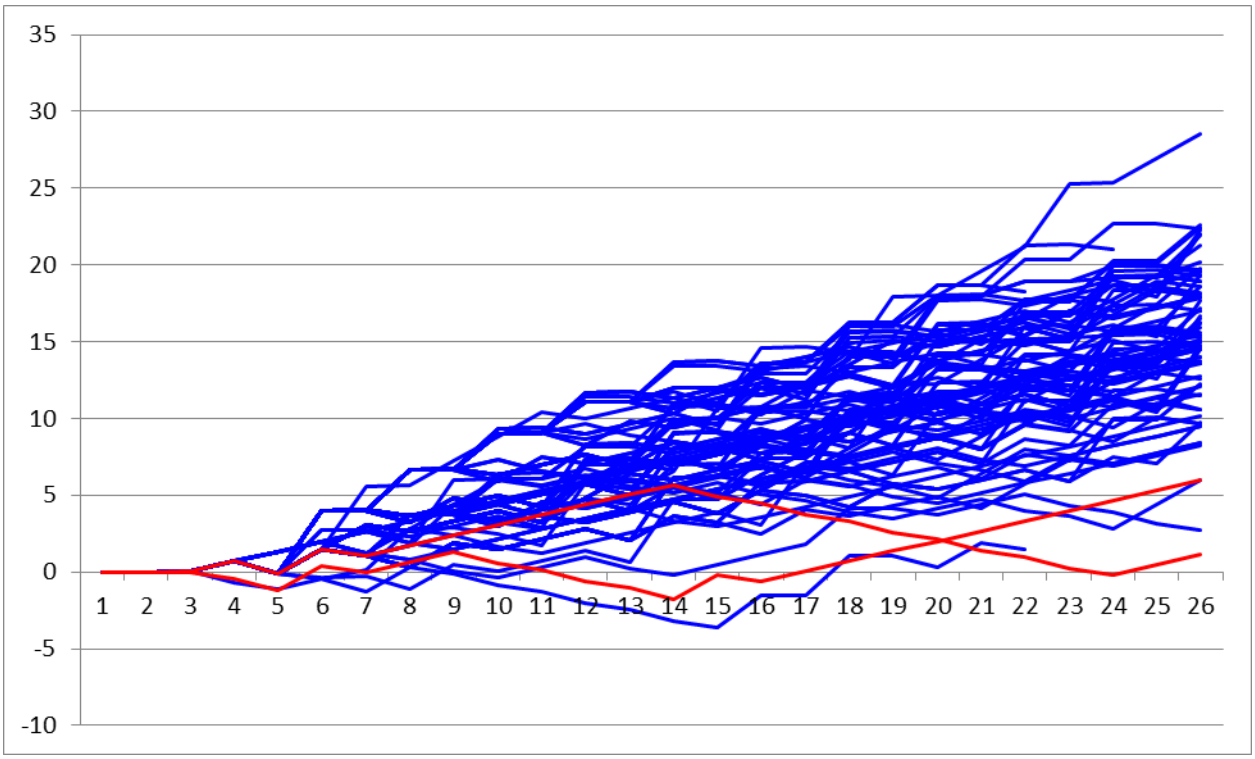

Figure B10. Order One, Two class, Weighted

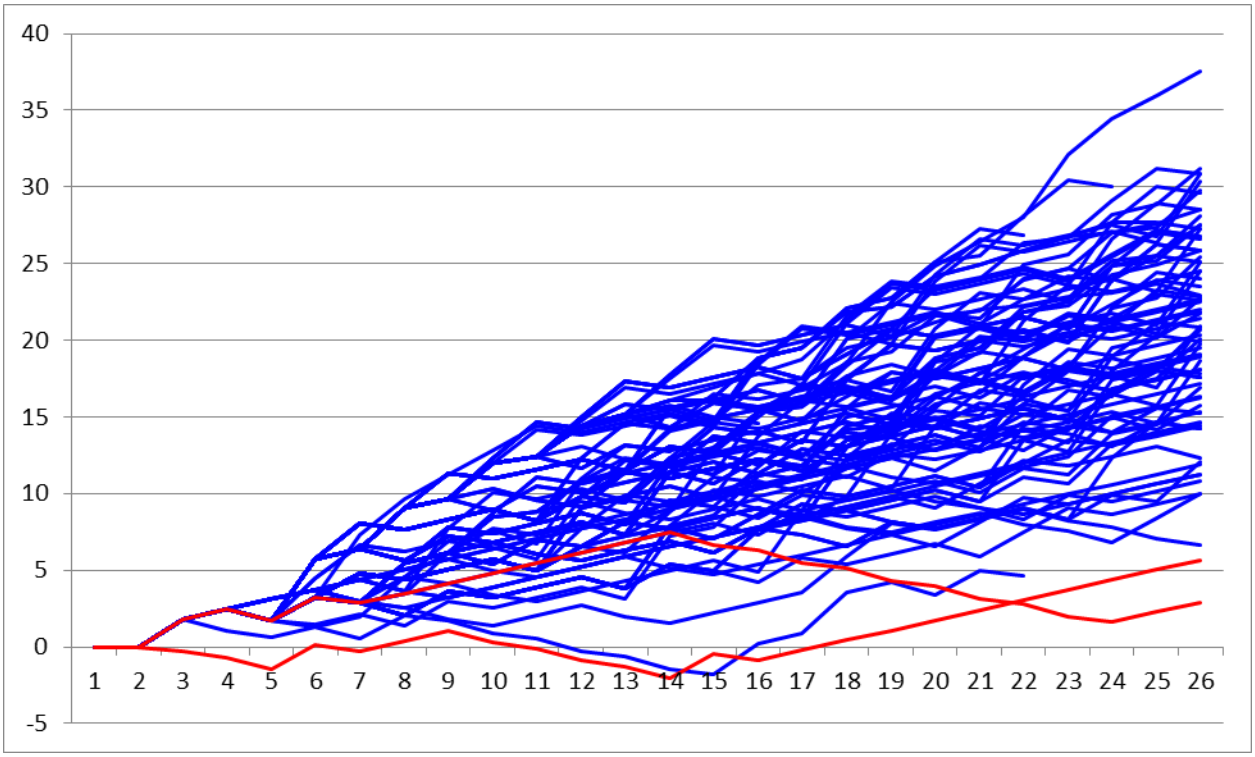

Figure B11. Order One, Multi class, Unweighted

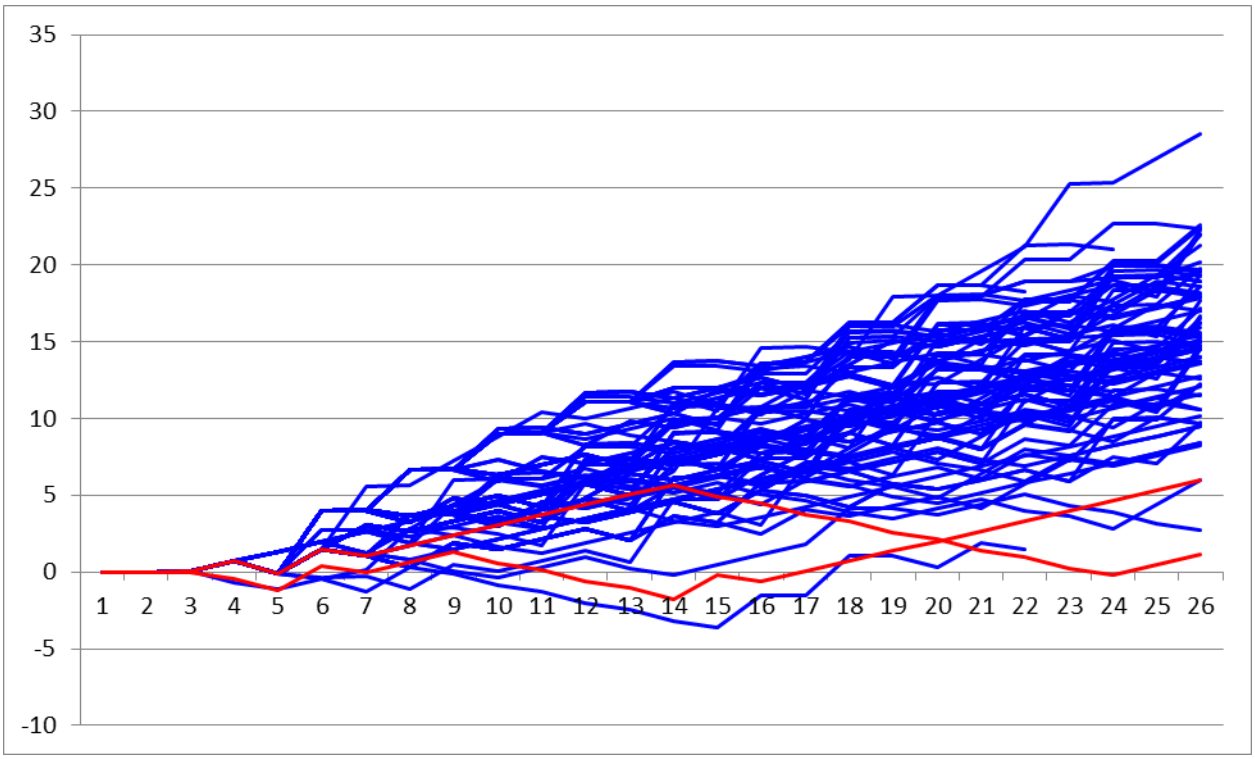

Figure B12. Order One, Multi class, Weighted

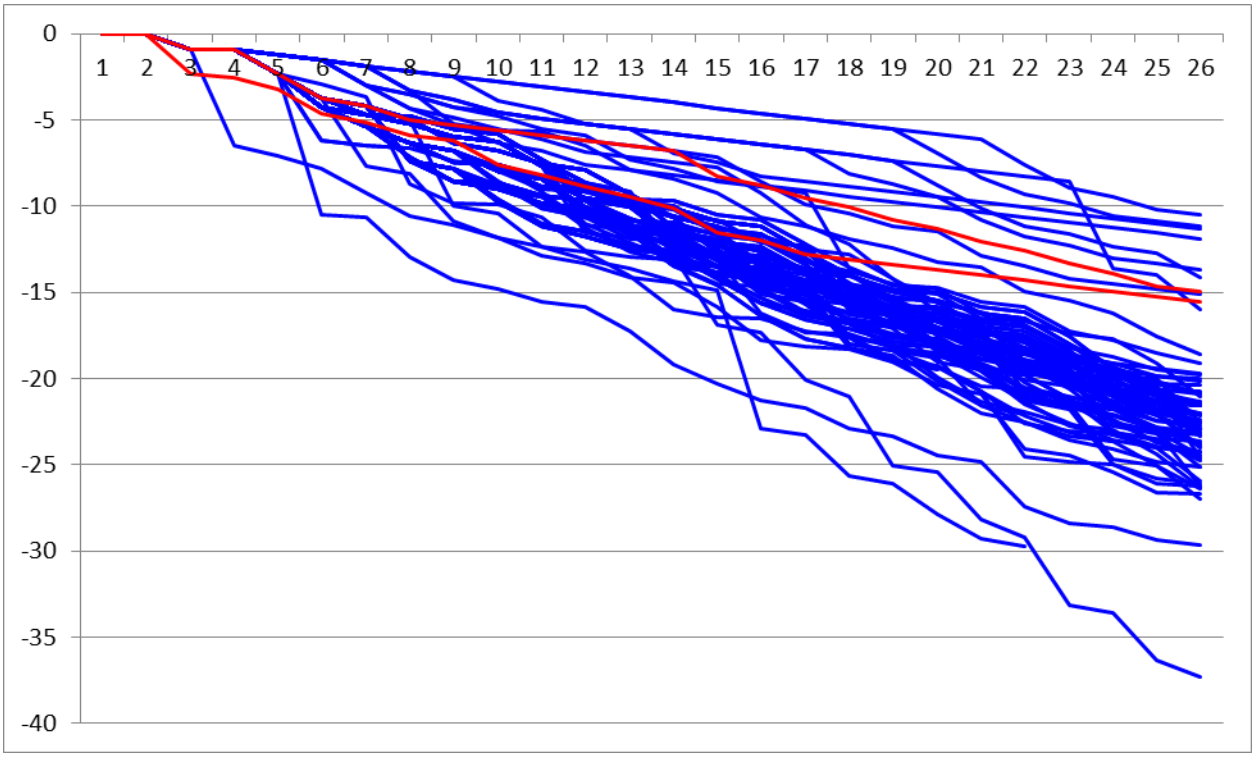

Figure B13. Order Two, One class, Unweighted

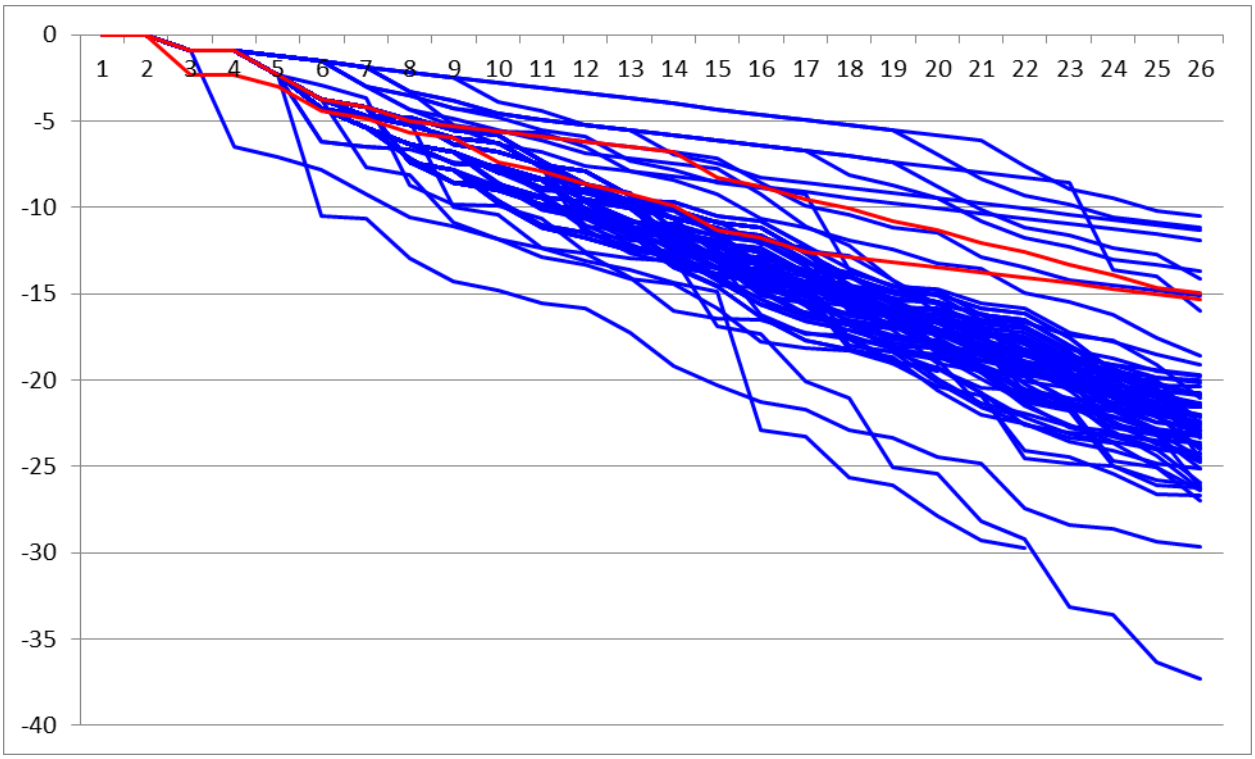

Figure B14. Order Two, One class, Weighted

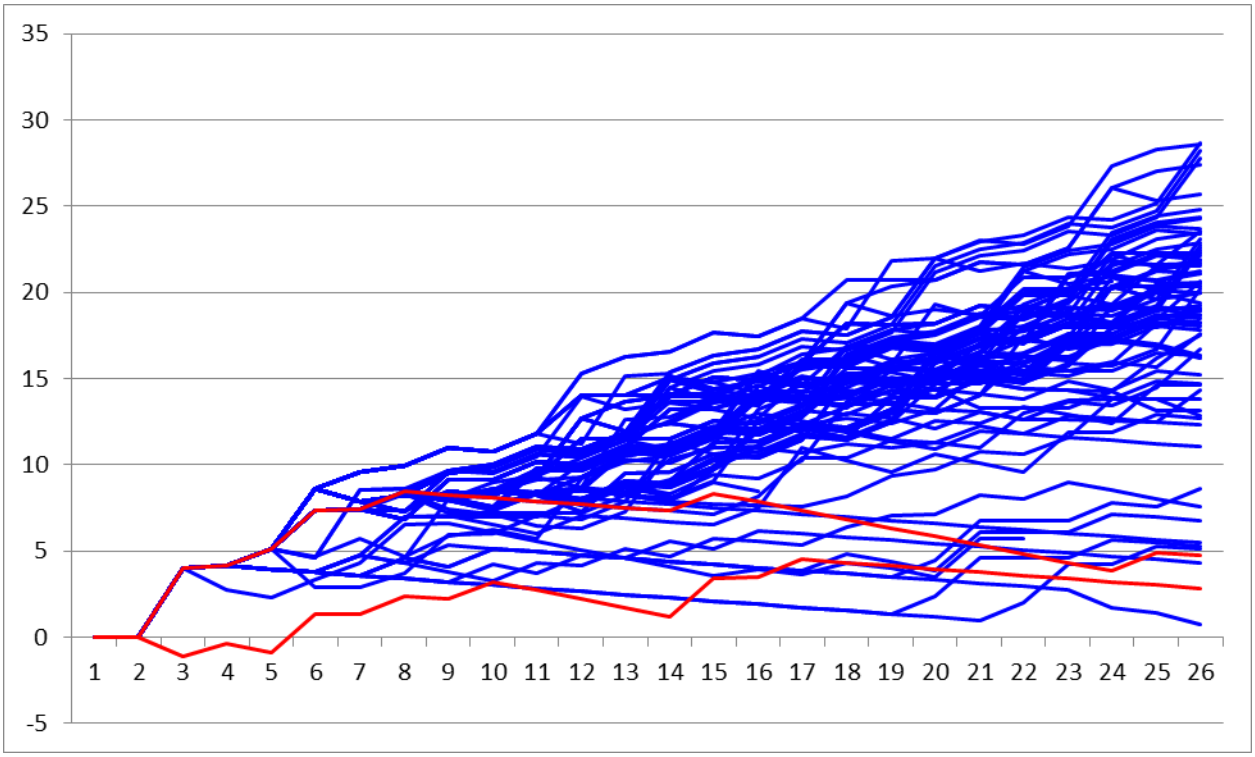

Figure B15. Order Two, Two class, Unweighted

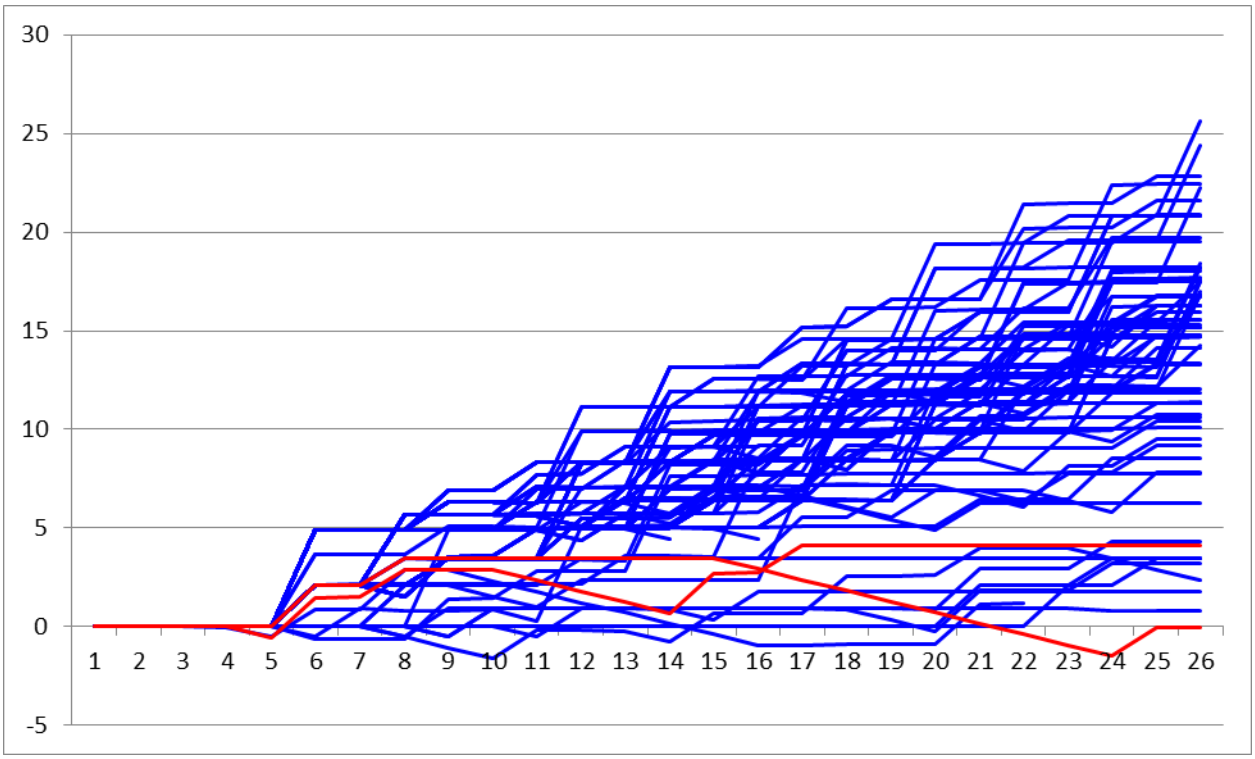

Figure B16. Order Two, Two class, Weighted

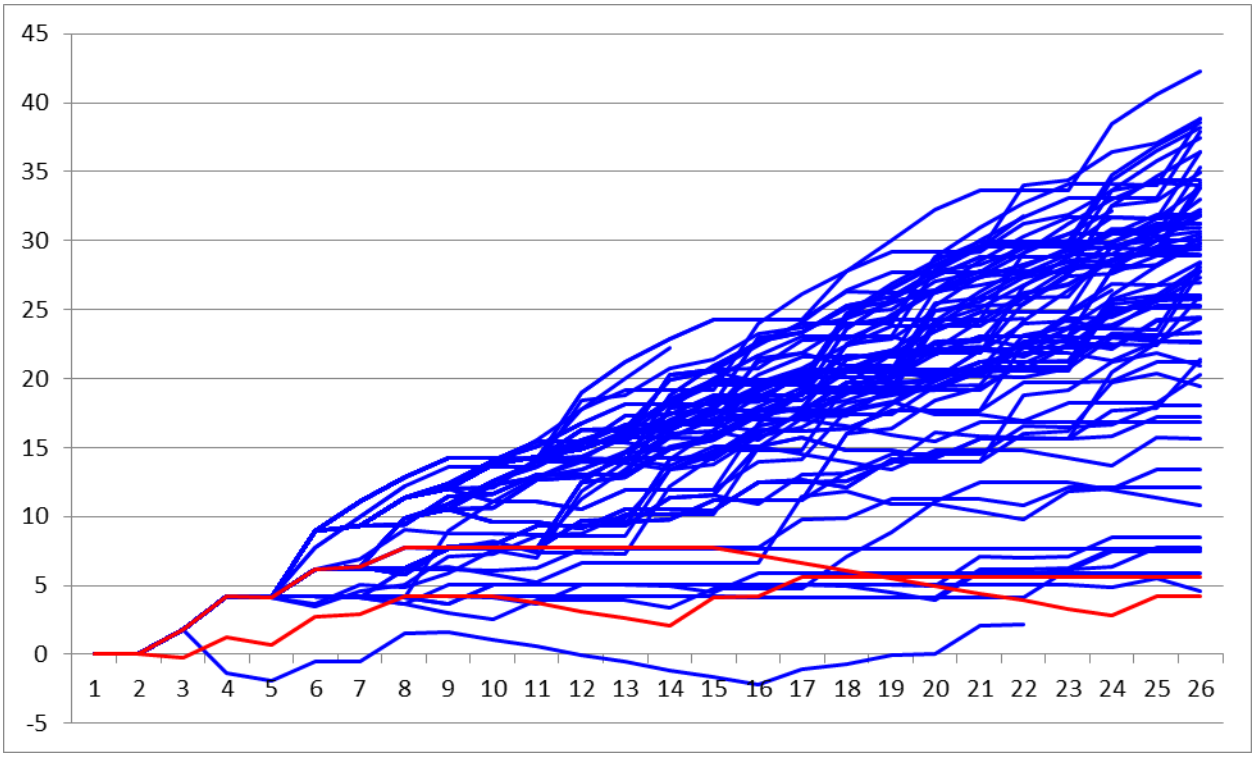

Figure B17. Order Two, Multi class, Unweighted

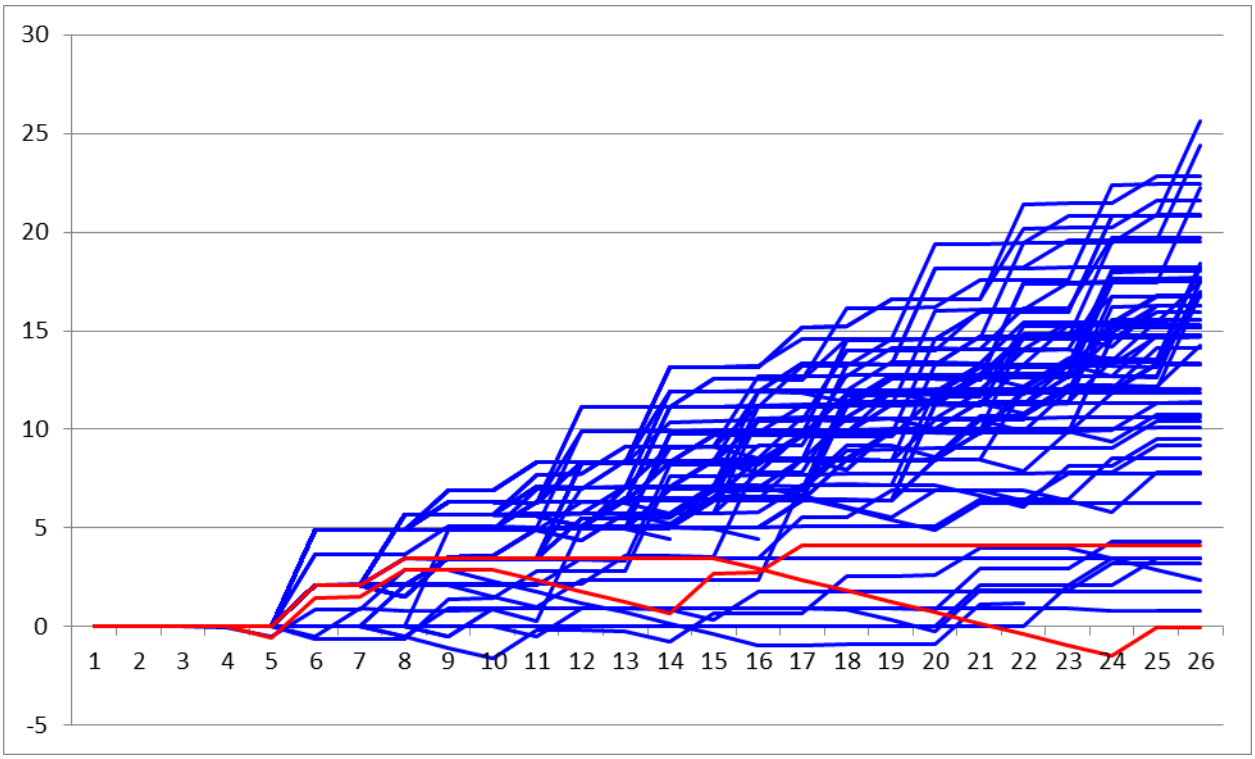

Figure B18. Order Two, Multi class, Weighted

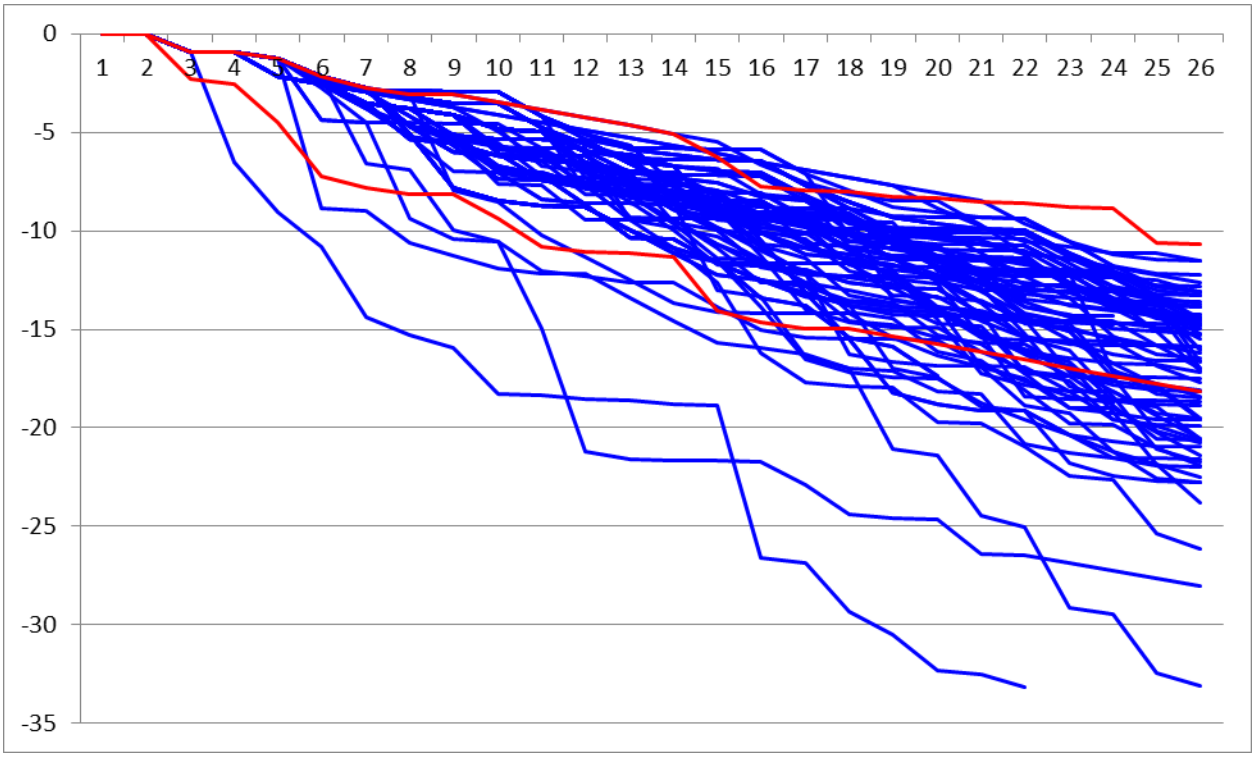

Figure B19. Order Three, One class, Unweighted

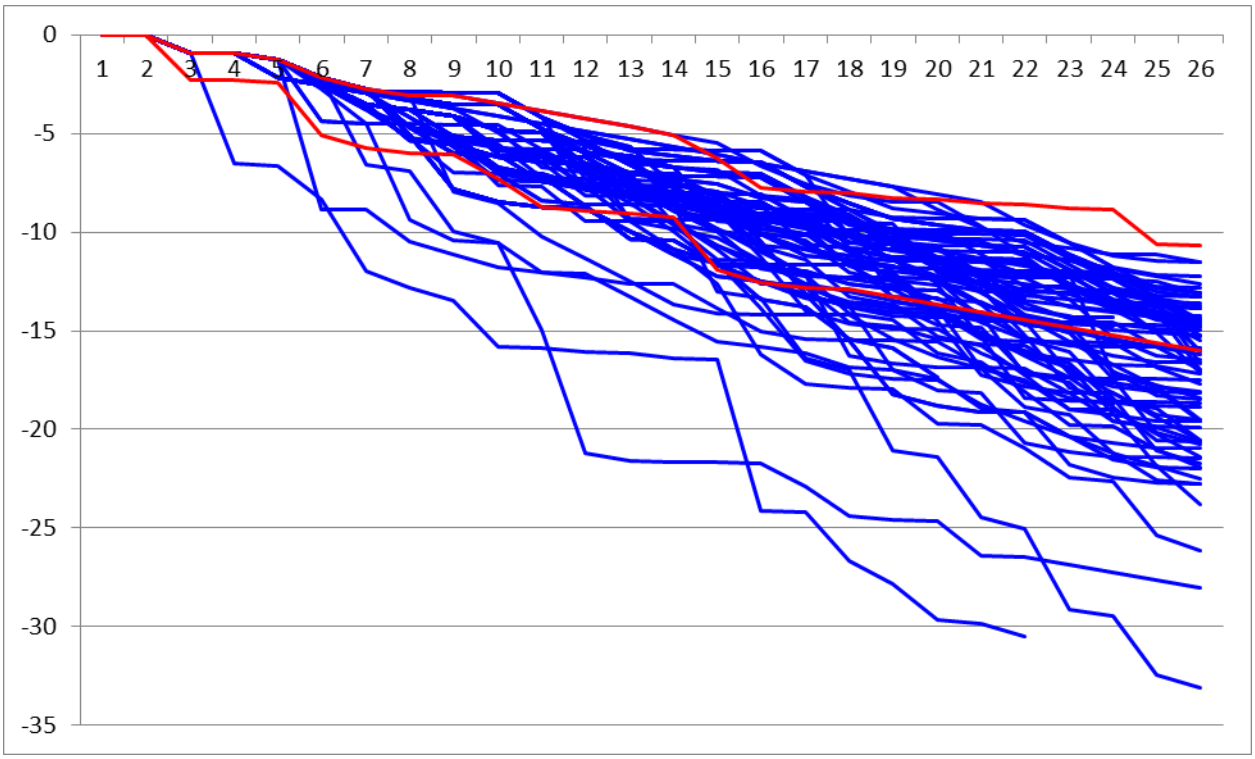

Figure B20. Order Three, One class, Weighted

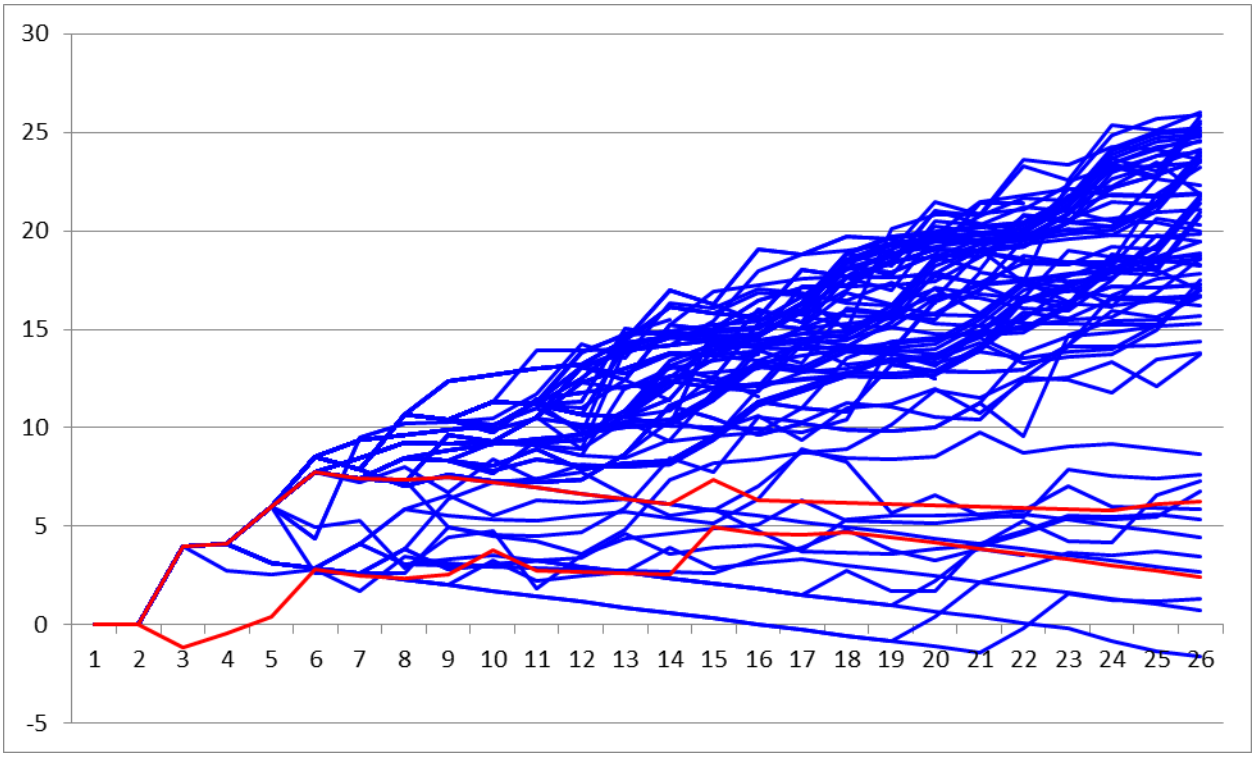

Figure B21. Order Three, Two class, Unweighted

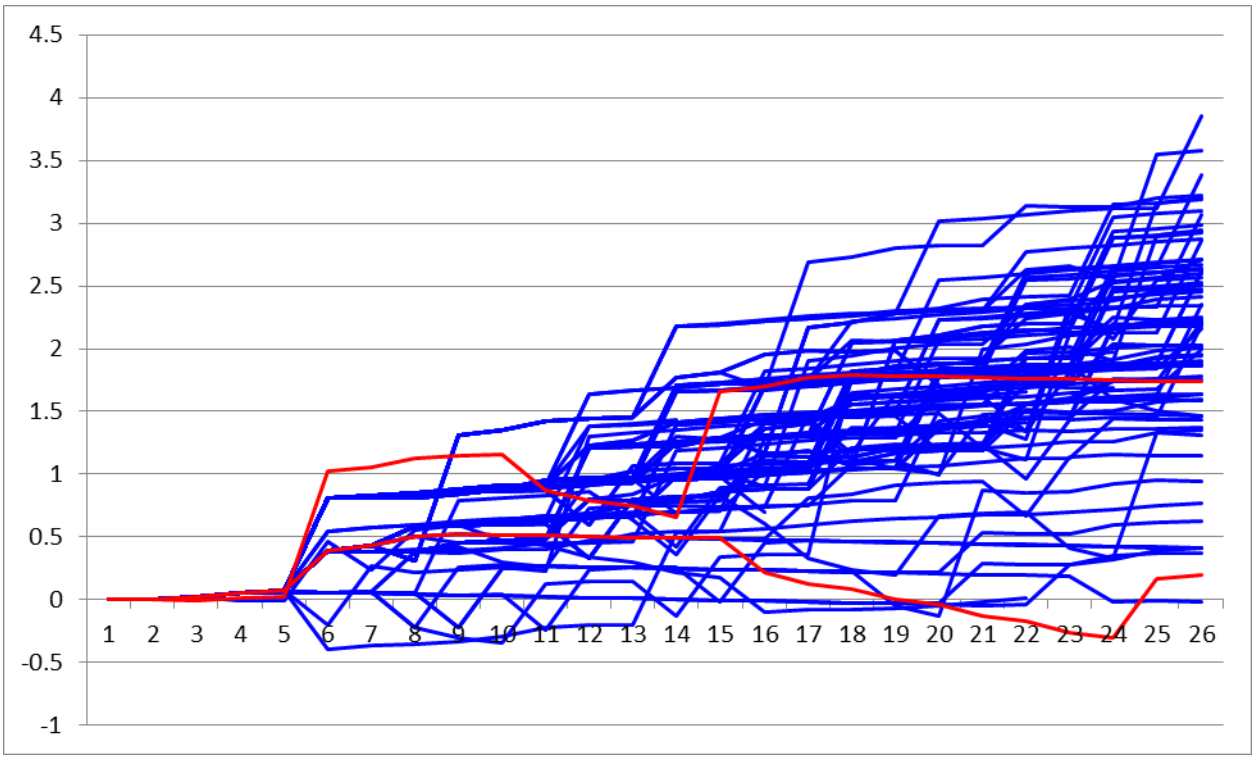

Figure B22. Order Three, Two class, Weighted

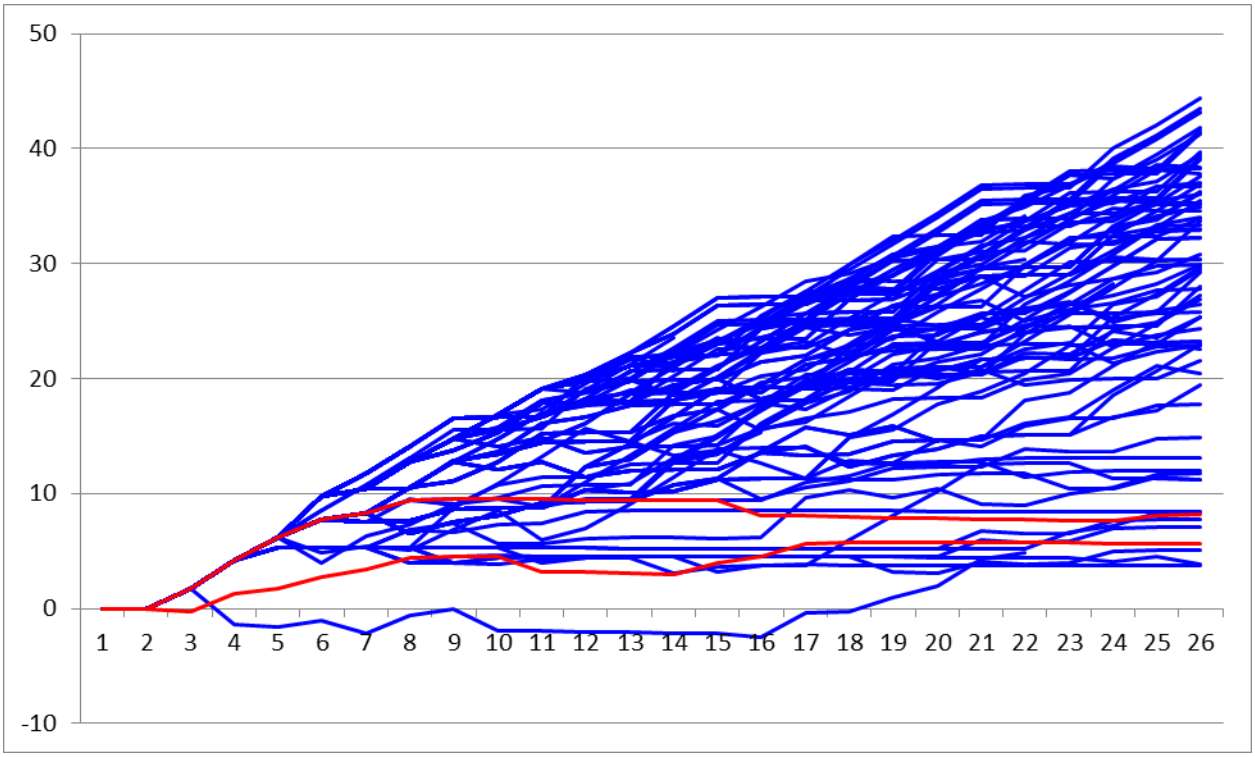

Figure B23. Order Three, Multi class, Unweighted

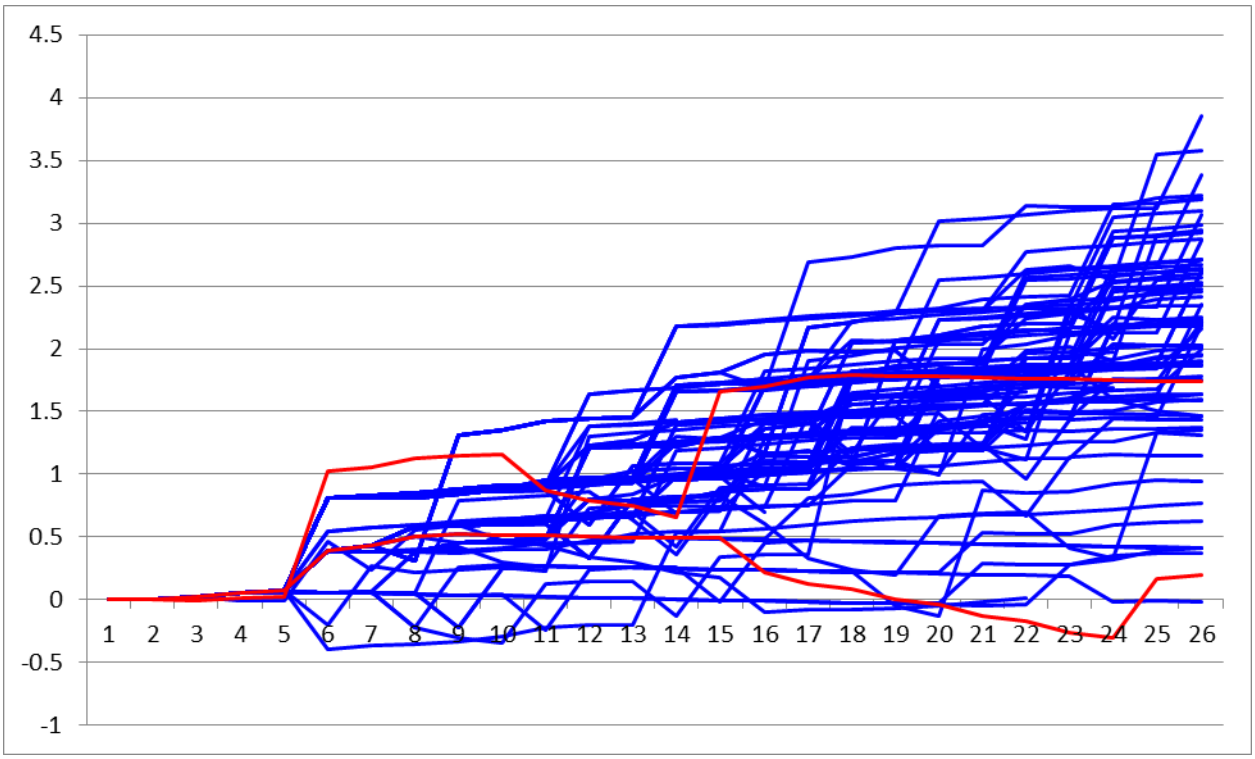

Figure B24. Order Three, Multi class, Weighted

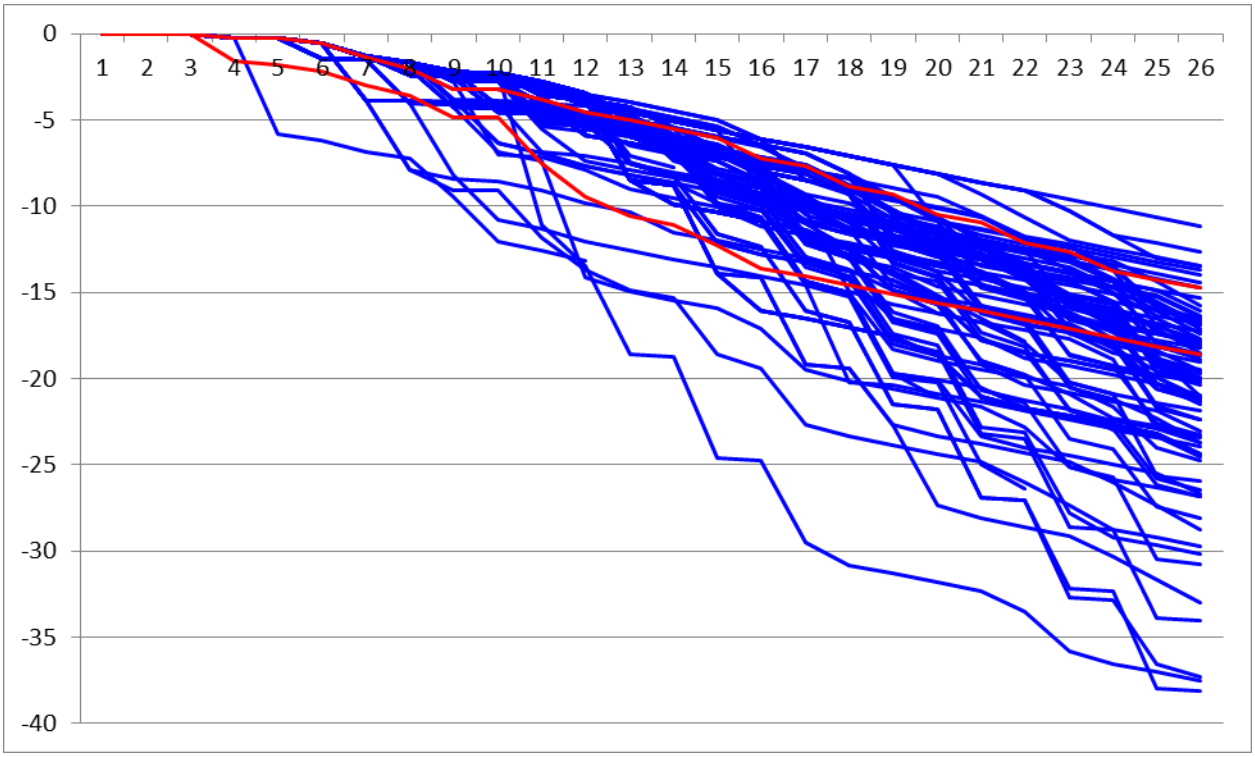

Figure B25. Order One, One class, Unweighted, Pseudo Timing

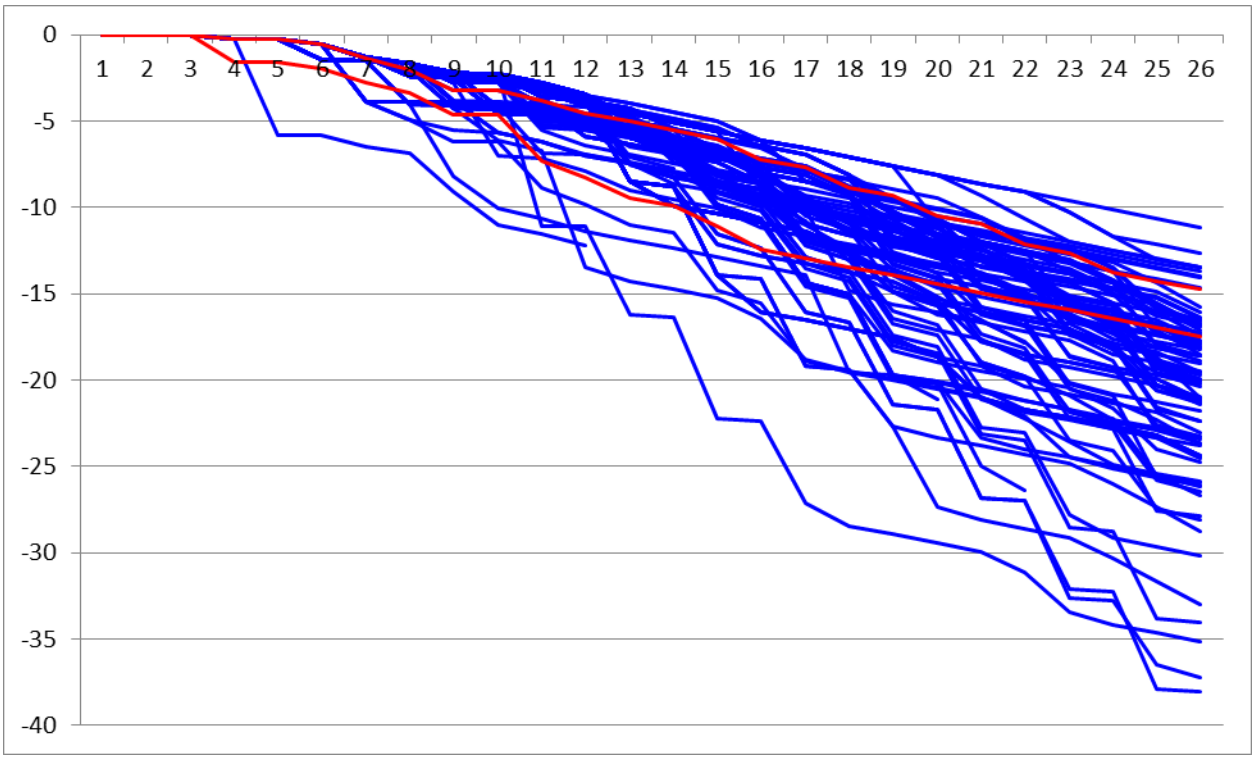

Figure B26. Order One, One class, Weighted, Pseudo Timing

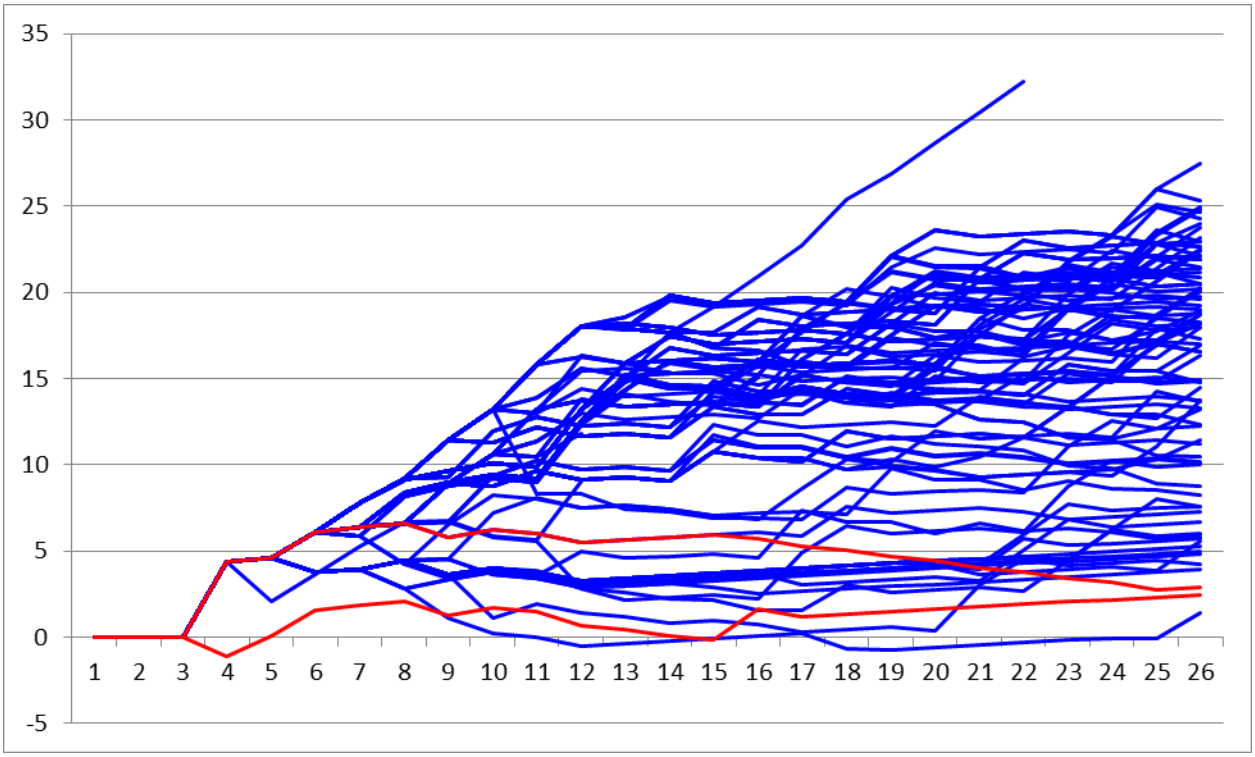

Figure B27. Order One, Two class, Unweighted, Pseudo Timing

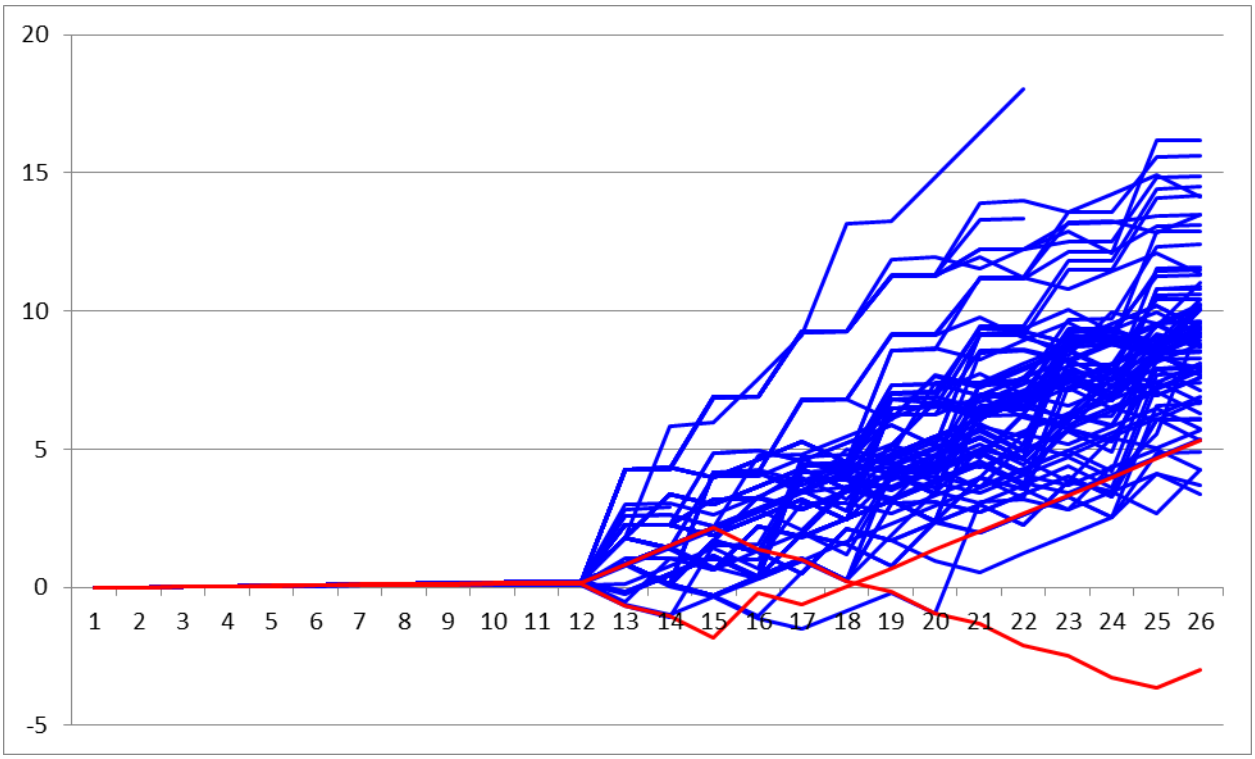

Figure B28. Order One, Two class, Weighted, Pseudo Timing

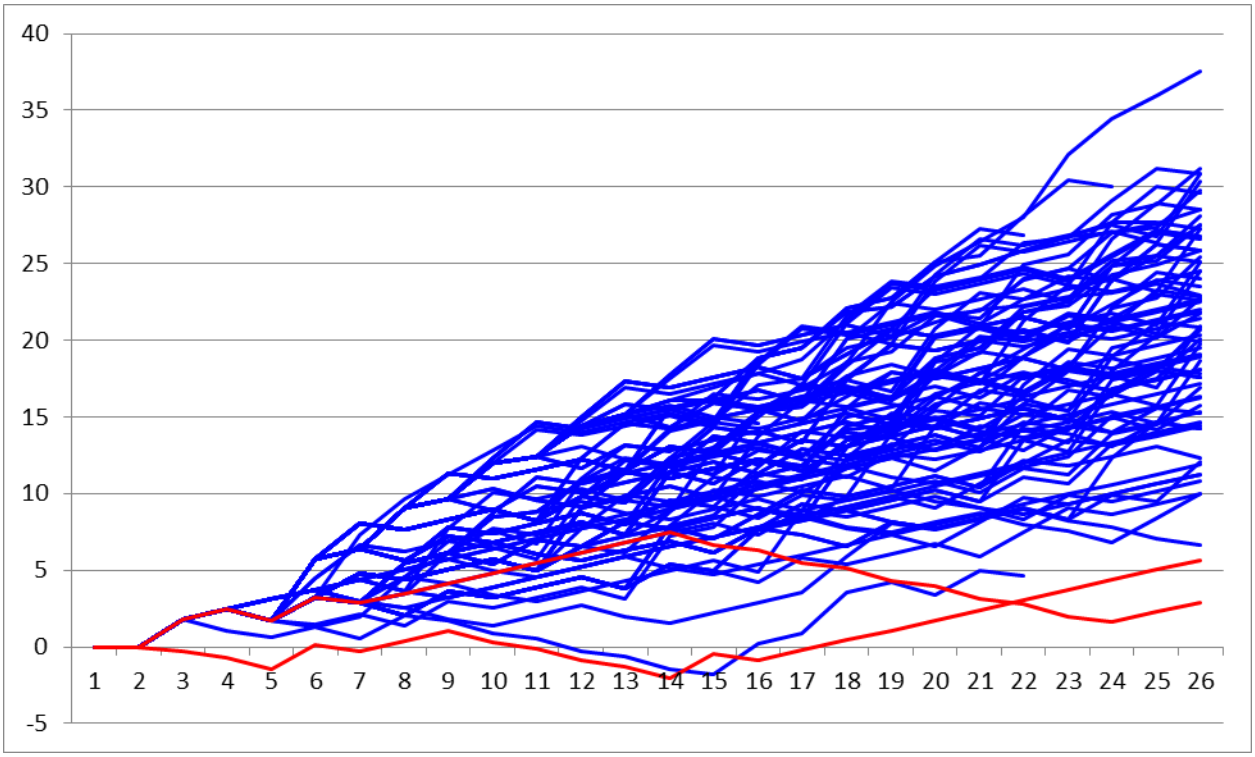

Figure B29. Order One, Multi class, Unweighted, Pseudo Timing

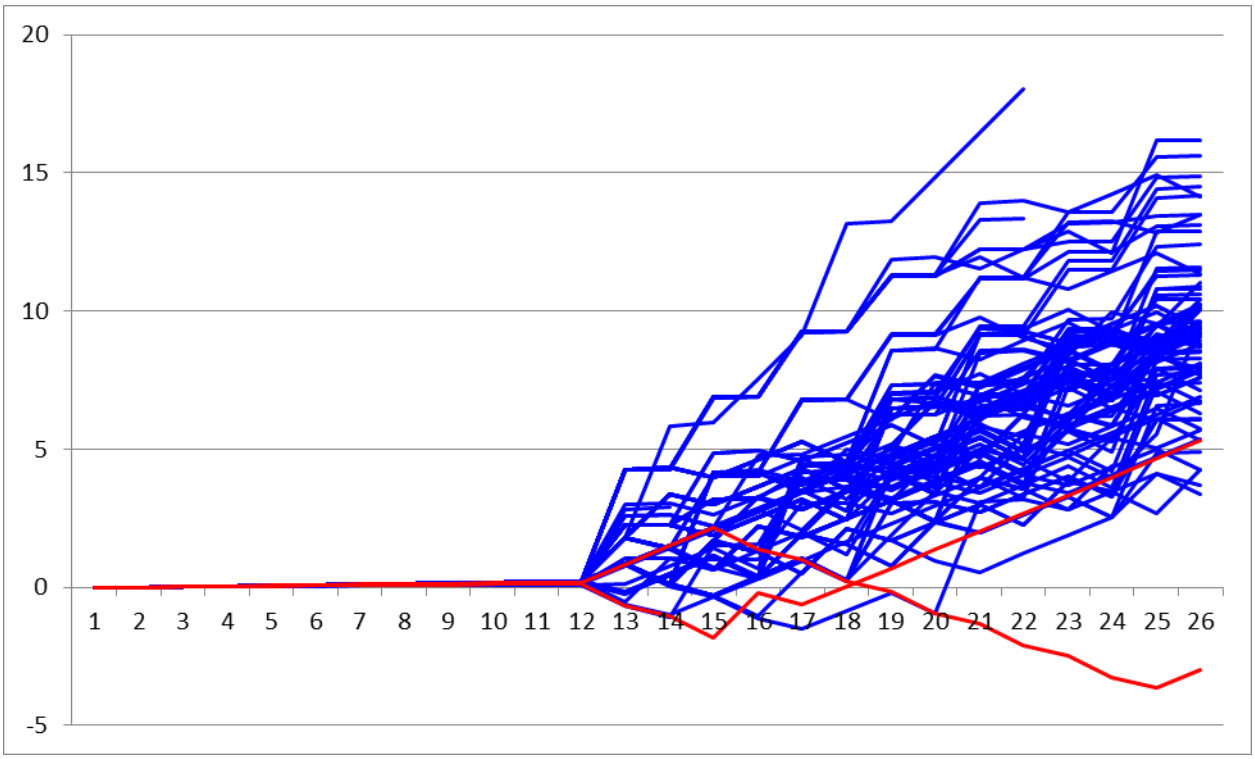

Figure B30. Order One, Multi class, Weighted, Pseudo Timing

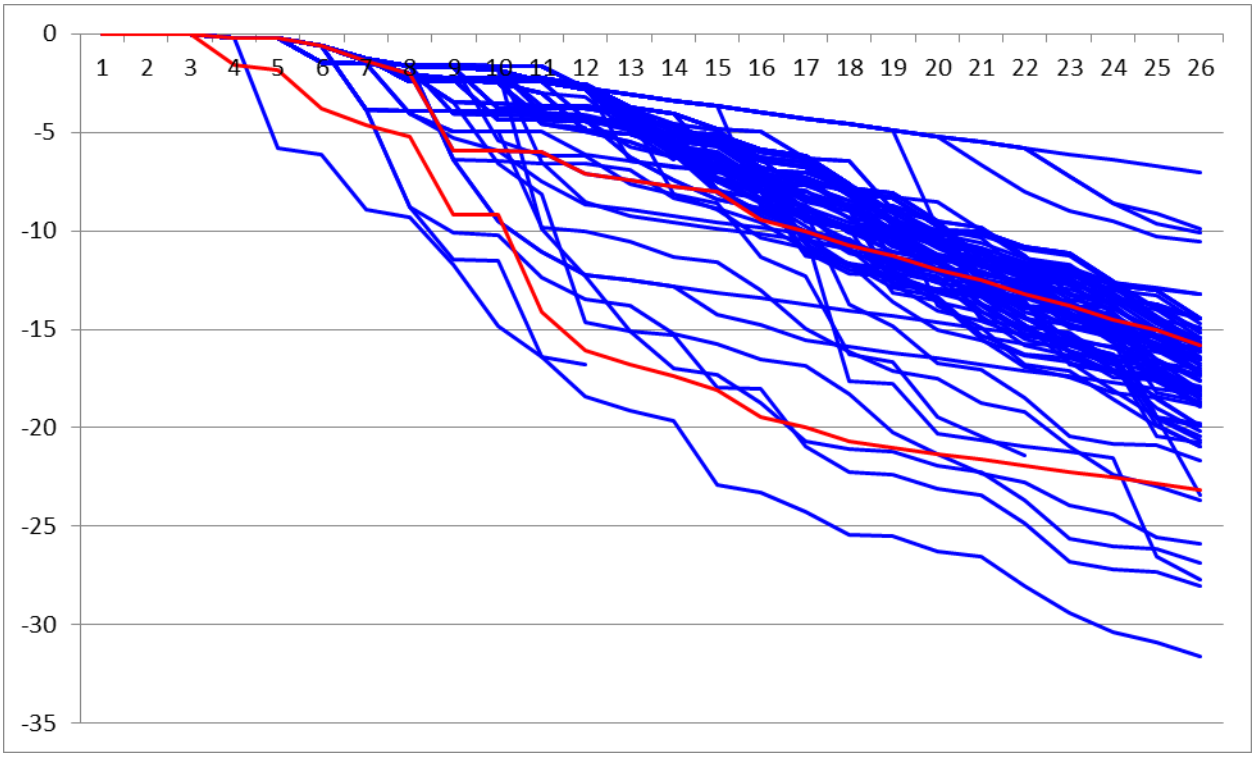

Figure B31. Order Two, One class, Unweighted, Pseudo Timing

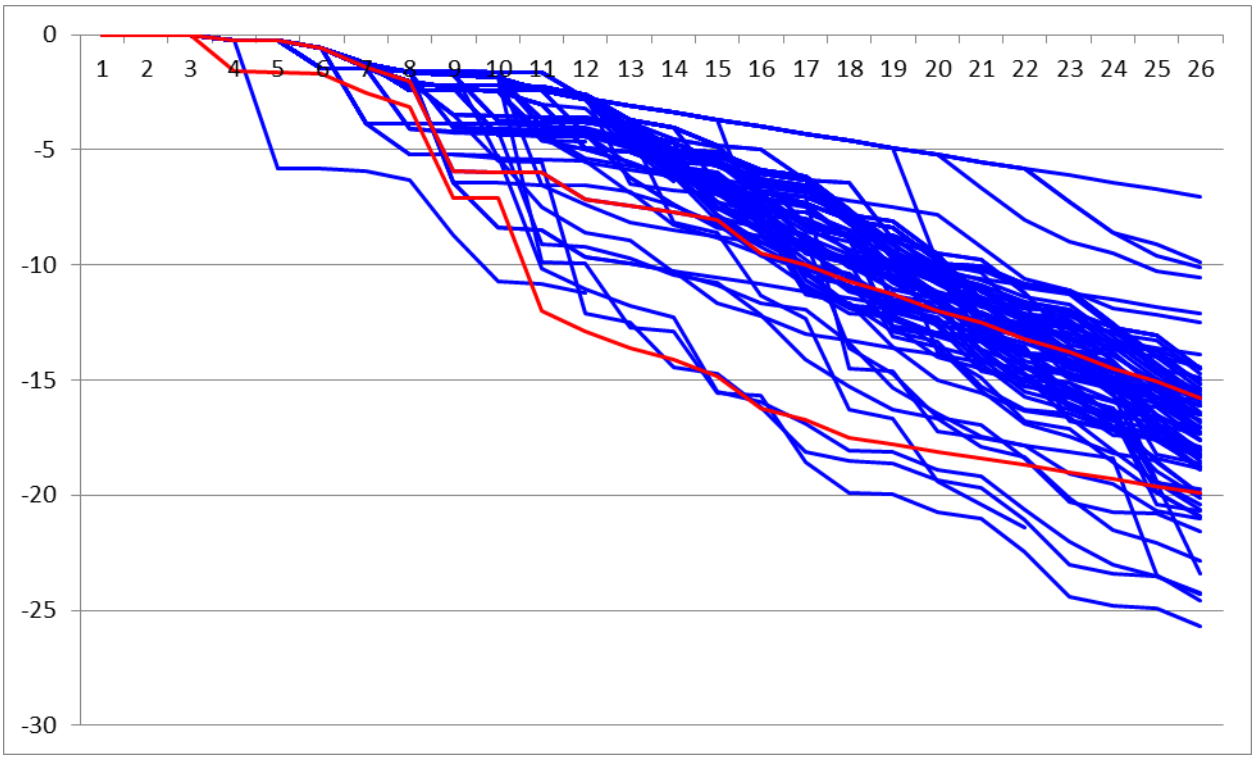

Figure B32. Order Two, One class, Weighted, Pseudo Timing

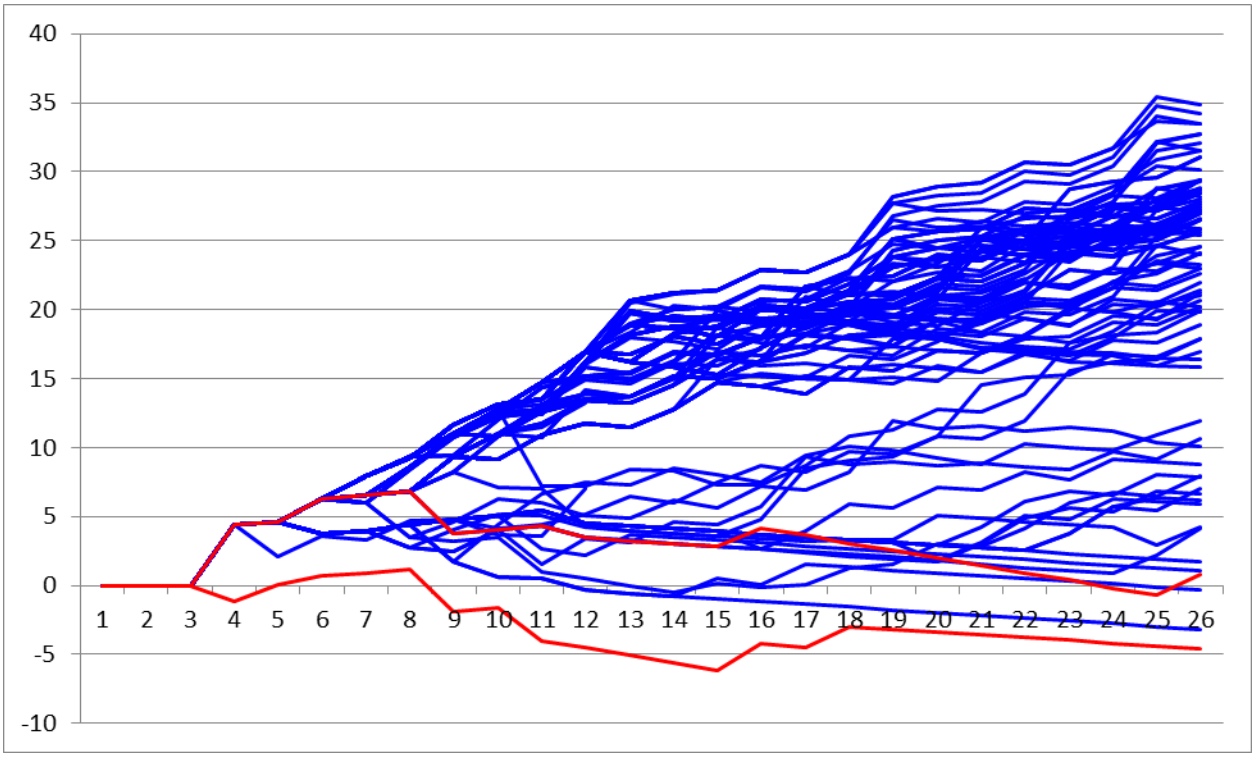

Figure B33. Order Two, Two class, Unweighted, Pseudo Timing

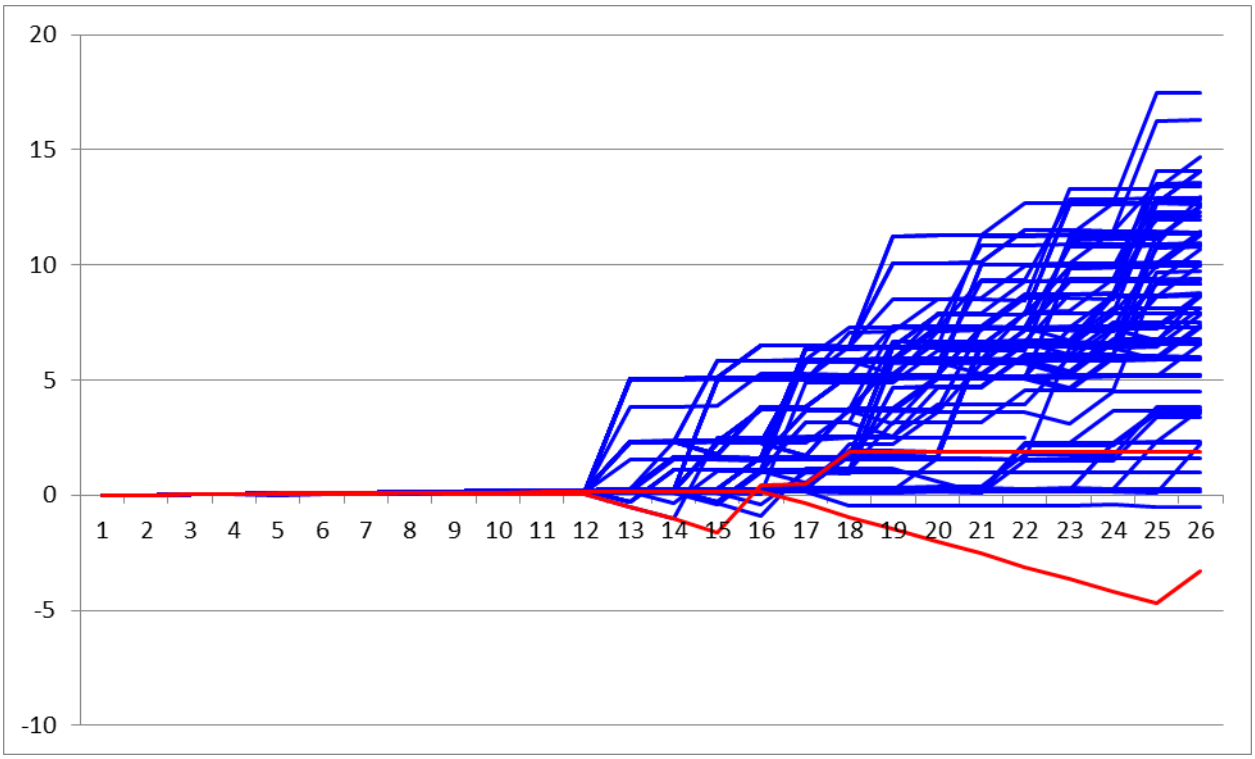

Figure B34. Order Two, Two class, Weighted, Pseudo Timing

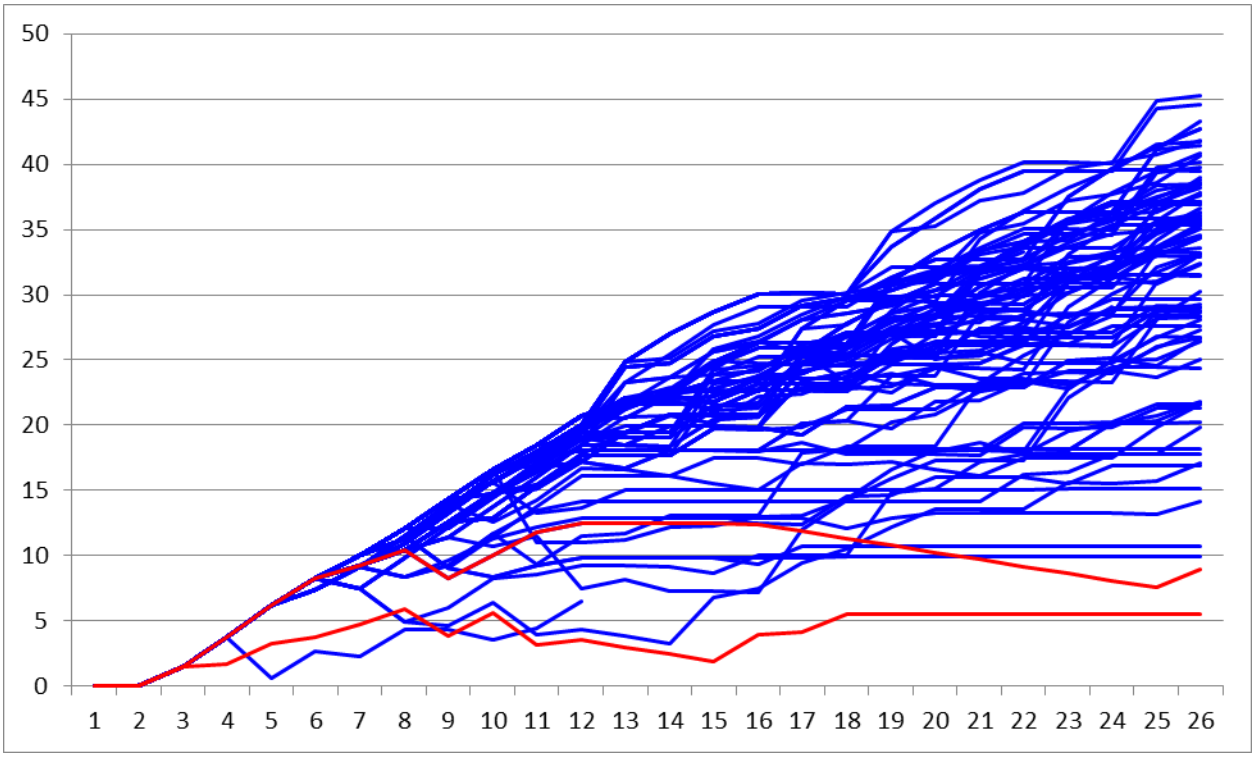

Figure B35. Order Two, Multi class, Unweighted, Pseudo Timing

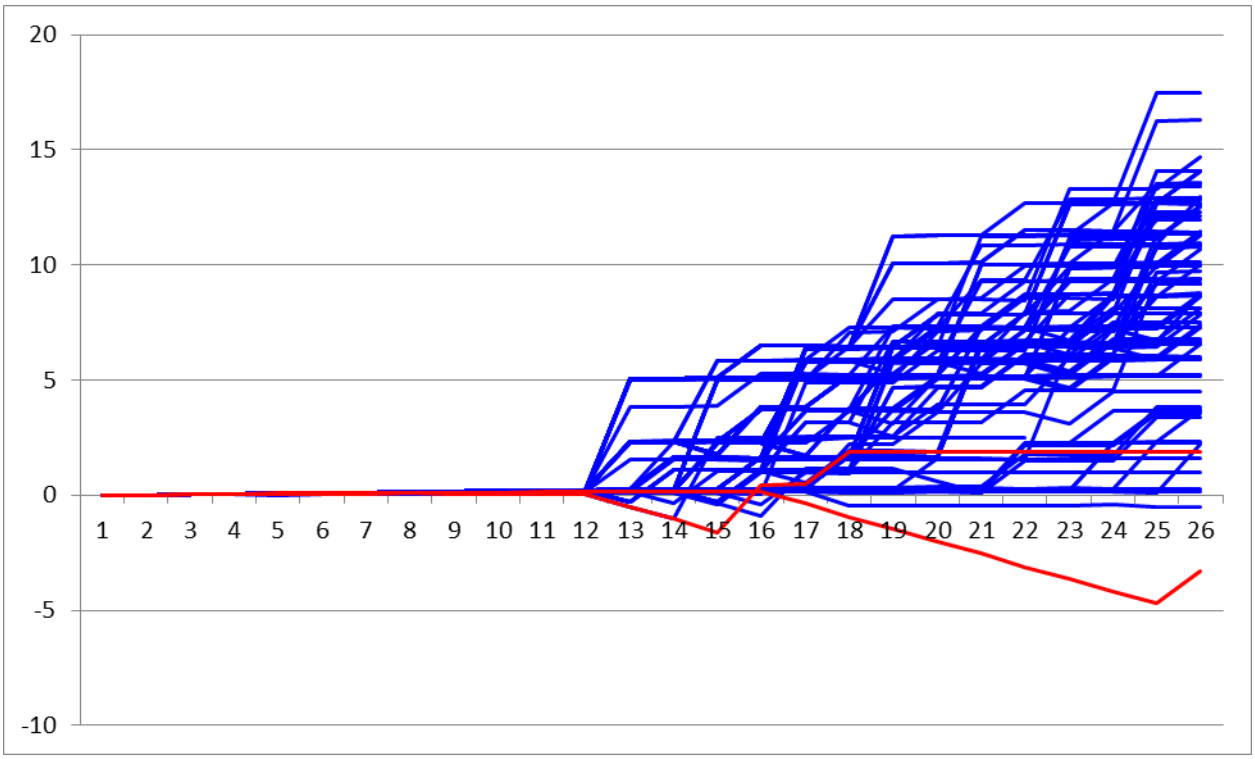

Figure B36. Order Two, Multi class, Weighted, Pseudo Timing

APPENDIX C. PORTSWEEP ATTACK DATASET SPRT GRAPHS

Portseep is a set of attacks from the 1998 DARPA intrusion detection evaluation dataset. Included for reference is an organized listing of SPRT graphs calculated under explained Conditions.

DARPA Description: Surveillance sweep through many ports to determine which services are supported on a single host.

Data Counts

Table C1. Data Counts

	Training	Testing
Flows	4	1
Transitions	5406	66
Packets	6193	283

Training Dataset

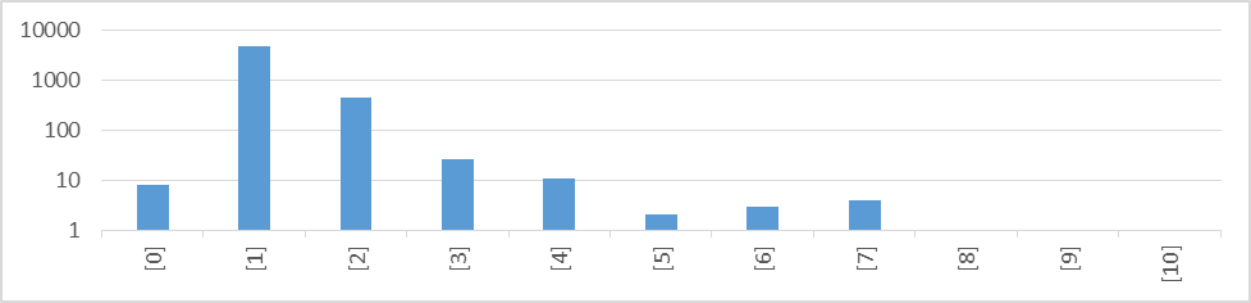

Figure C1. Portsweep Order 1 Training Dataset Histogram

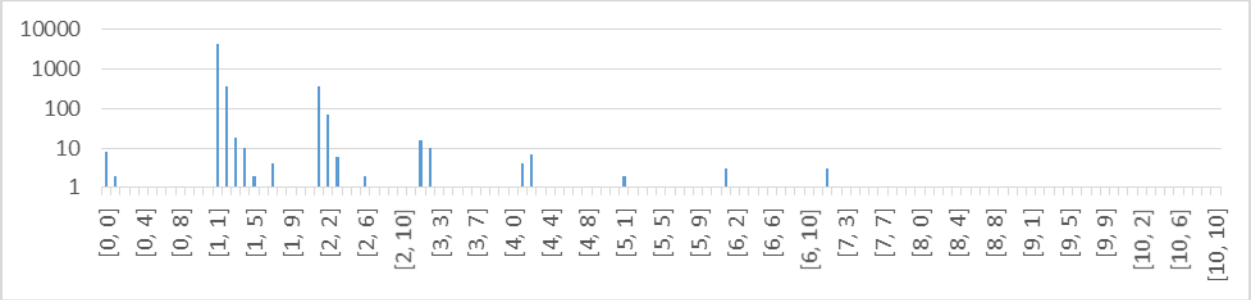

Figure C2. Portsweep Order 3 Training Dataset Histogram

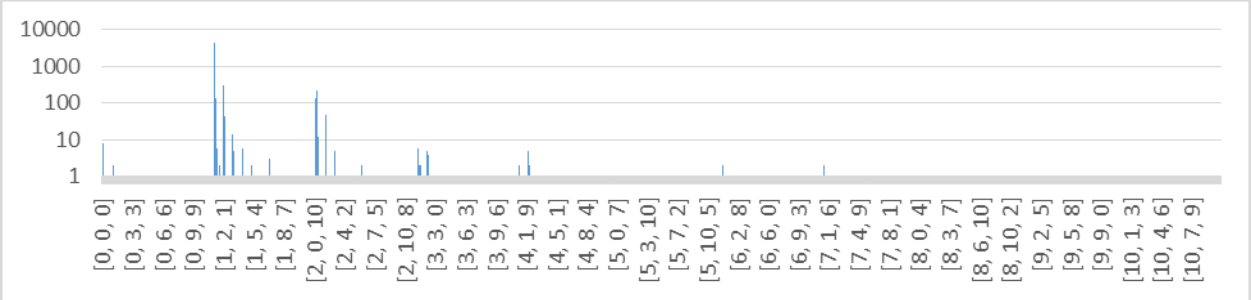

Figure C3. Portsweep Order 3 Training Dataset Histogram

Testing Data Histograms

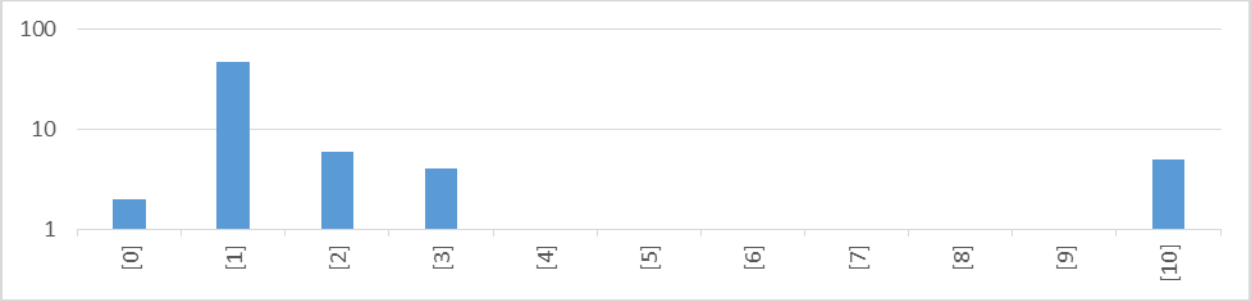

Figure C4. Portsweep Order 1 Testing Dataset Histogram

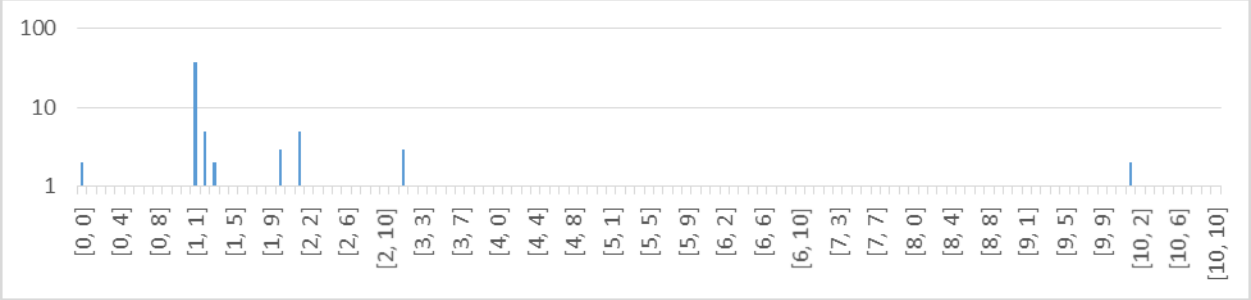

Figure C5. Portsweep Order 2 Testing Dataset Histogram

Figure C6. Portsweep Order 3 Testing Dataset Histogram

SPRT Graphs

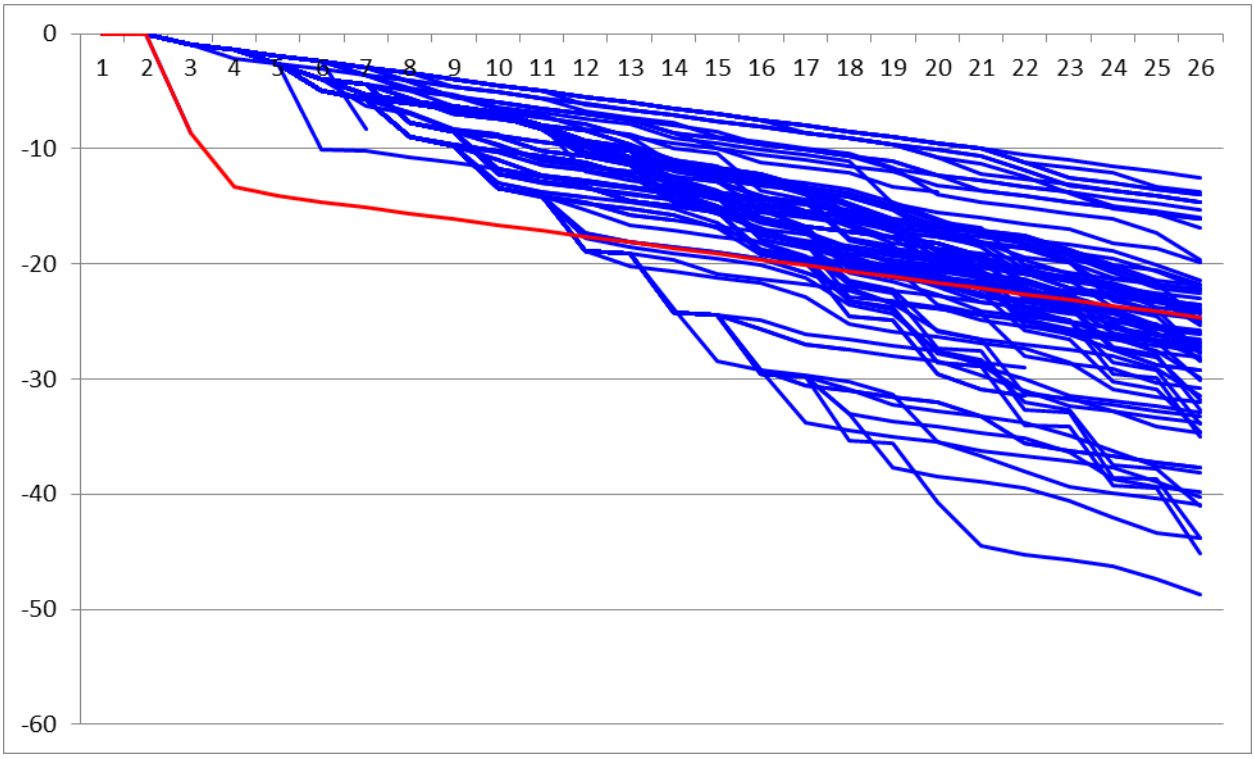

Figure C7. Order One, One class, Unweighted

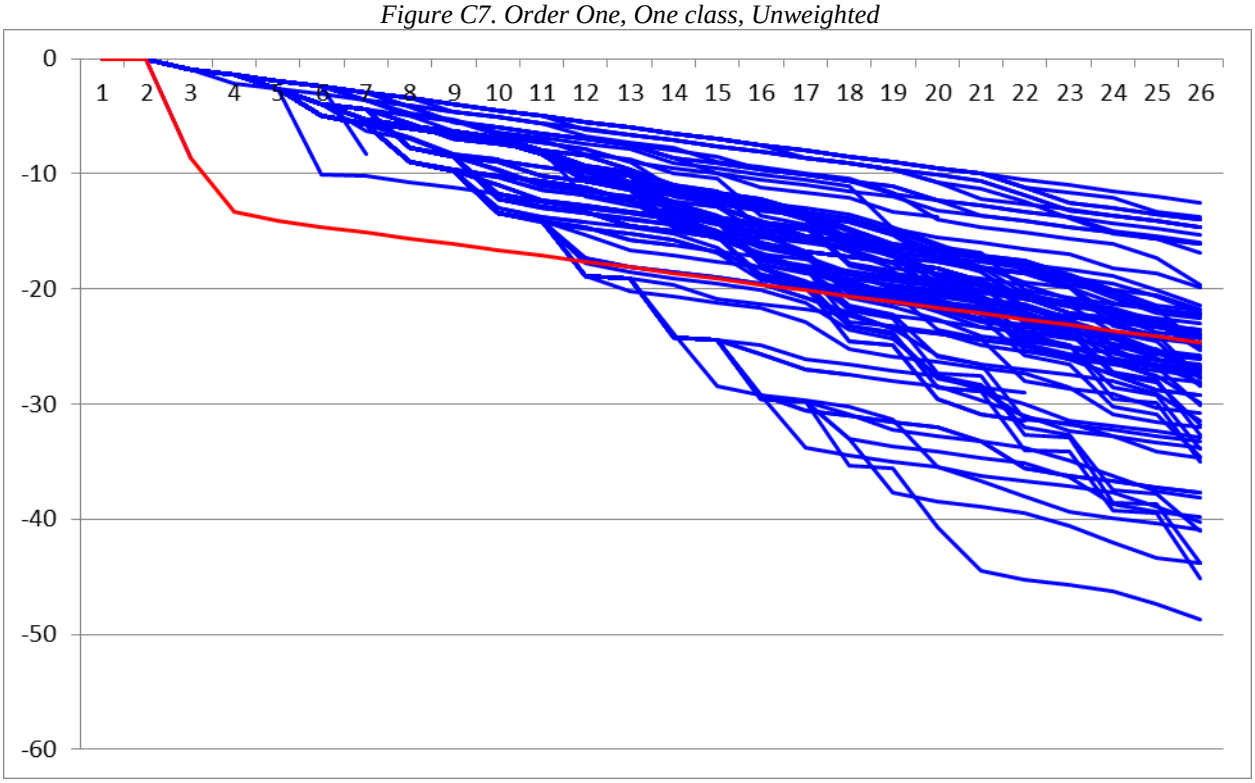

Figure C8.Order One, One class, Weighted

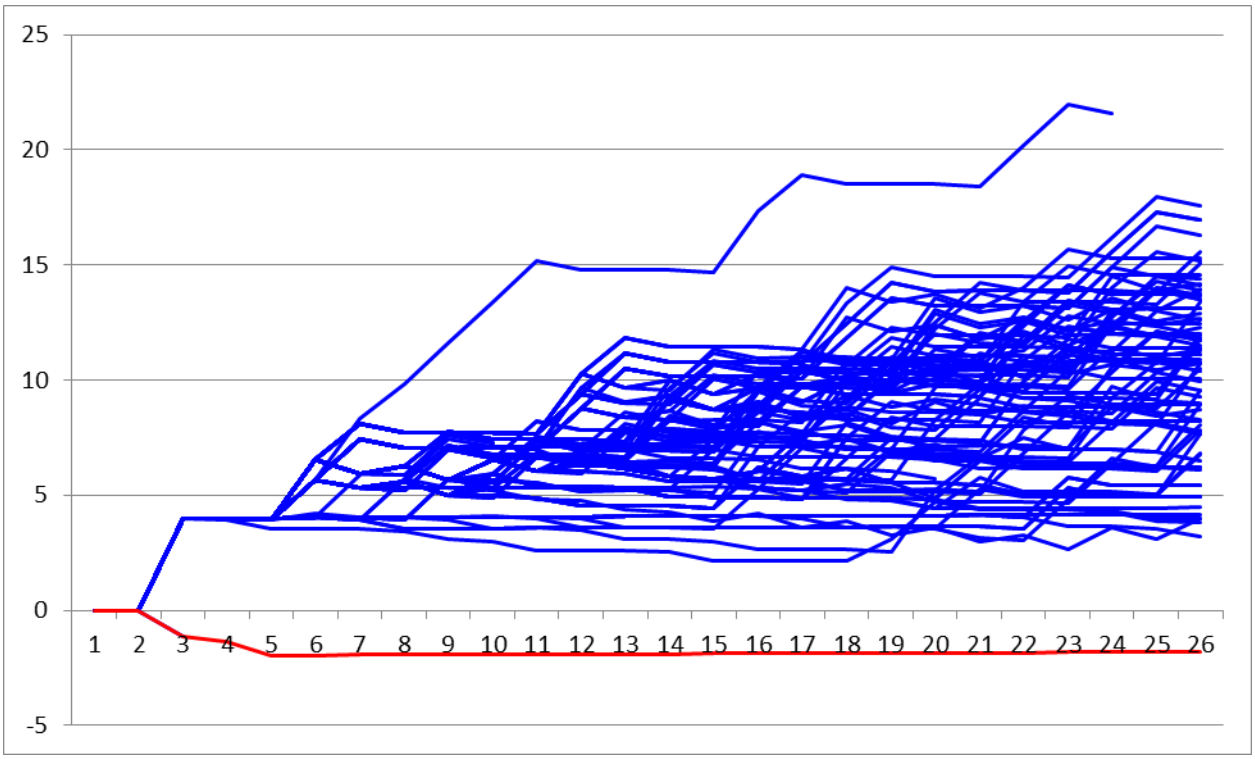

Figure C9. Order One, Two class, Unweighted

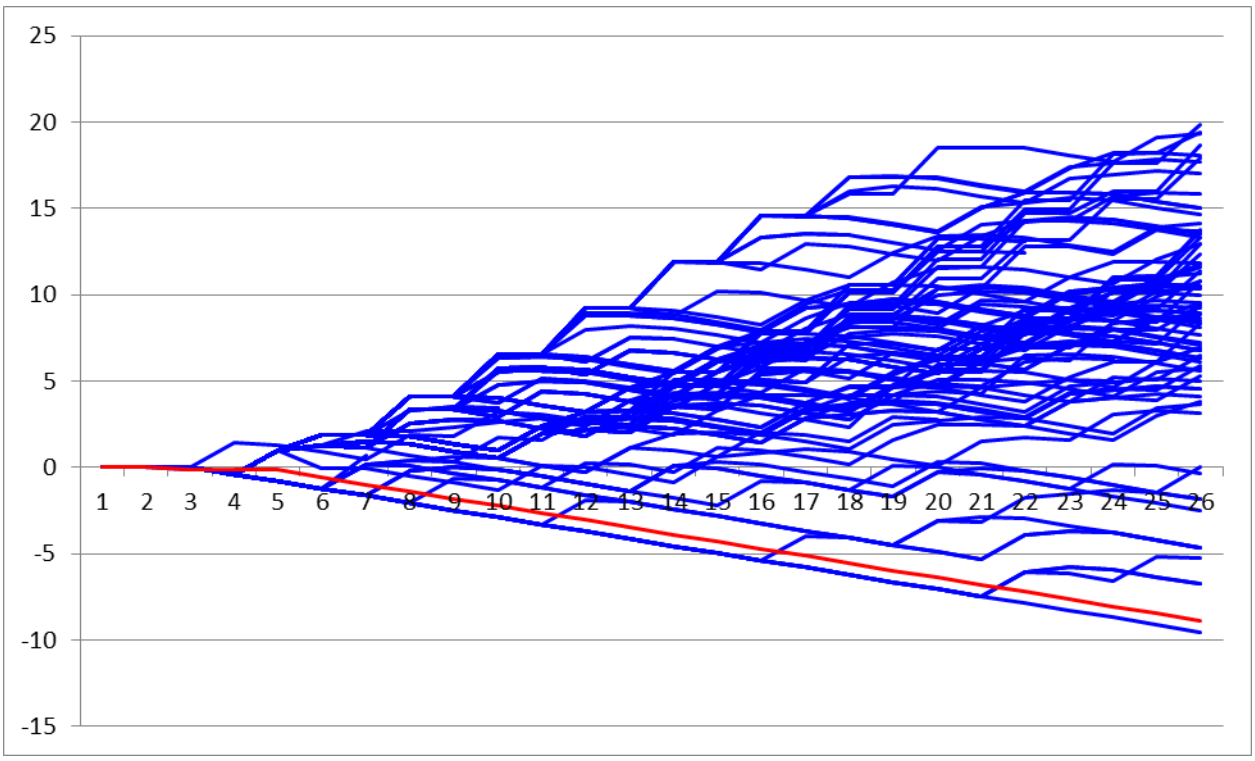

Figure C10. Order One, Two class, Weighted

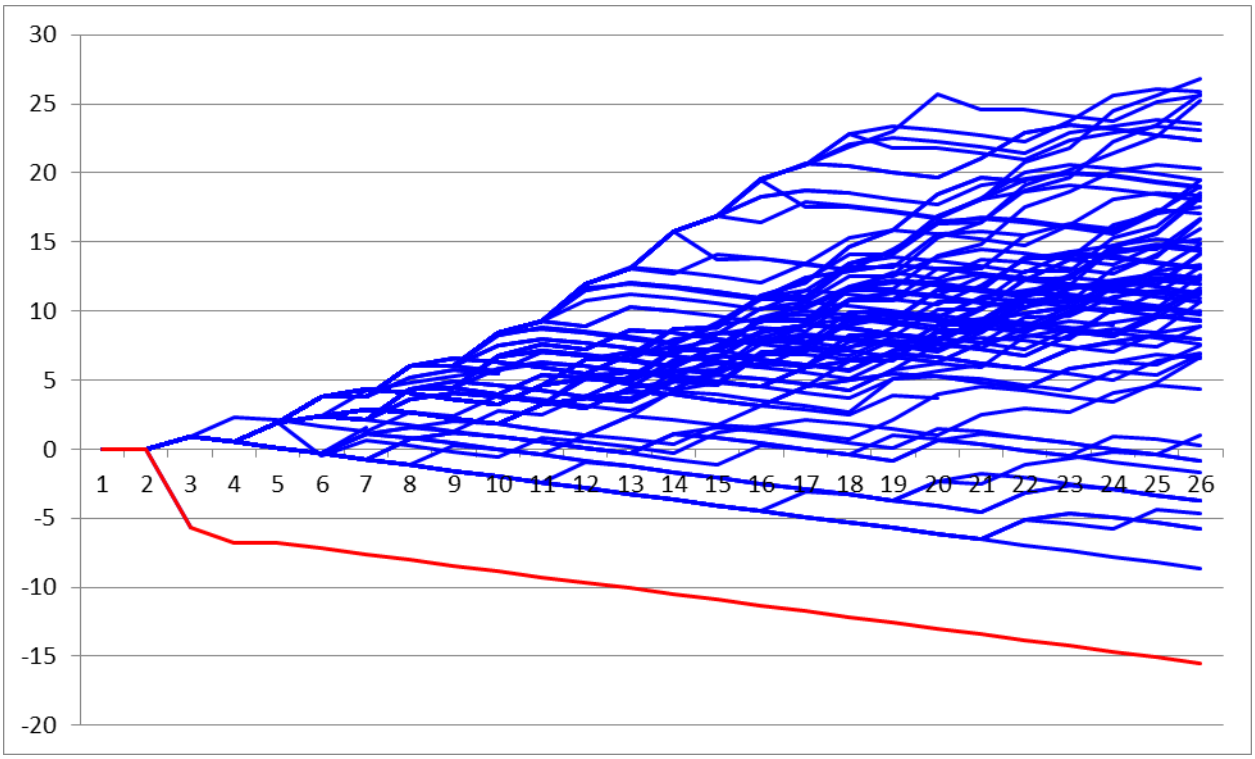

Figure C11. Order One, Multi class, Unweighted

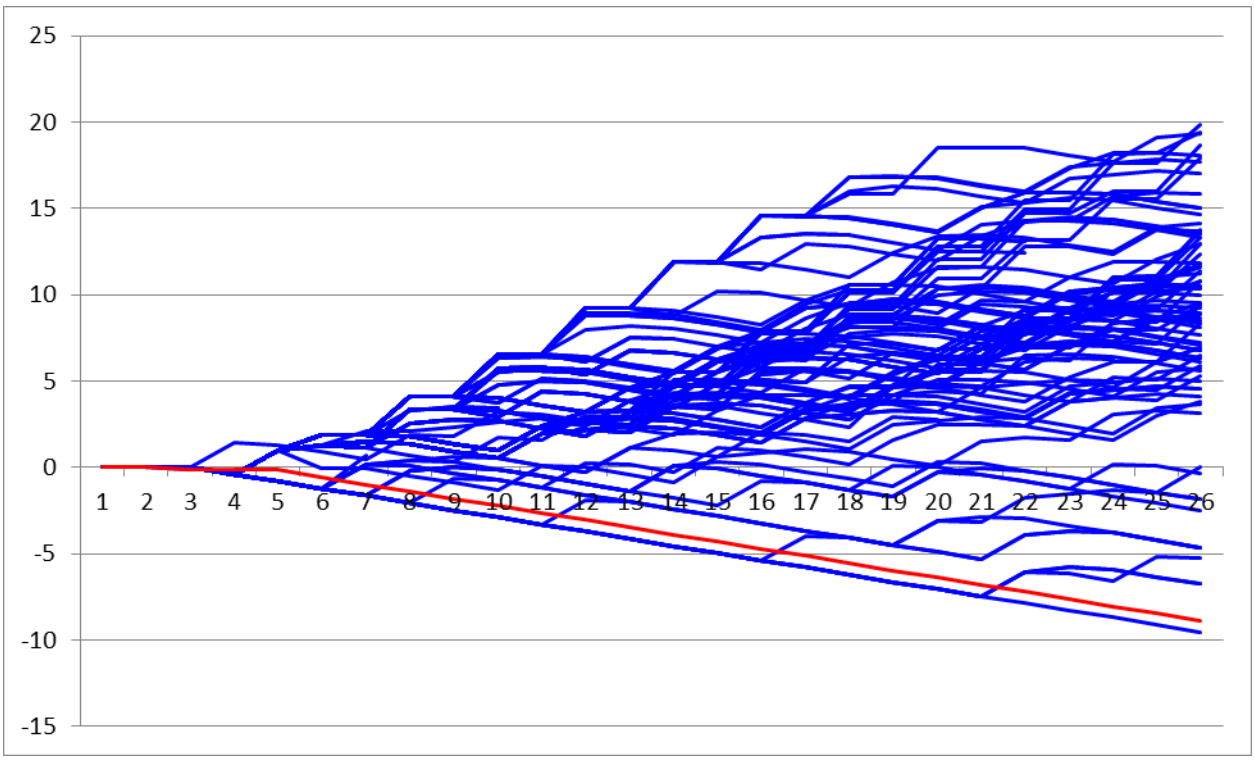

Figure C12. Order One, Multi class, Weighted

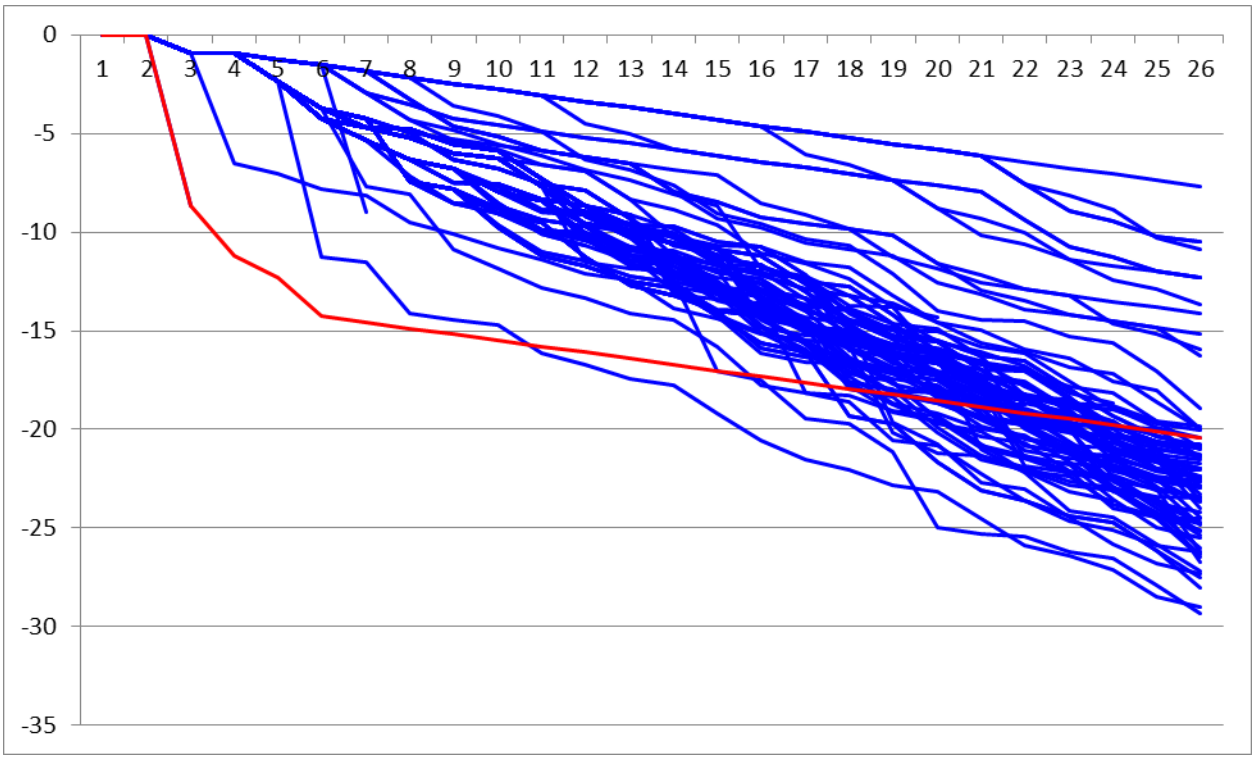

Figure C13. Order Two, One class, Unweighted

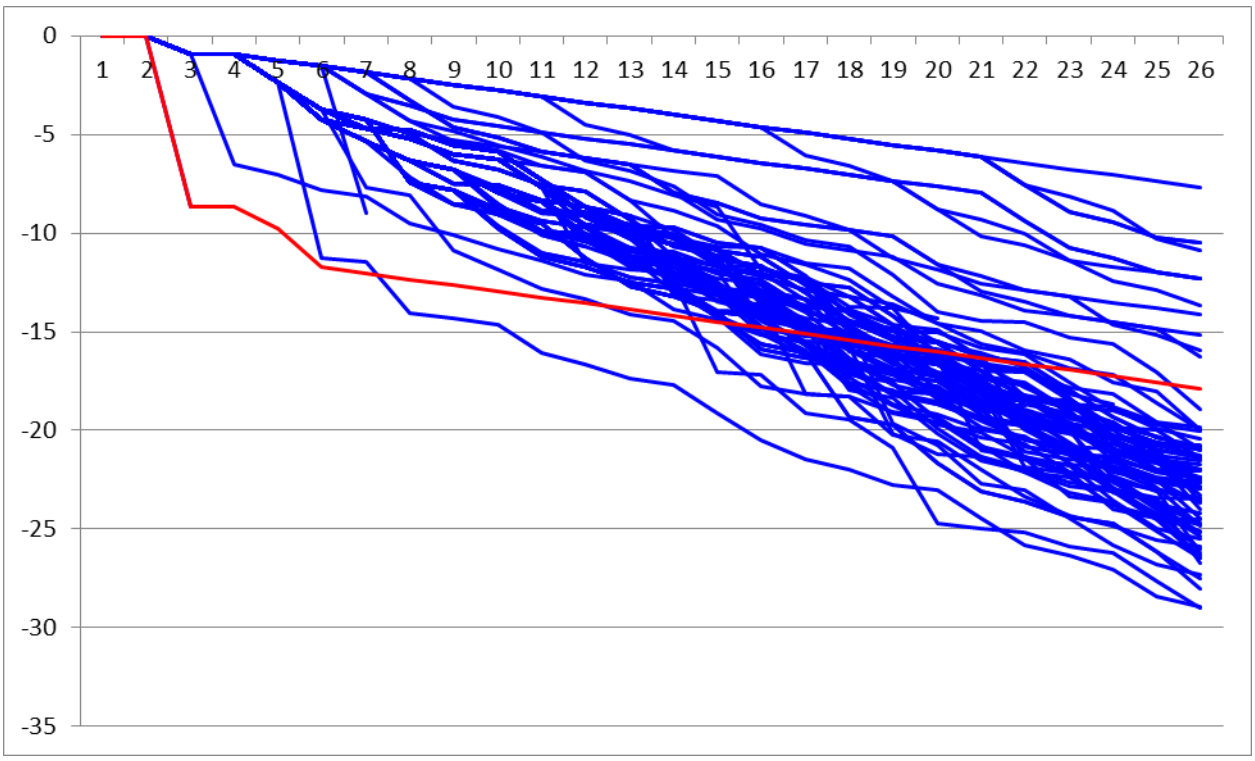

Figure C14. Order Two, One class, Weighted

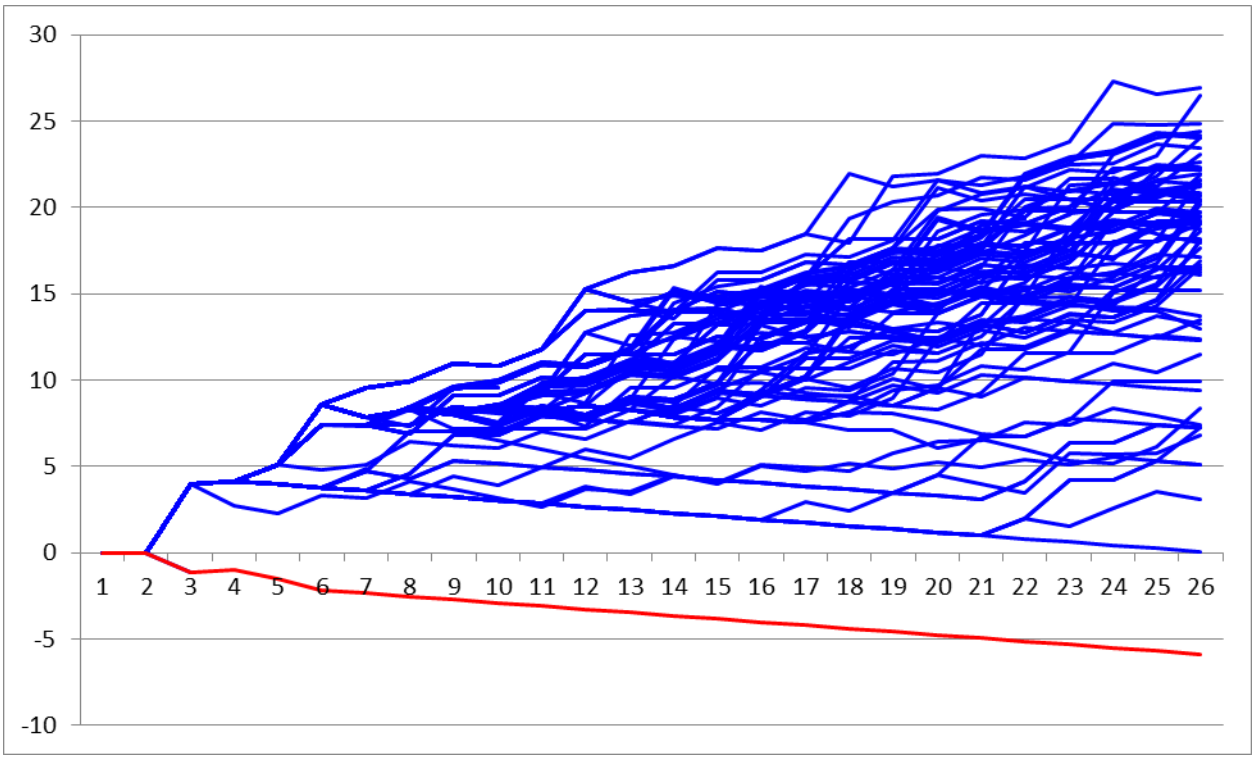

Figure C15. Order Two, Two class, Unweighted

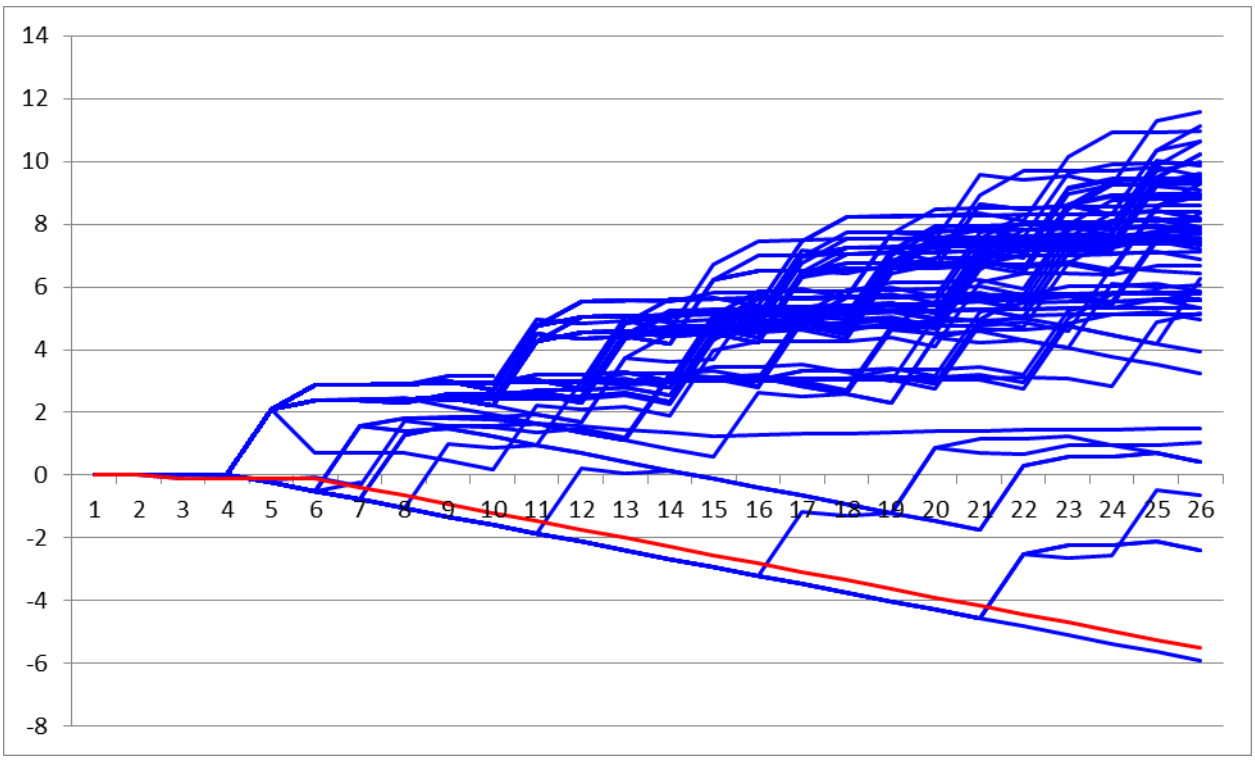

Figure C16. Order Two, Two class, Weighted

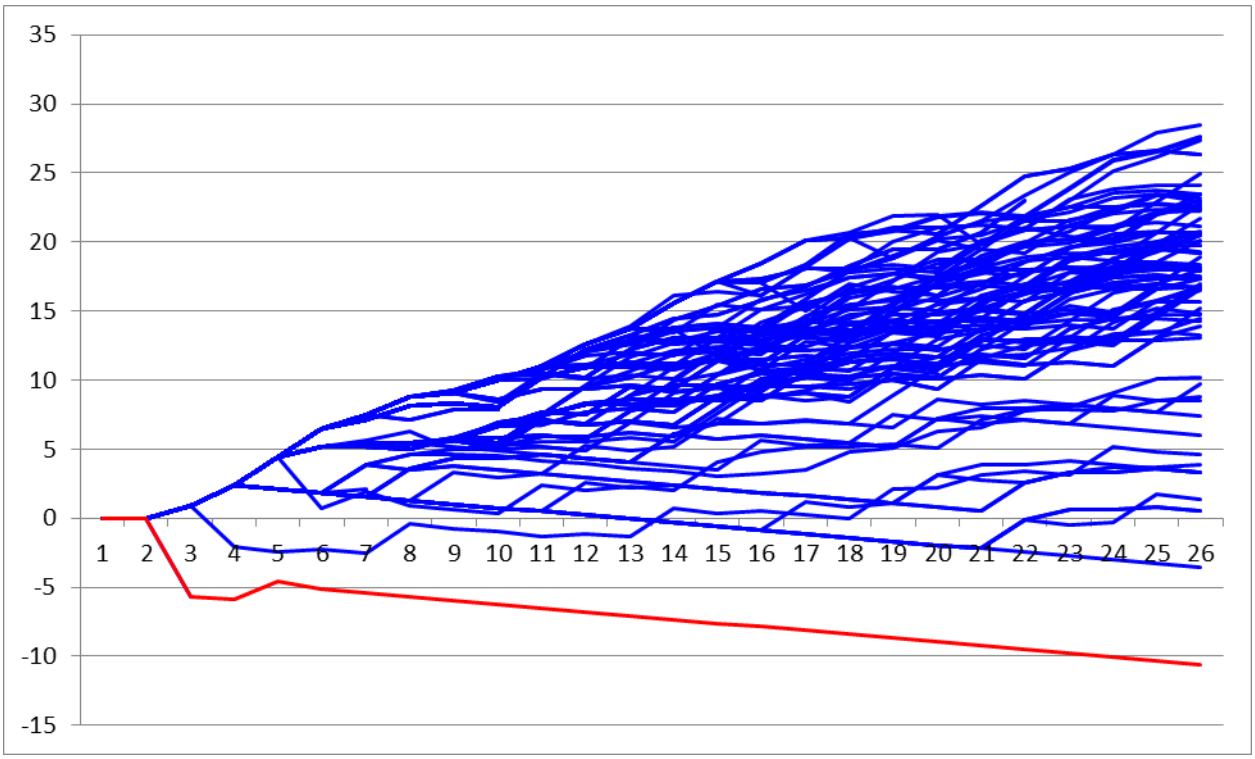

Figure C17. Order Two, Multi class, Unweighted

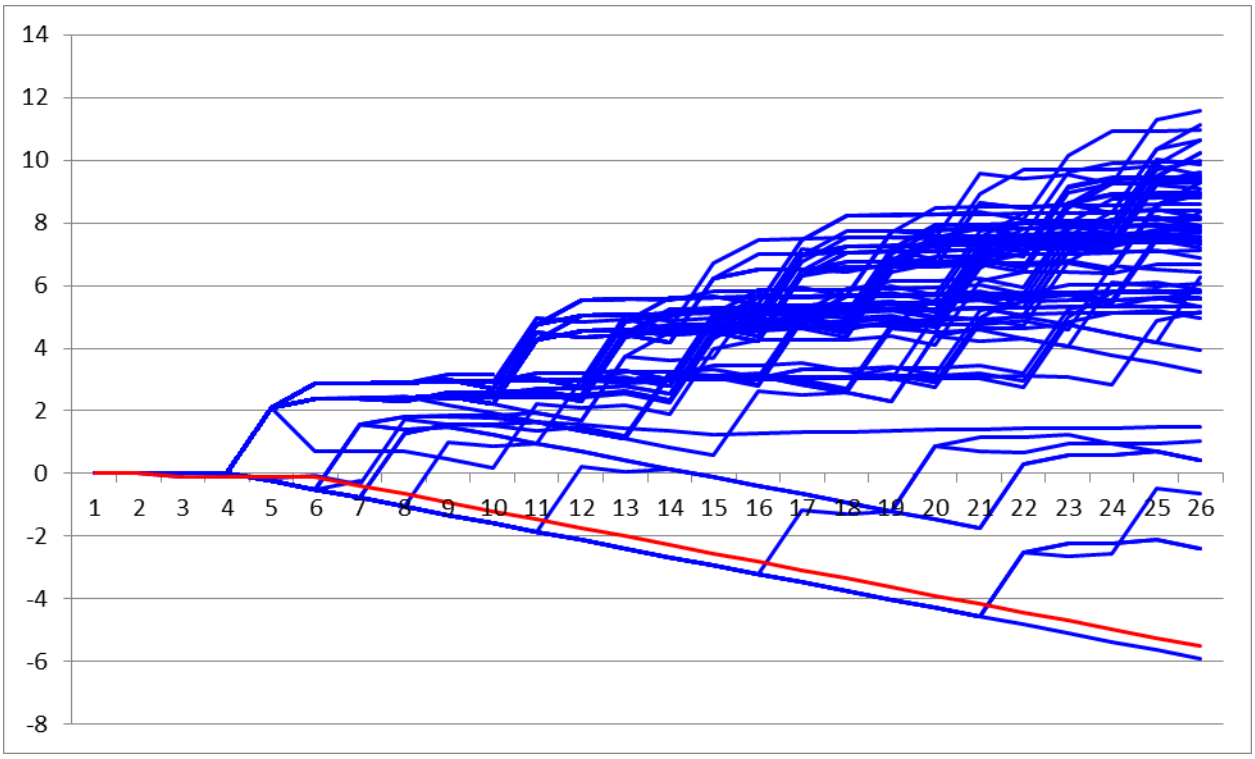

Figure C18. Order Two, Multi class, Weighted

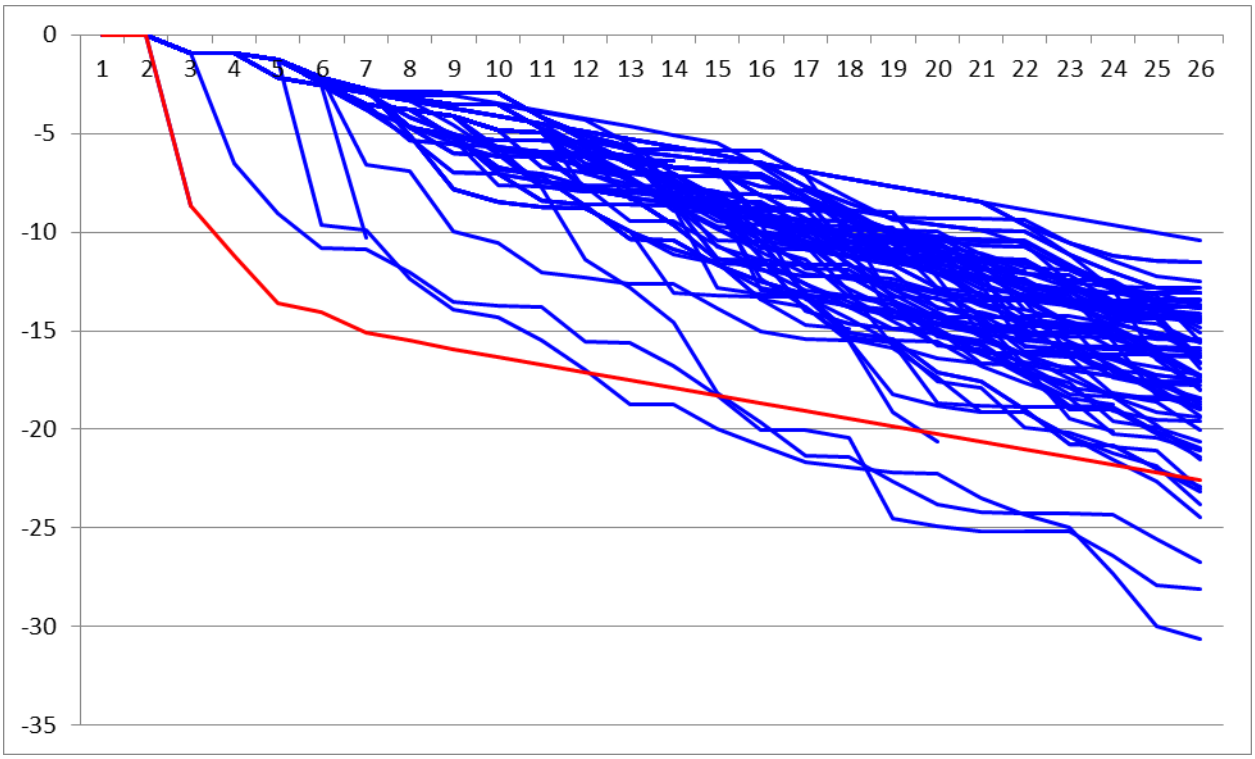

Figure C19. Order Three, One class, Unweighted

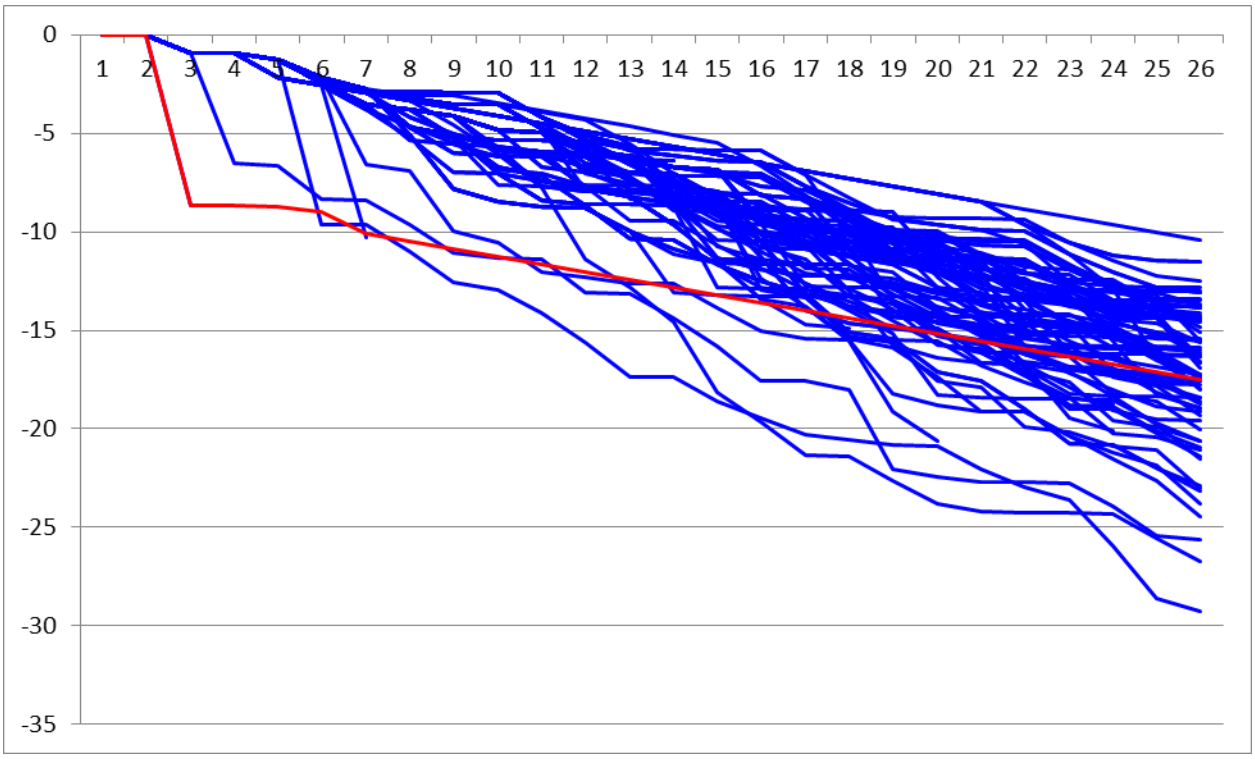

Figure C20. Order Three, One class, Weighted

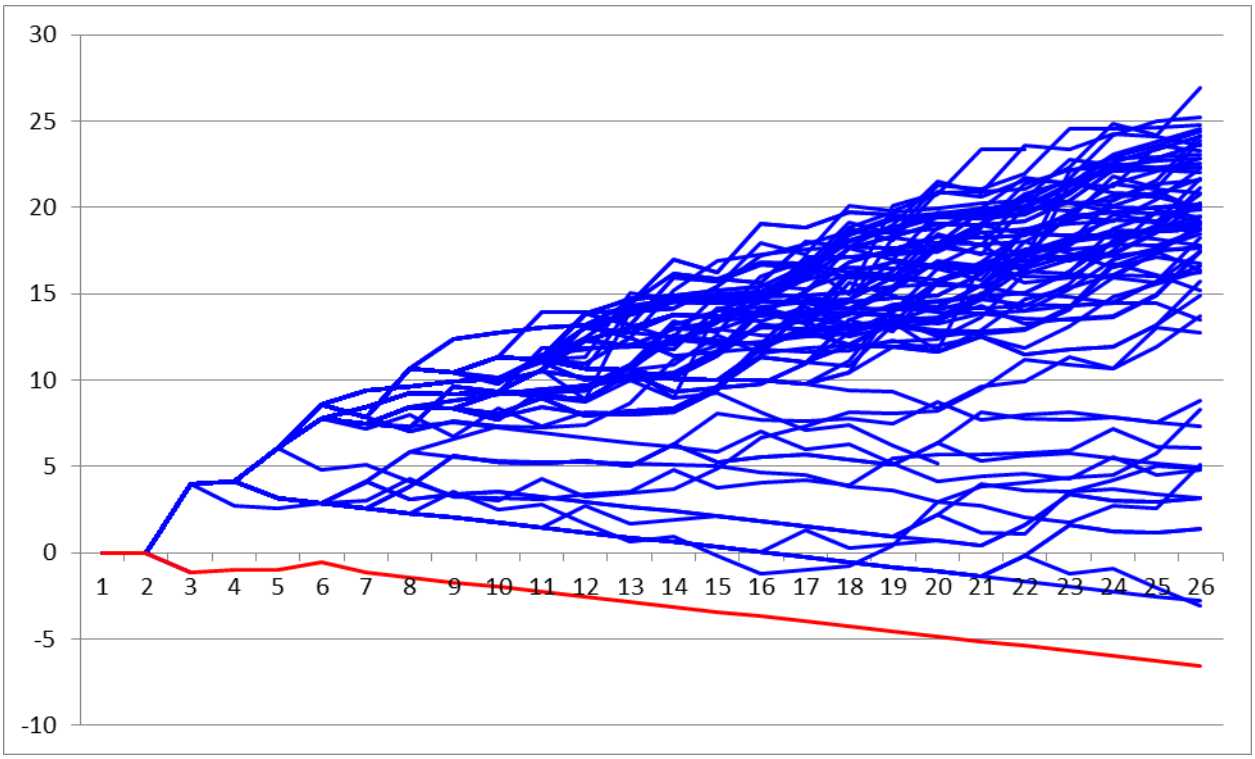

Figure C21. Order Three, Two class, Unweighted

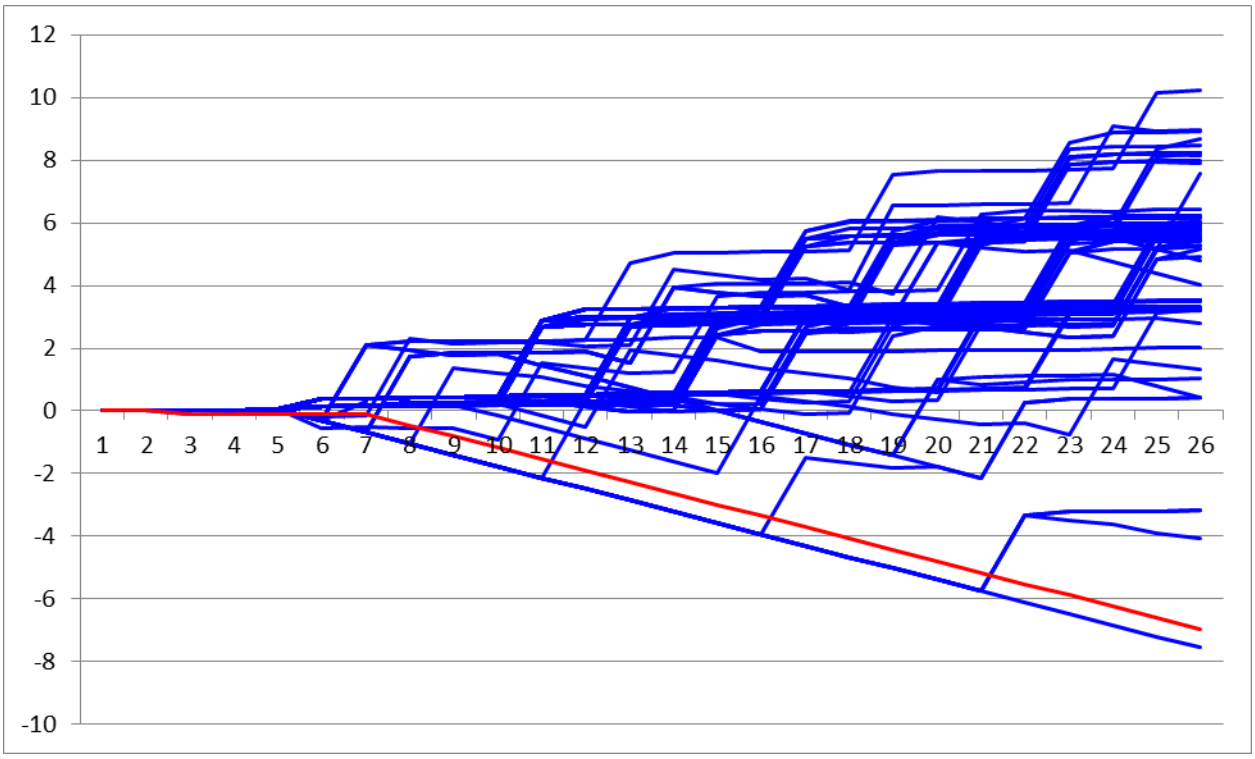

Figure C22. Order Three, Two class, Weighted

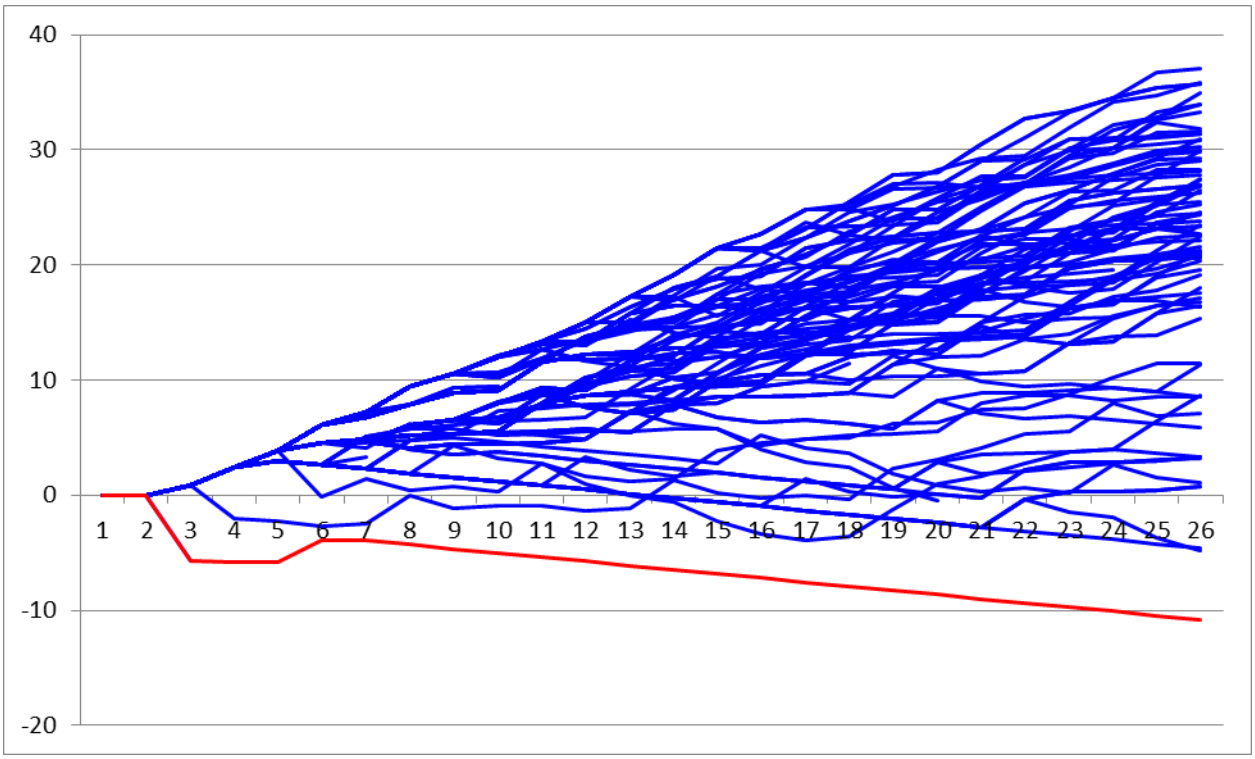

Figure C23. Order Three, Multi class, Unweighted

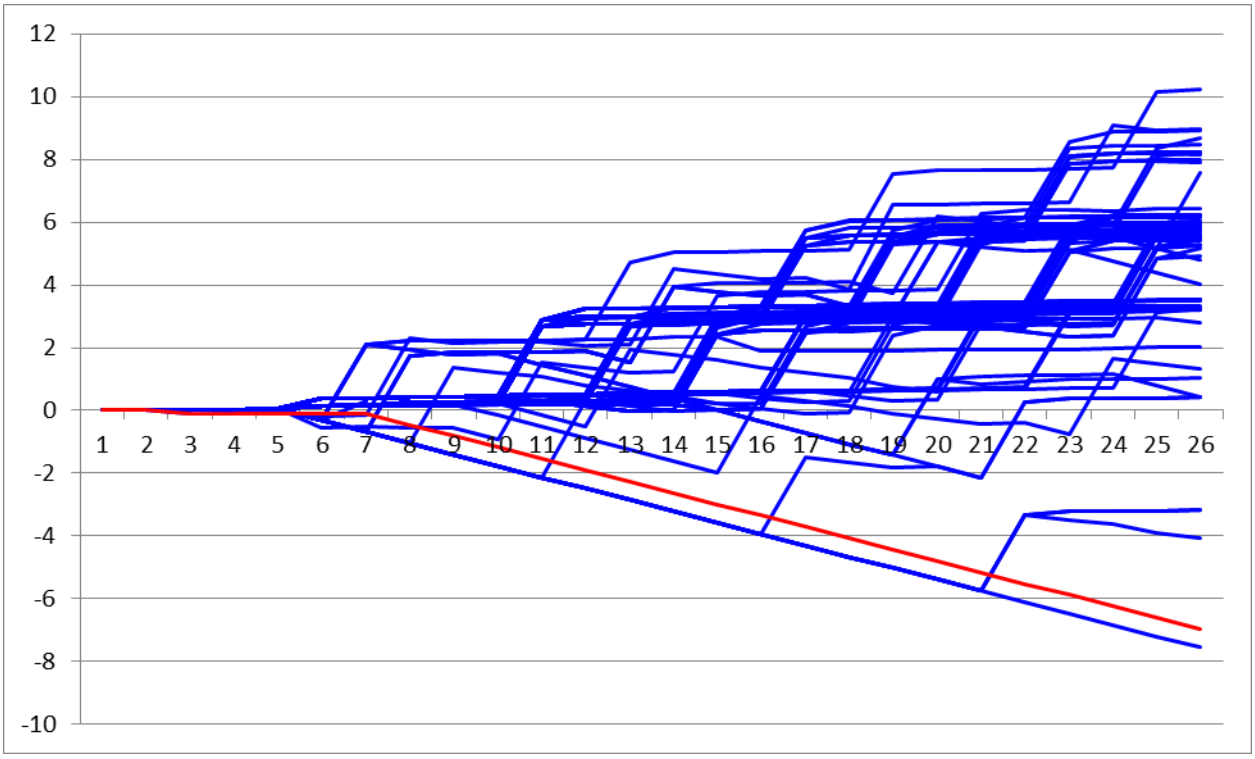

Figure C24. Order Three, Multi class, Weighted

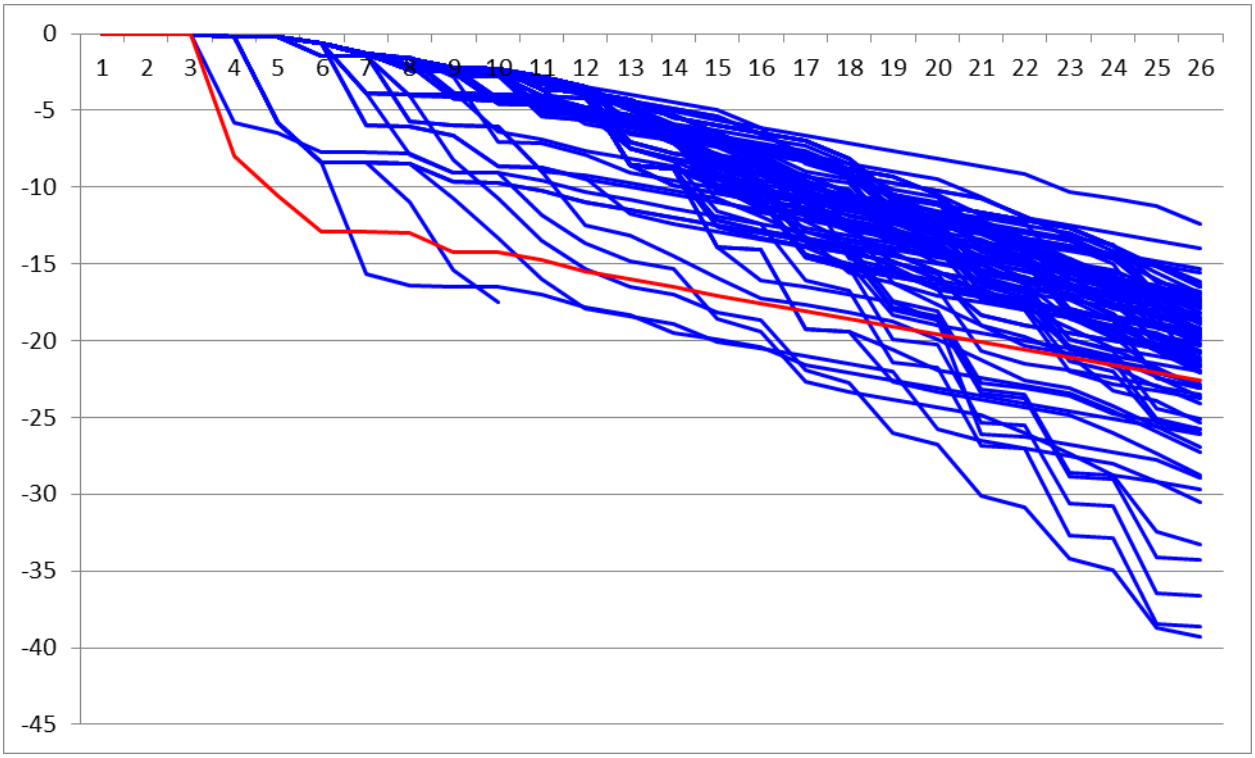

Figure C25. Order One, One class, Unweighted, Pseudo Timing

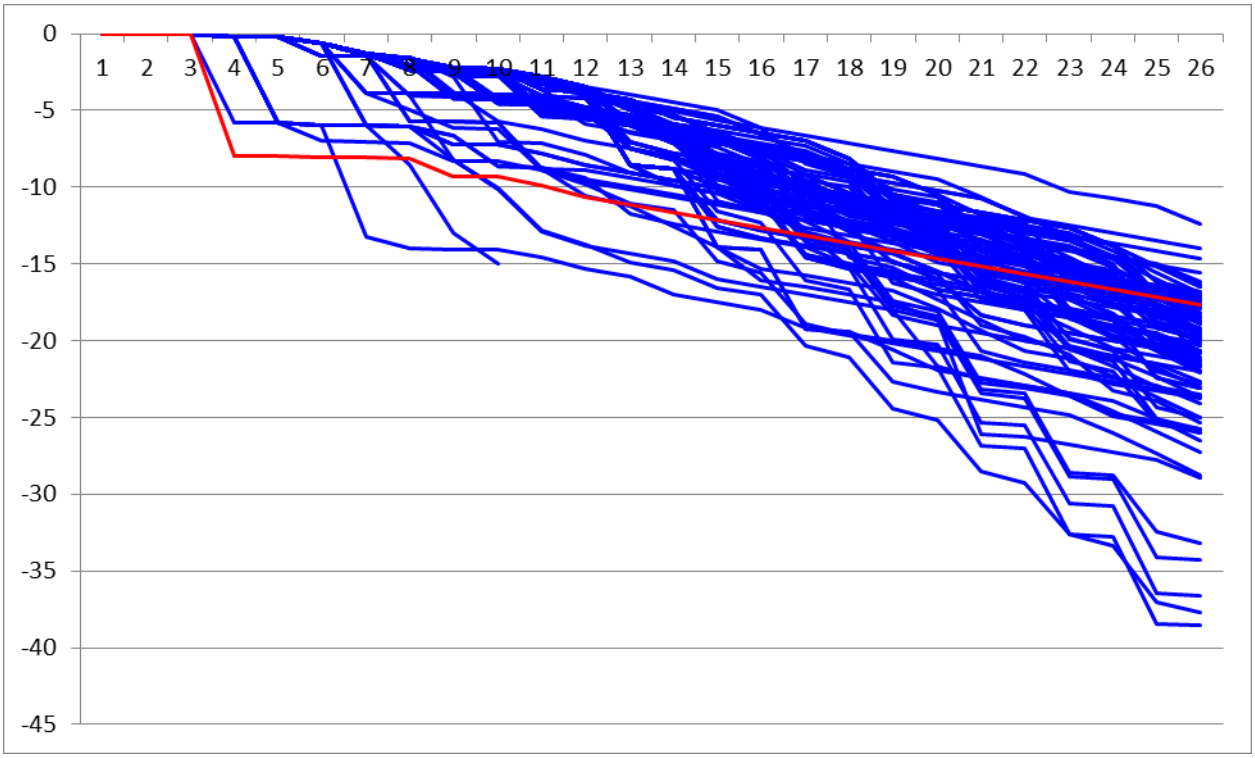

Figure C26. Order One, One class, Weighted, Pseudo Timing

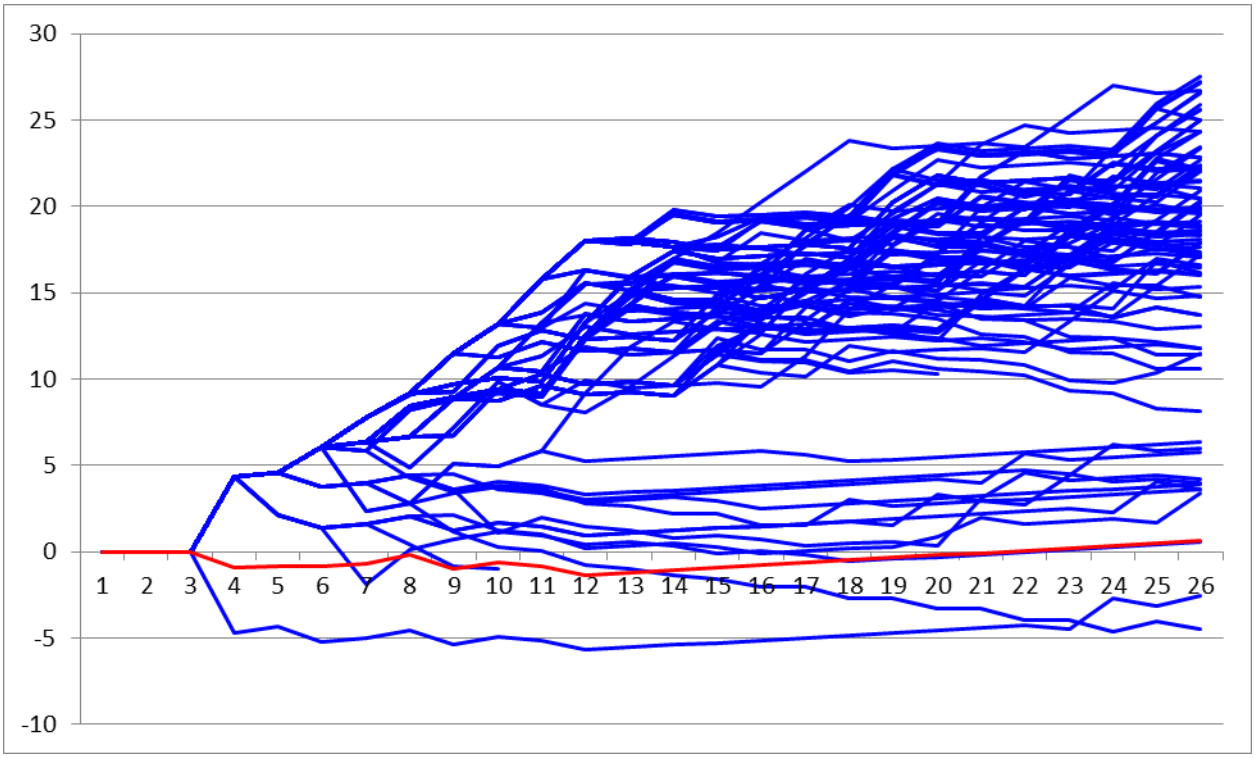

Figure C27. Order One, Two class, Unweighted, Pseudo Timing

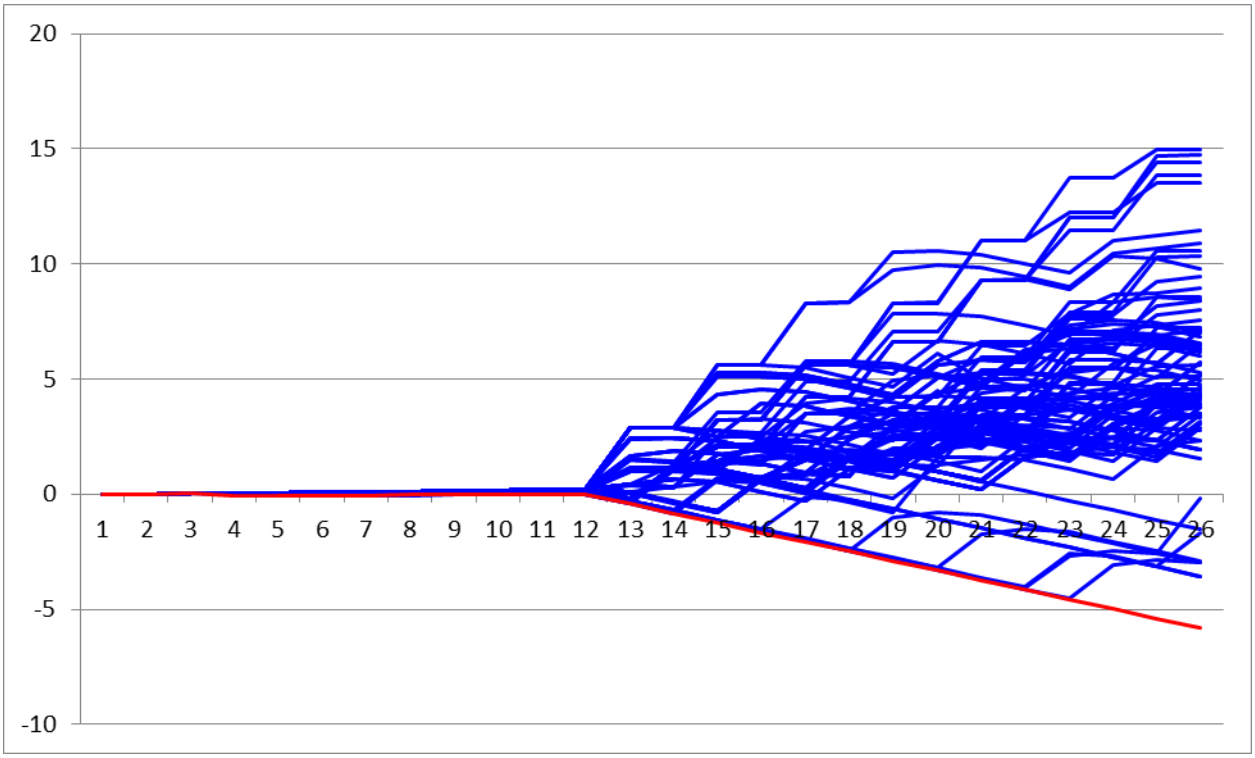

Figure C28. Order One, Two class, Weighted, Pseudo Timing

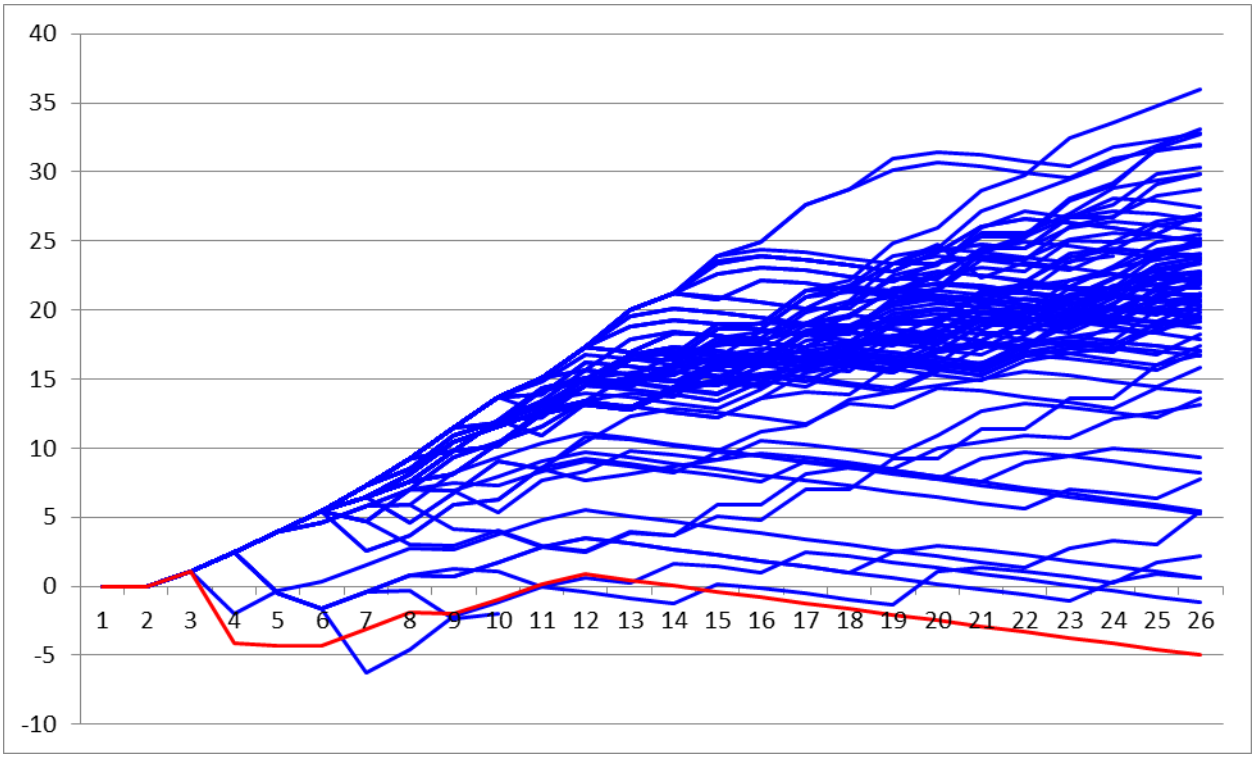

Figure C29. Order One, Multi class, Unweighted, Pseudo Timing

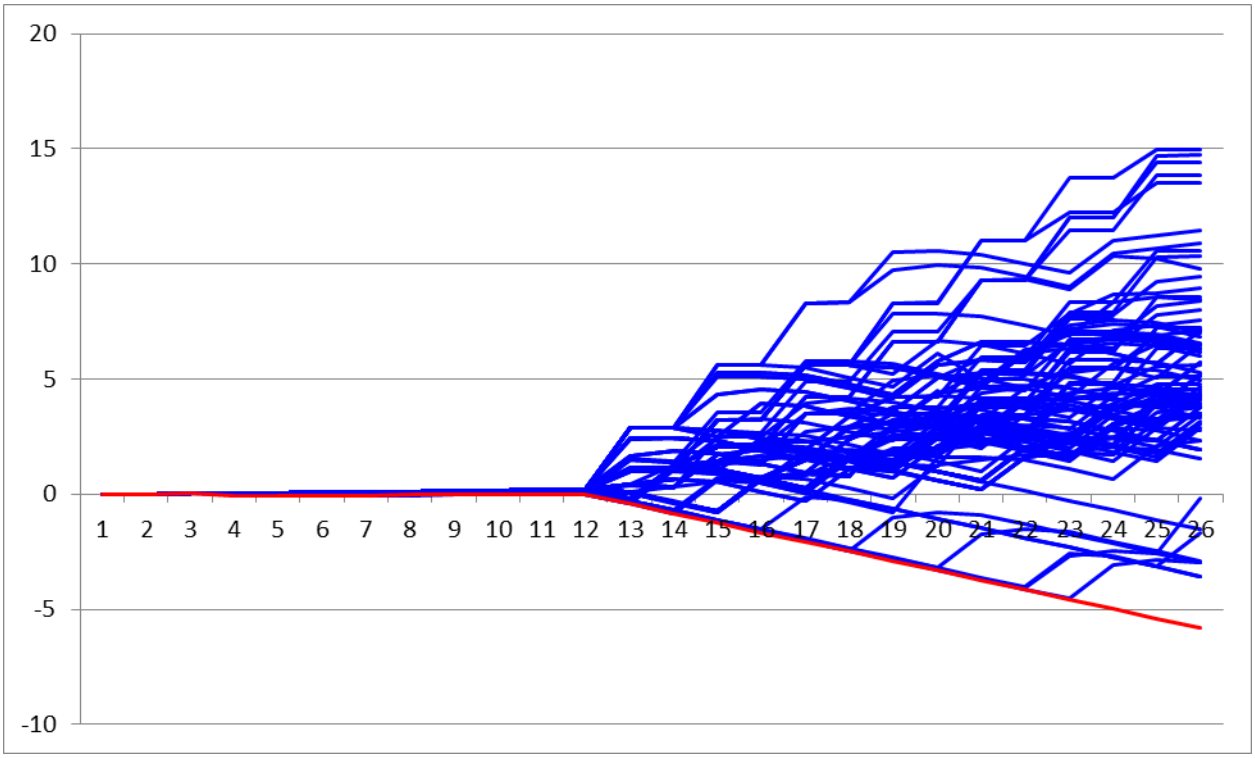

Figure C30. Order One, Multi class, Weighted, Pseudo Timing

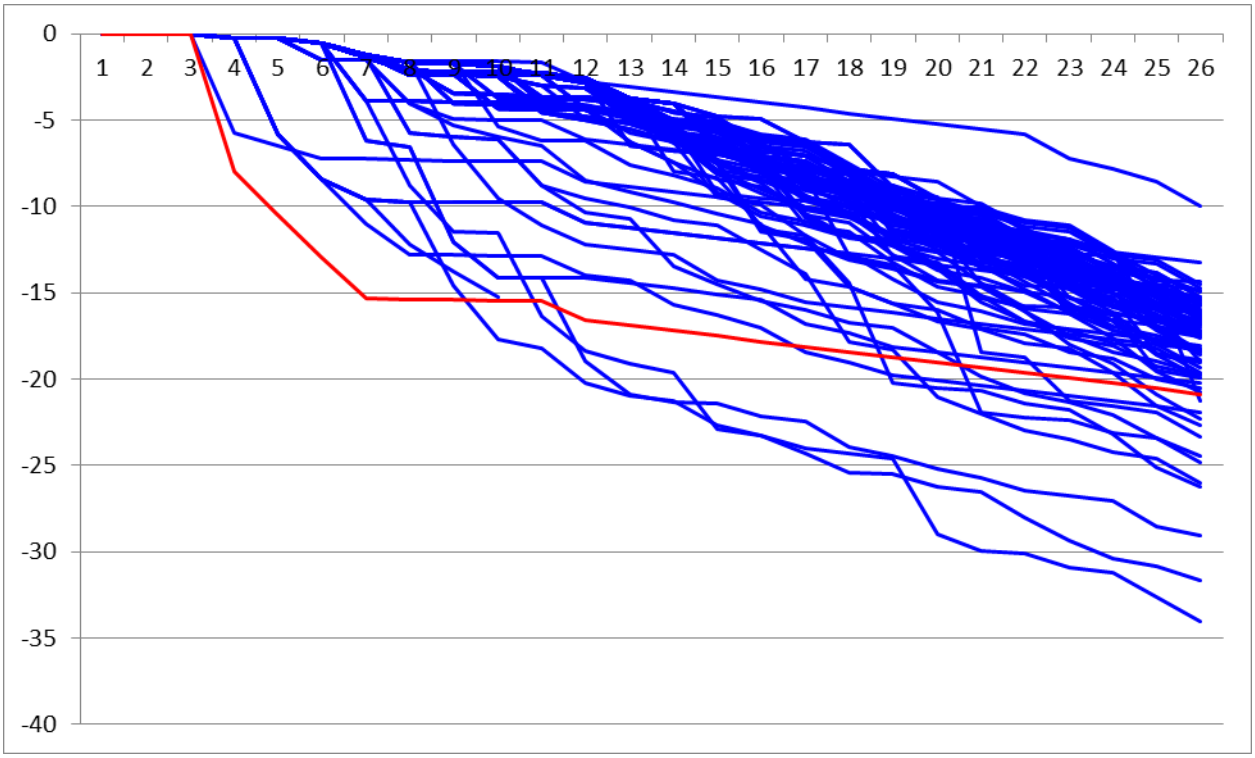

Figure C31. Order Two, One class, Unweighted, Pseudo Timing

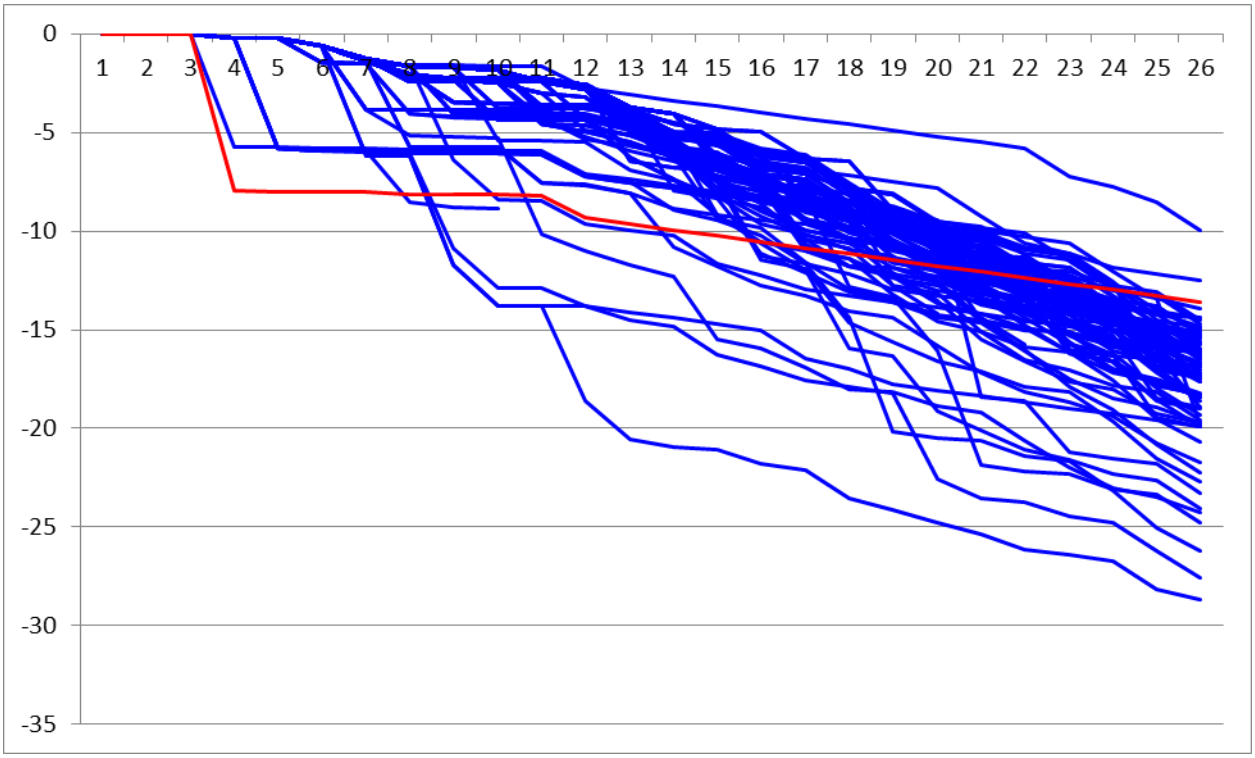

Figure C32. Order Two, One class, Weighted, Pseudo Timing

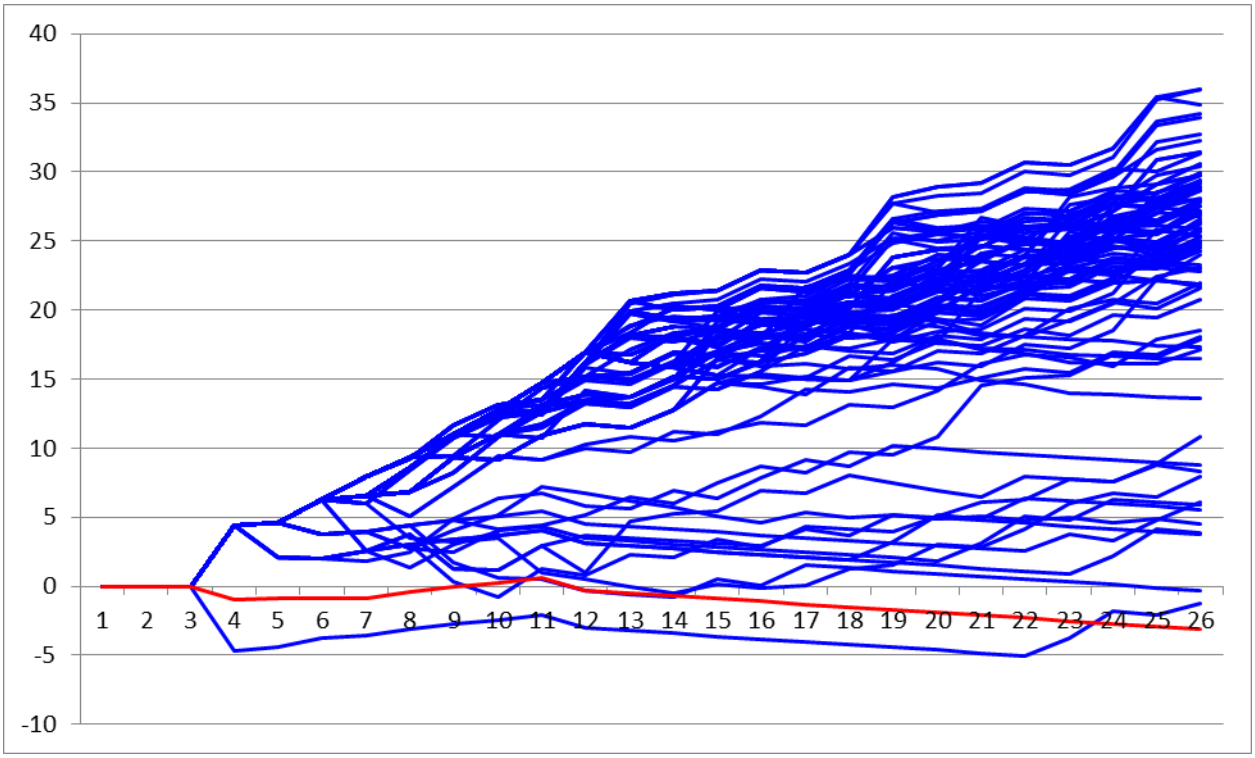

Figure C33. Order Two, Two class, Unweighted, Pseudo Timing

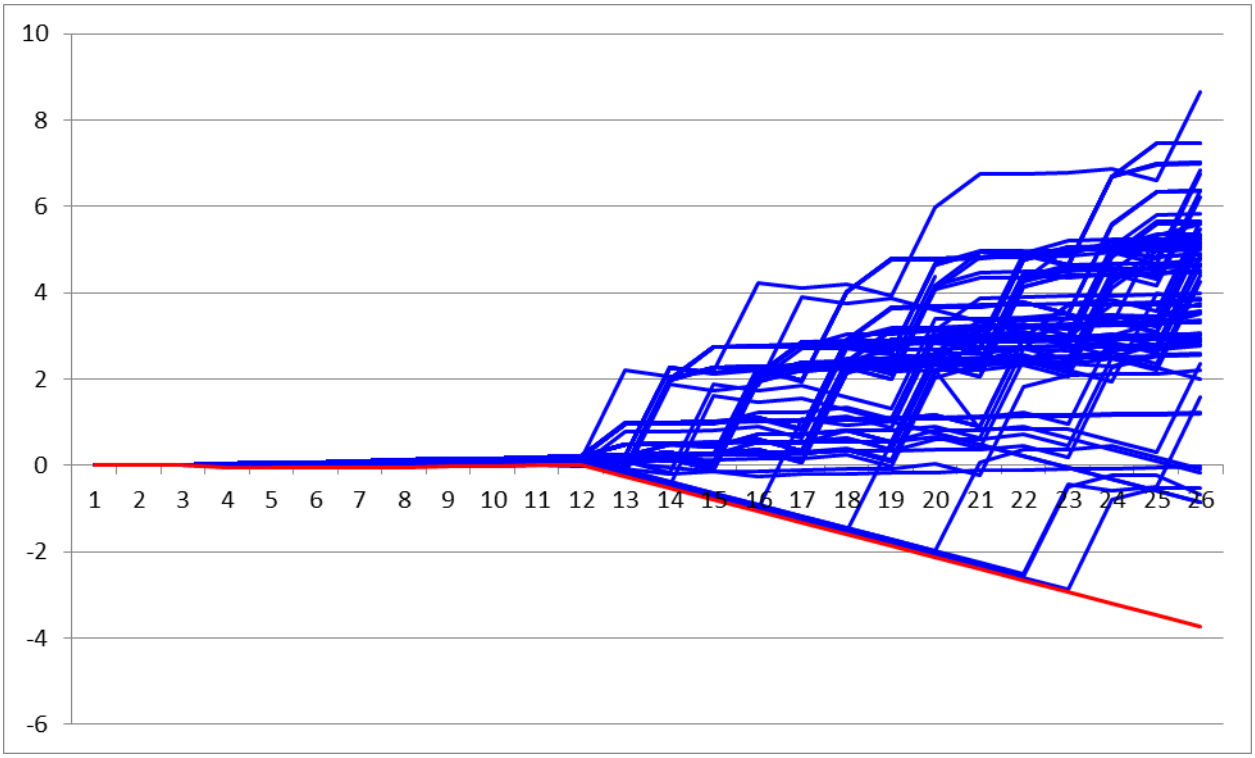

Figure C34. Order Two, Two class, Weighted, Pseudo Timing

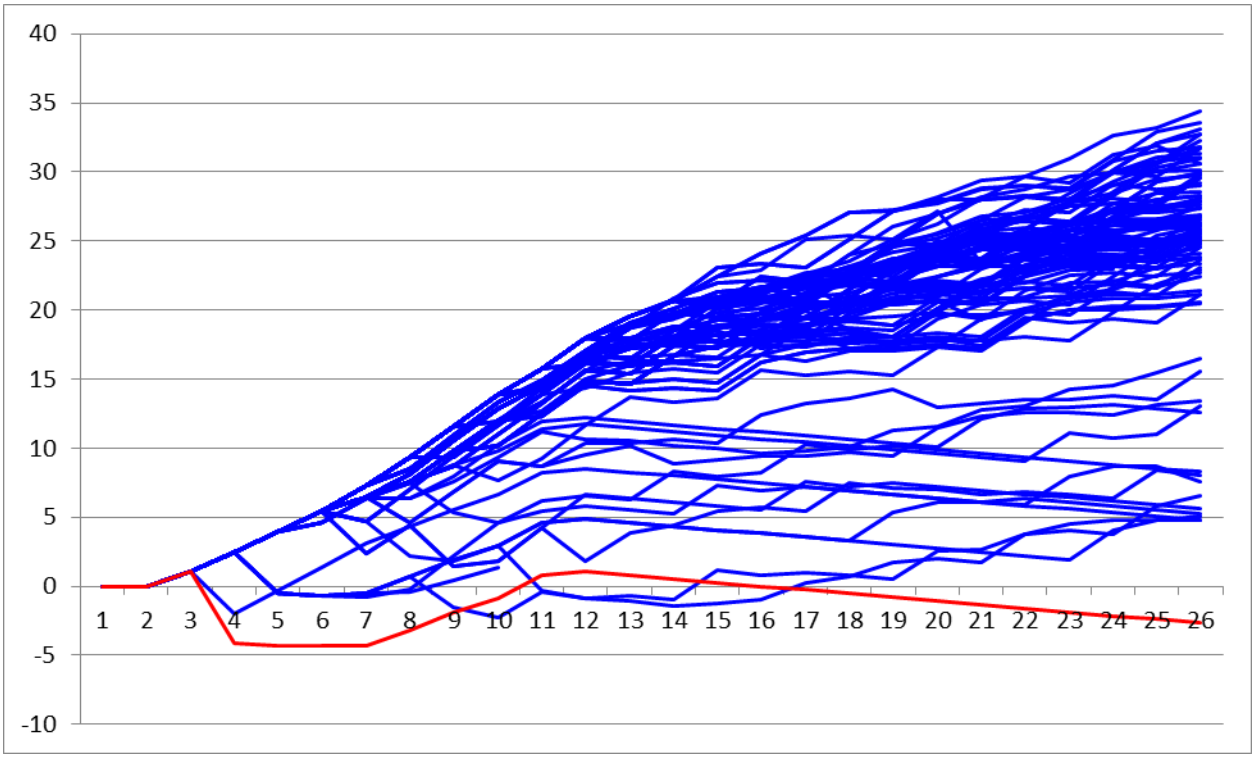

Figure C35. Order Two, Multi class, Unweighted, Pseudo Timing

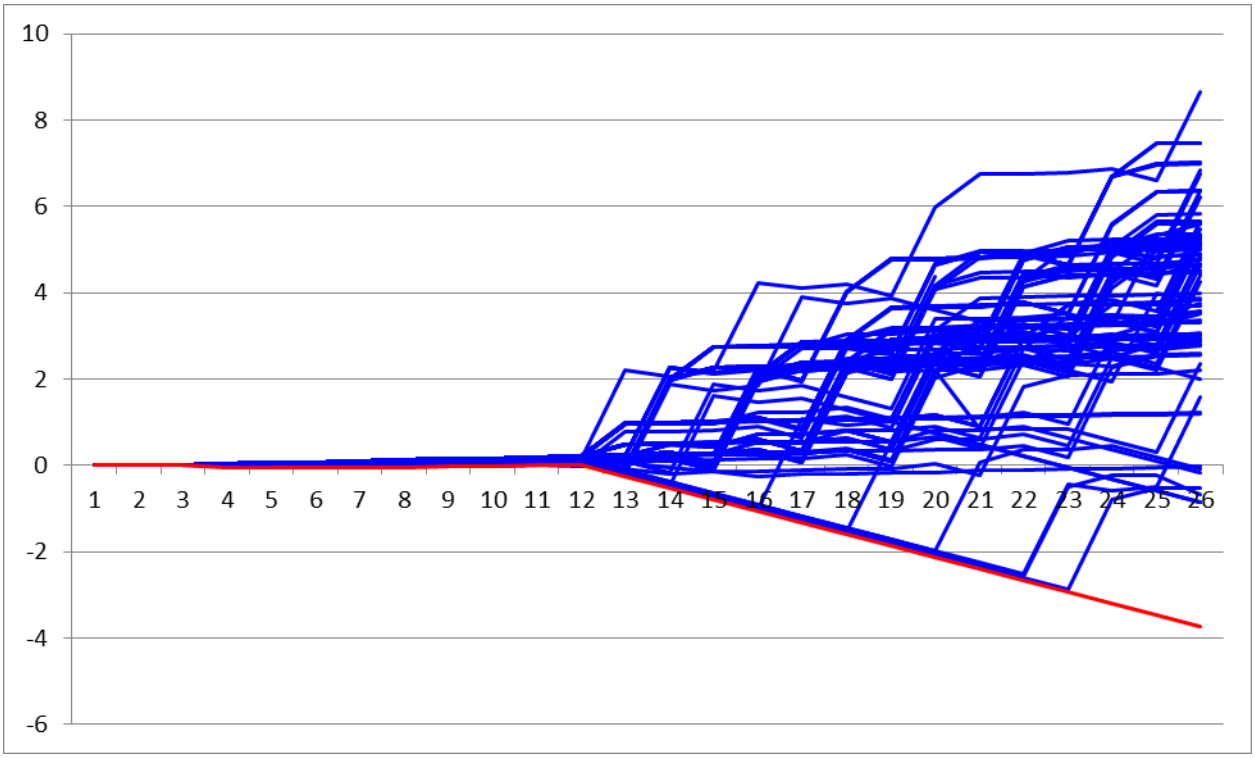

Figure C36. Order Two, Multi class, Weighted, Pseudo Timing

APPENDIX D. SATAN ATTACK DATASET SPRT GRAPHS

Satan is a set of attacks from the 1998 DARPA intrusion detection evaluation dataset. Included for reference is an organized listing of SPRT graphs calculated under explained Conditions.

DARPA Description: Network probing tool which looks for well-known weaknesses.

Data Counts

Table D1. Data Counts

	Training	Testing
Flows	2	2
Transitions	7189	247
Packets	8957	320

Training Dataset

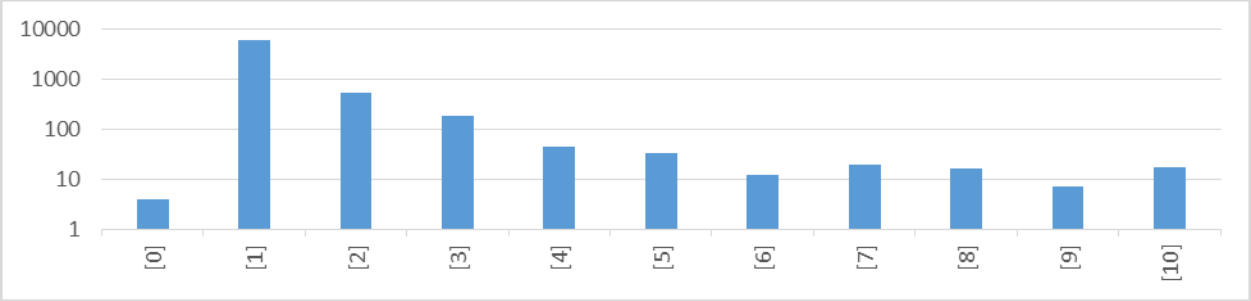

Figure D1. Satan Order 1 Training Dataset Histogram

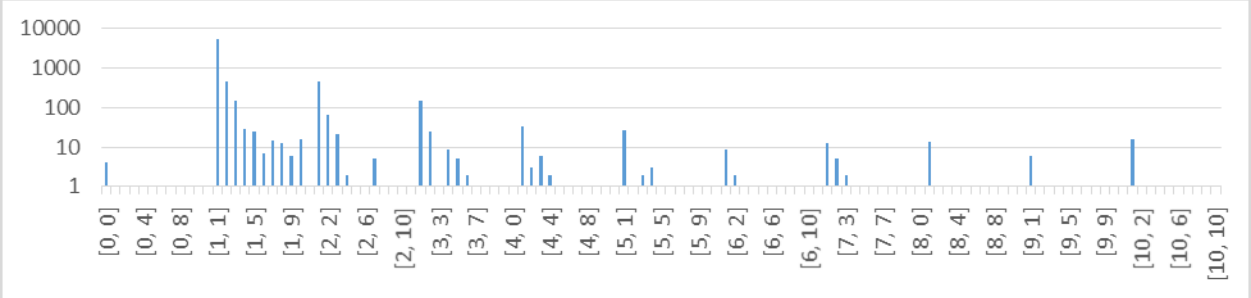

Figure D2. Satan Order 3 Training Dataset Histogram

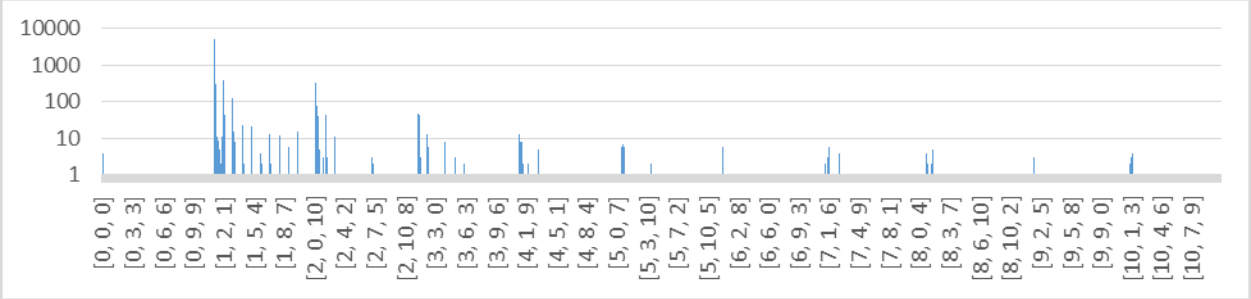

Figure D3. Satan Order 3 Training Dataset Histogram

Testing Dataset Histograms

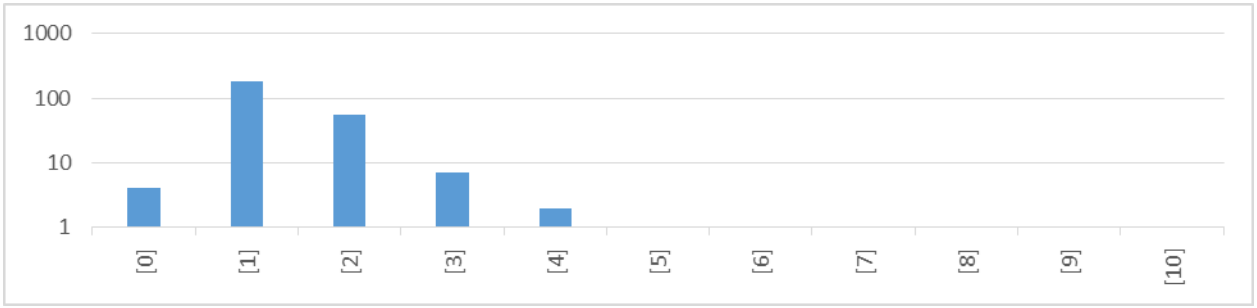

Figure D4. Satan Order 1 Testing Dataset Histogram

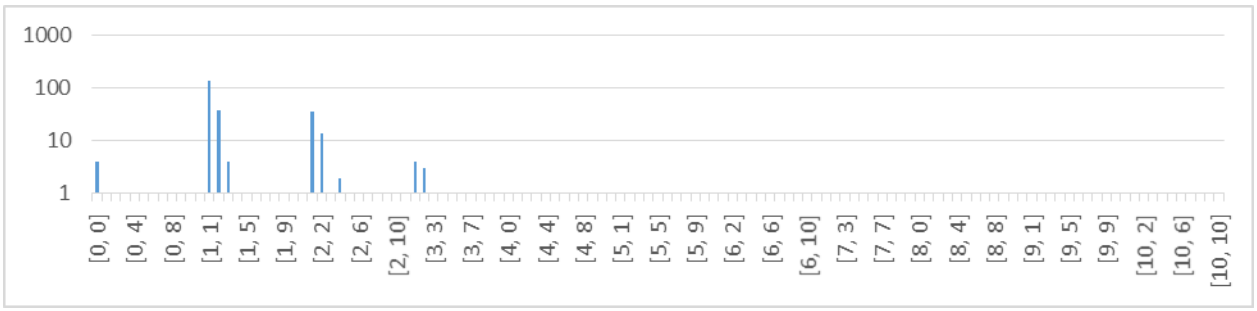

Figure D5. Satan Order 2 Testing Dataset Histogram

Figure D6. Satan Order 3 Testing Dataset Histogram

SPRT Graphs

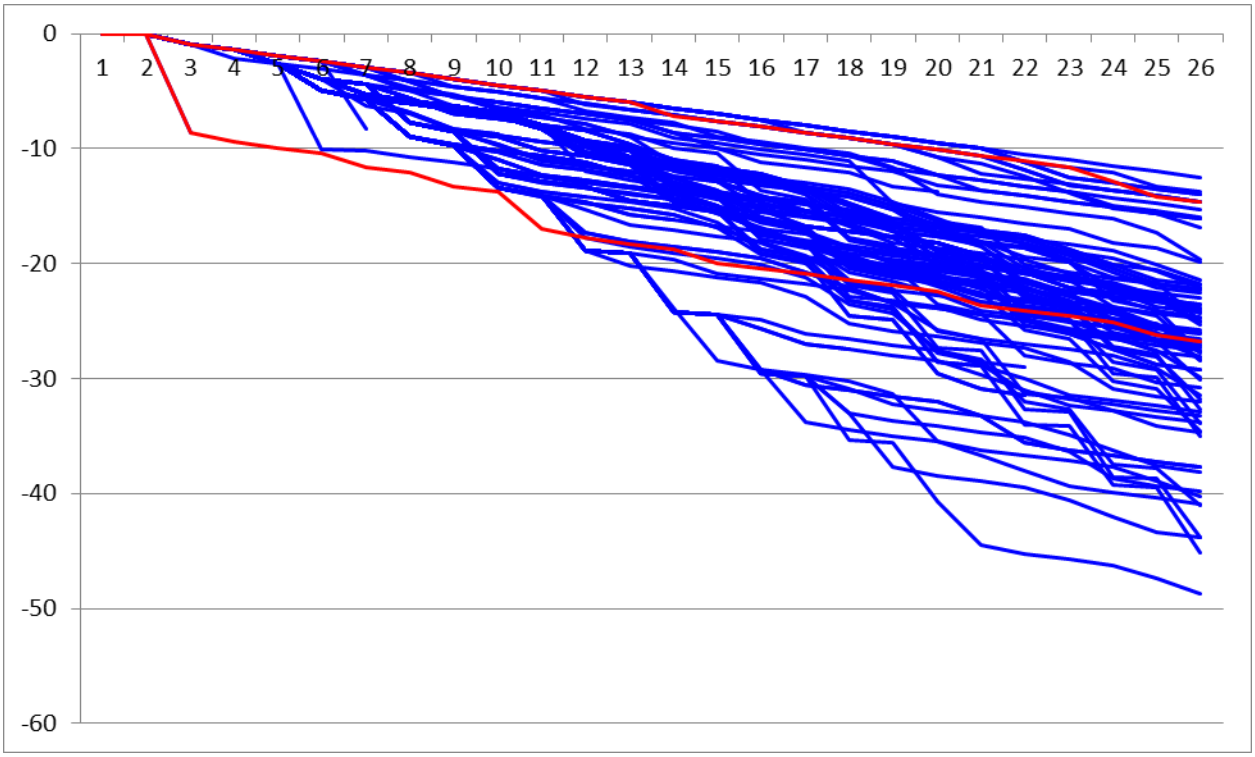

Figure D7. Order One, One class, Unweighted

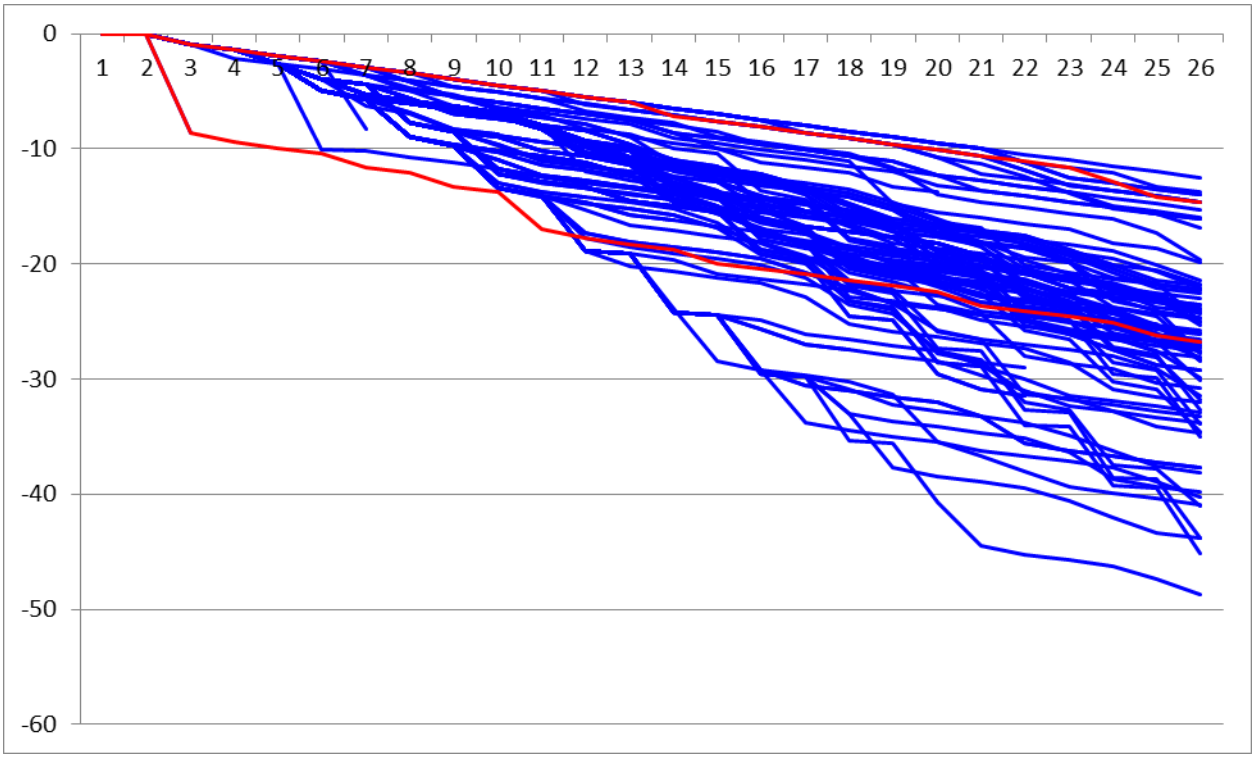

Figure D8.Order One, One class, Weighted

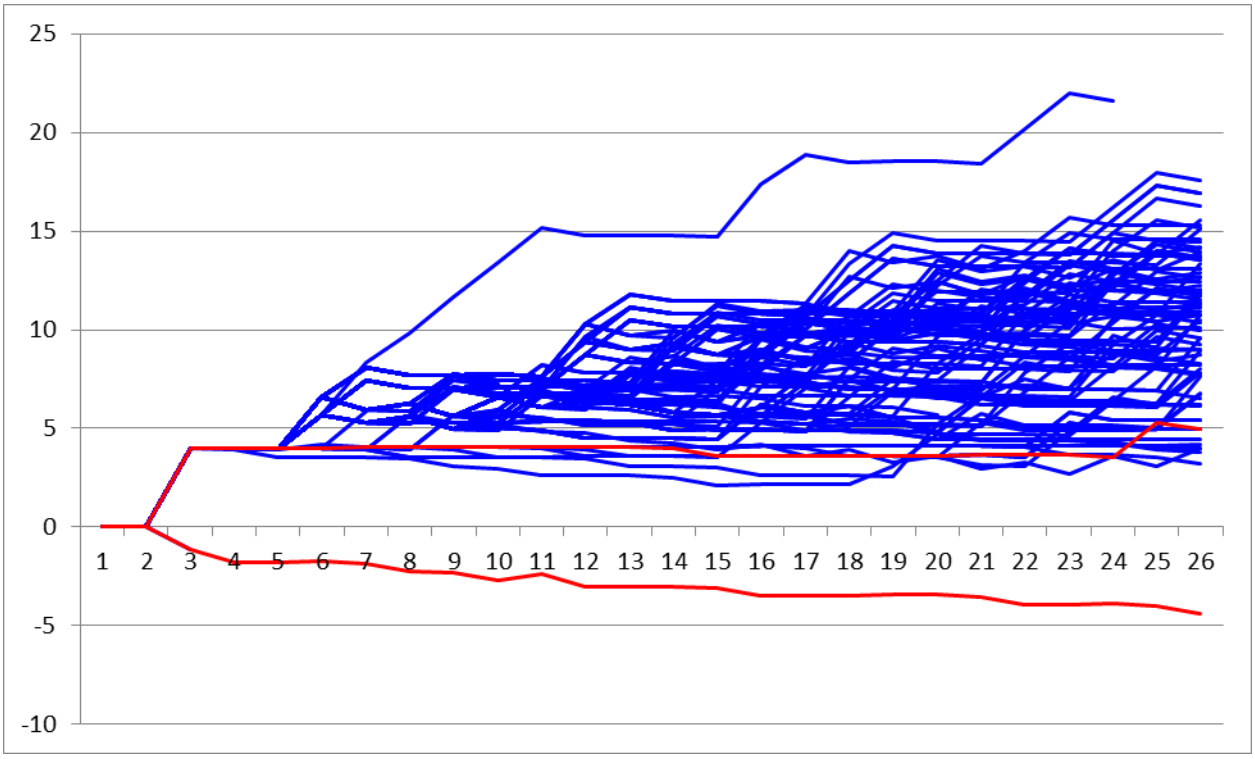

Figure D9. Order One, Two class, Unweighted

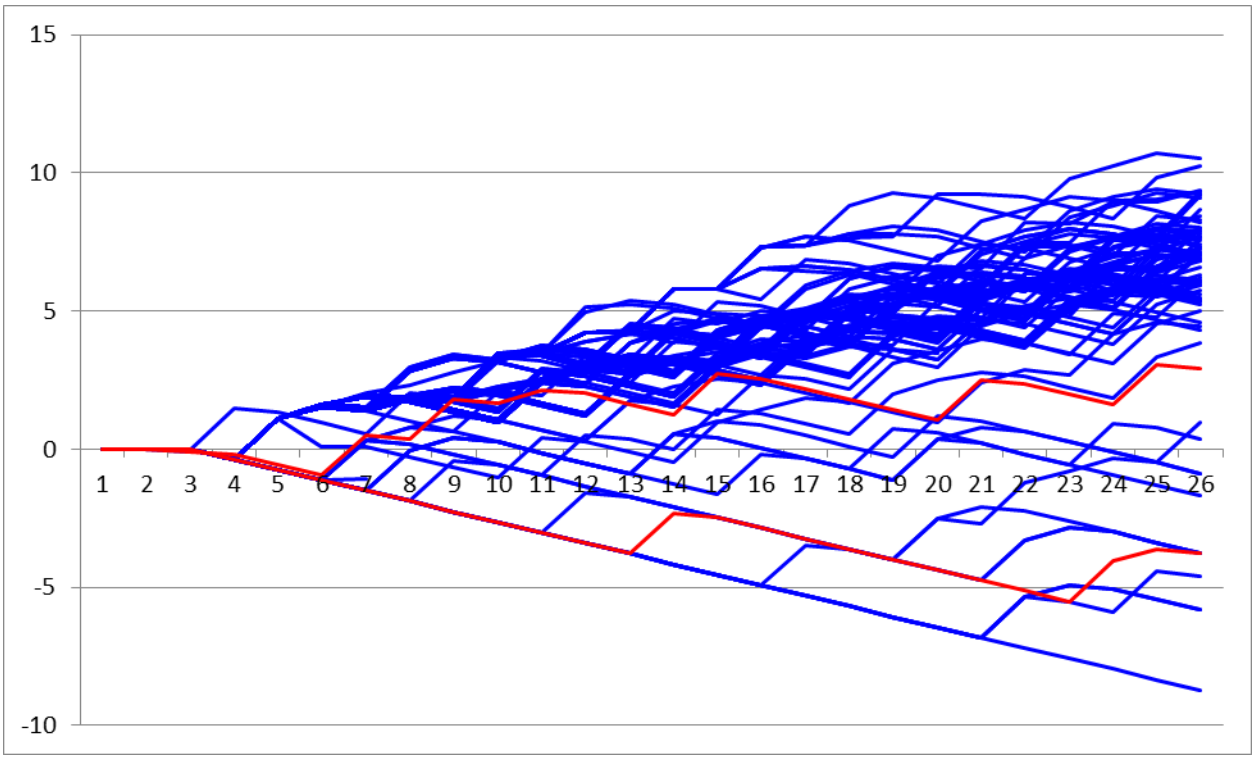

Figure D10. Order One, Two class, Weighted

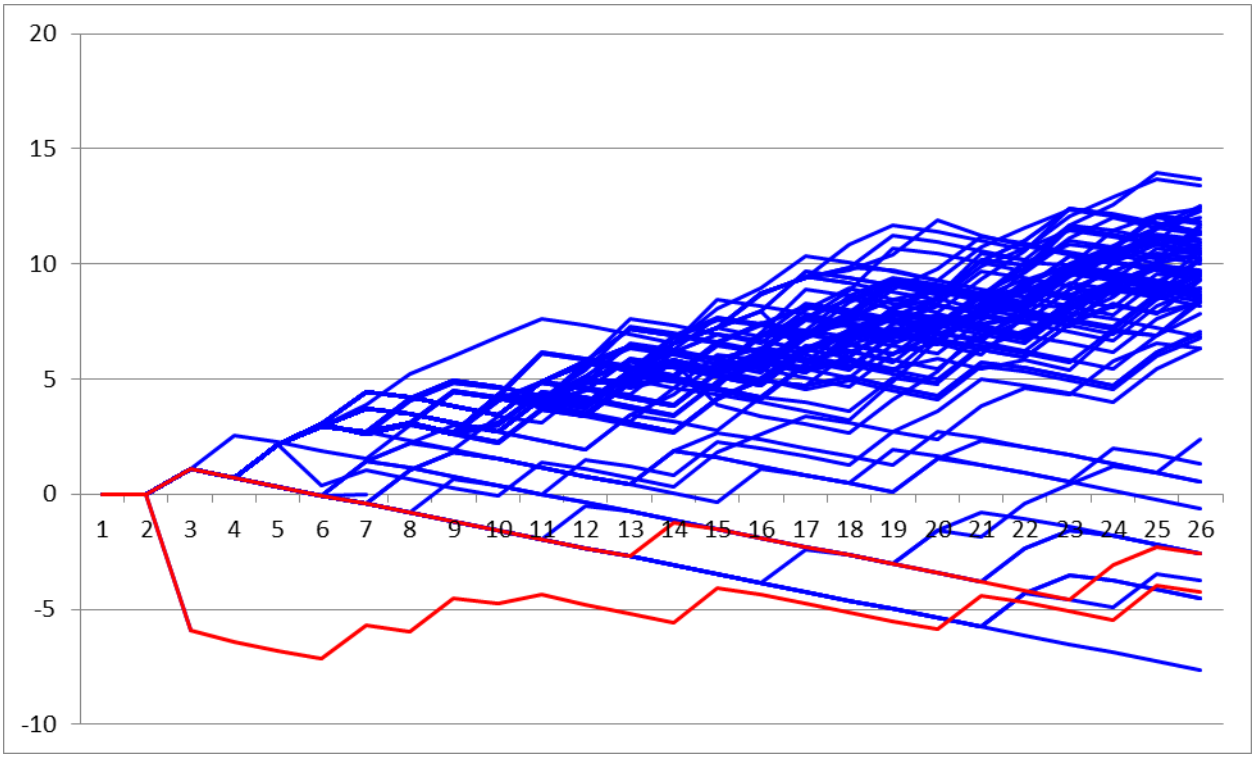

Figure D11. Order One, Multi class, Unweighted

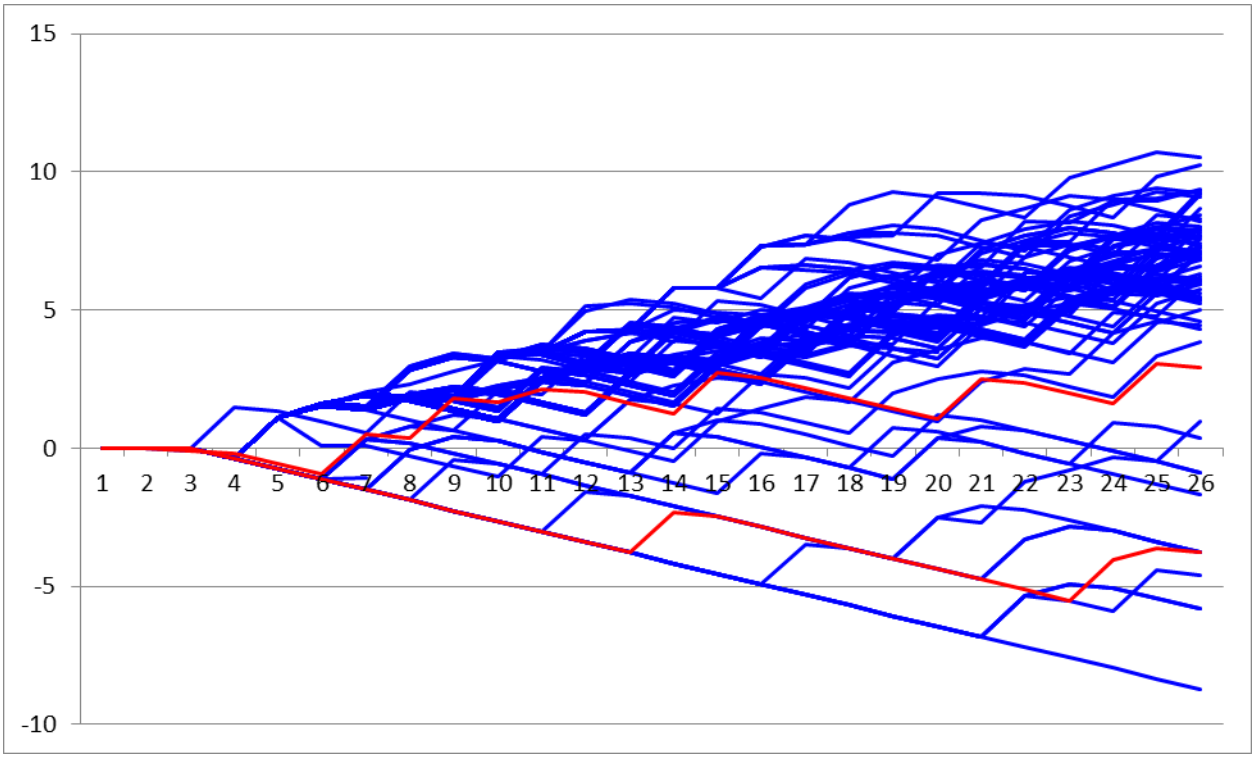

Figure D12. Order One, Multi class, Weighted

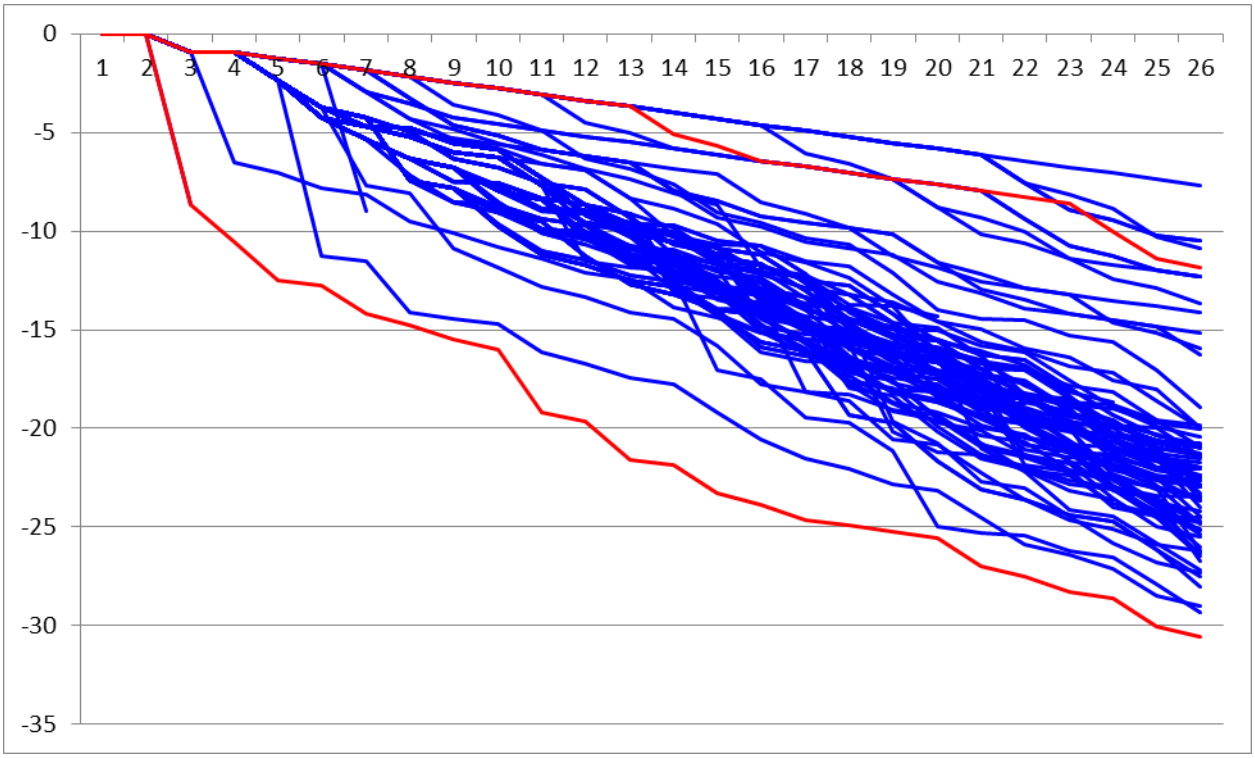

Figure D13. Order Two, One class, Unweighted

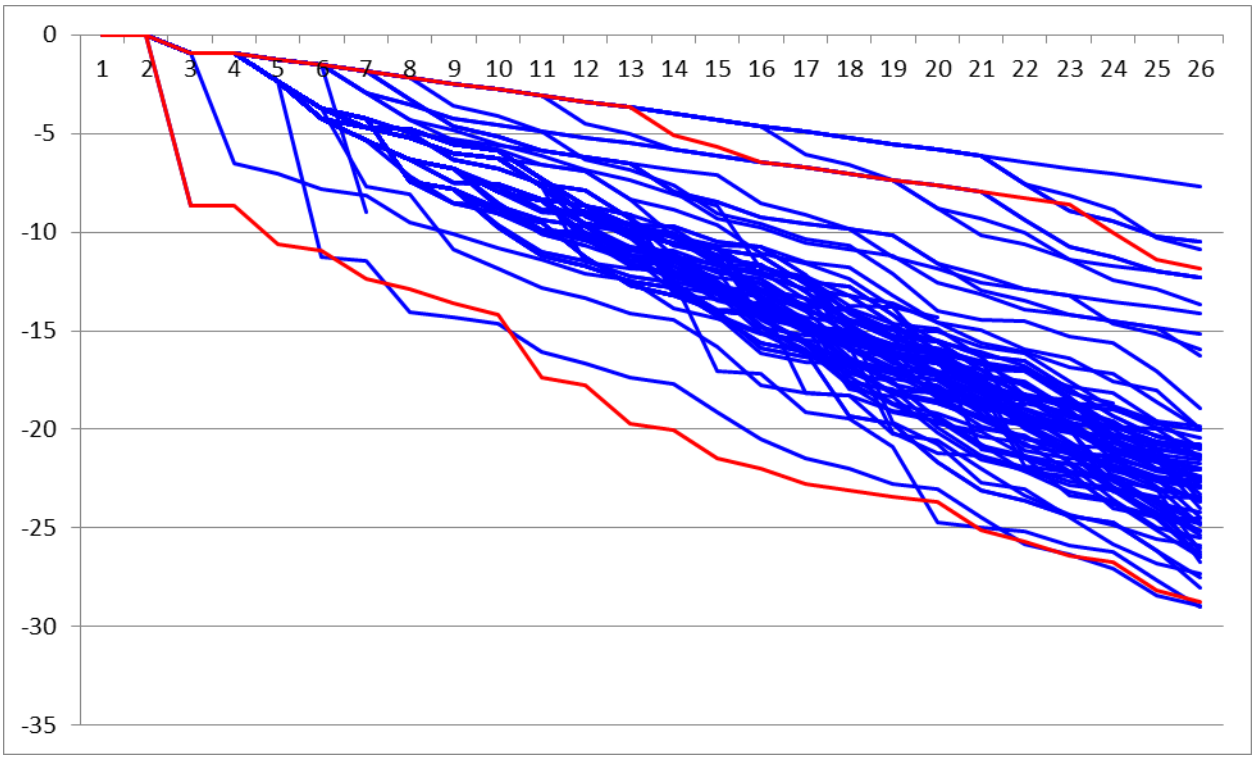

Figure D14. Order Two, One class, Weighted

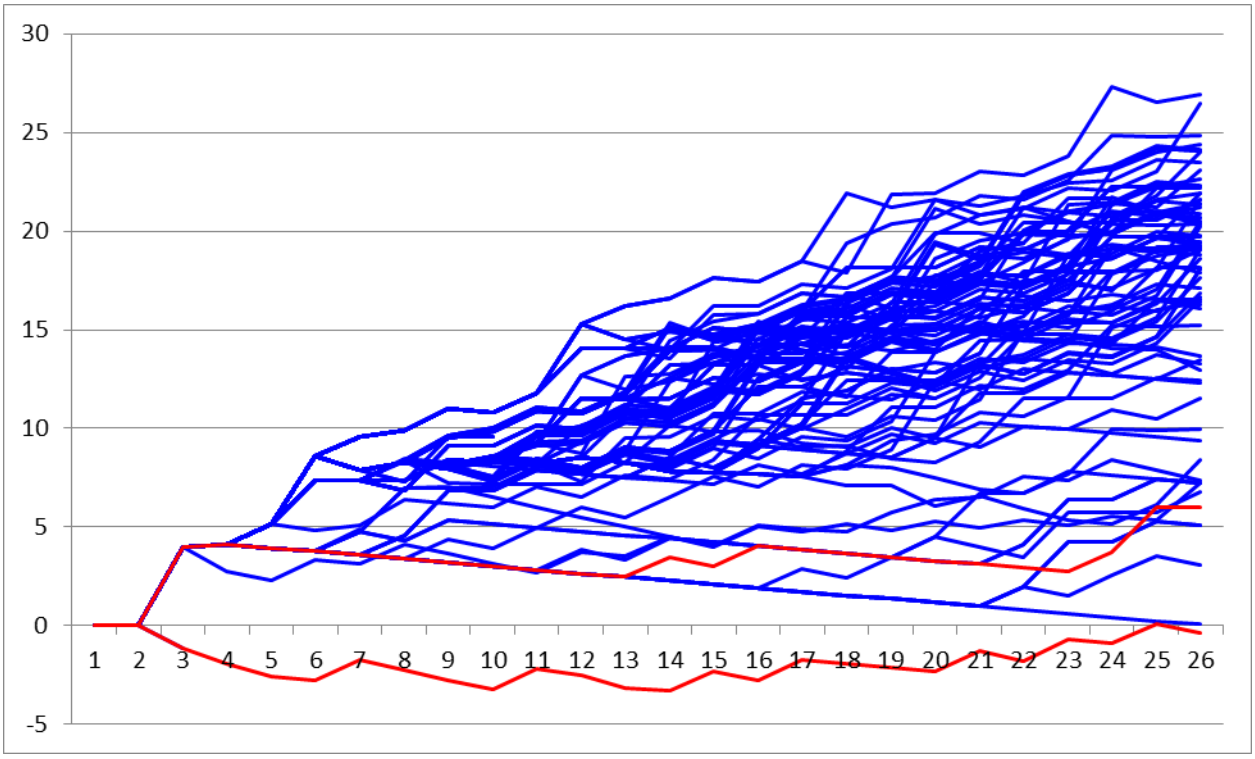

Figure D15. Order Two, Two class, Unweighted

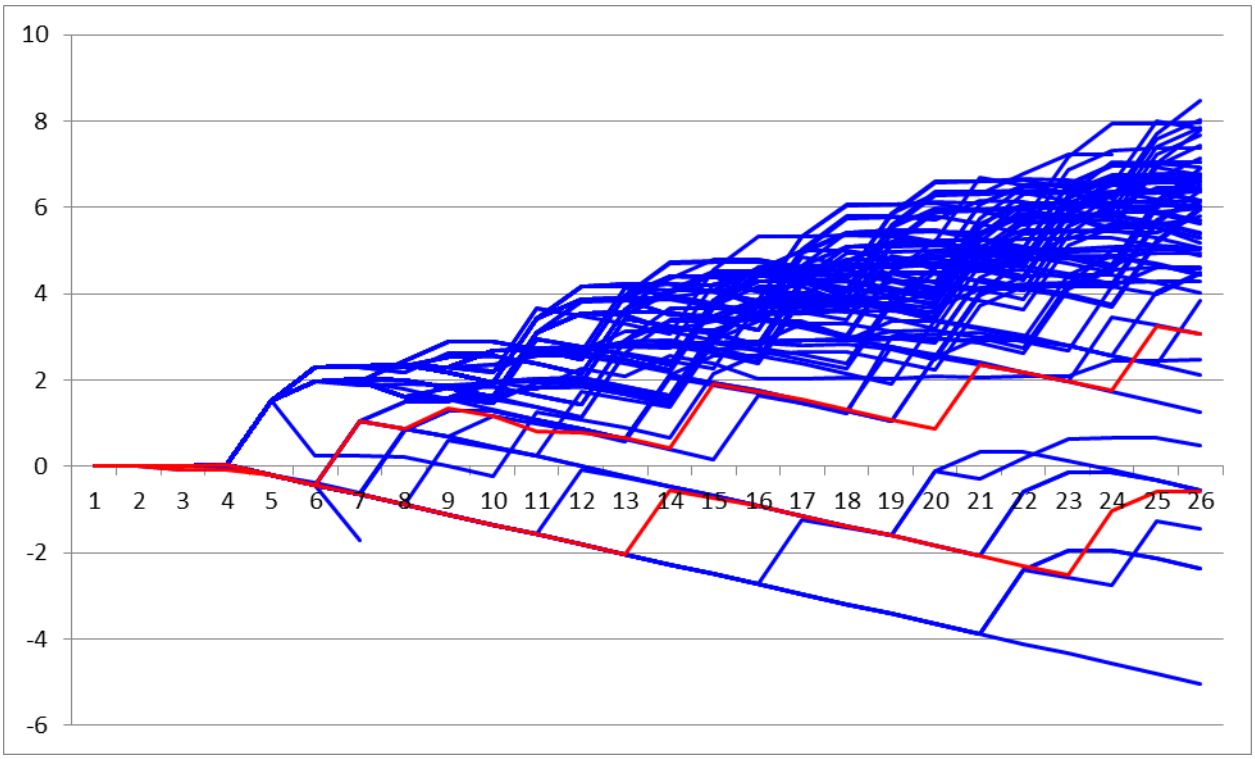

Figure D16. Order Two, Two class, Weighted

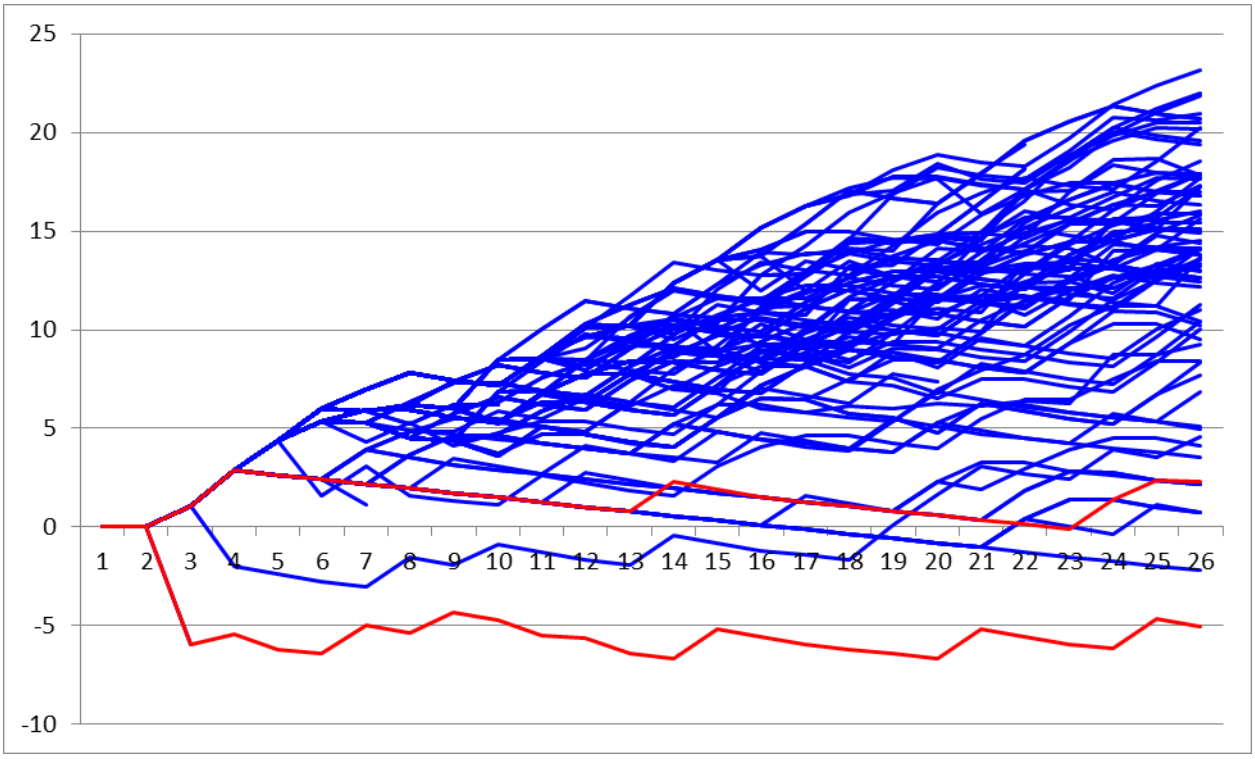

Figure D17. Order Two, Multi class, Unweighted

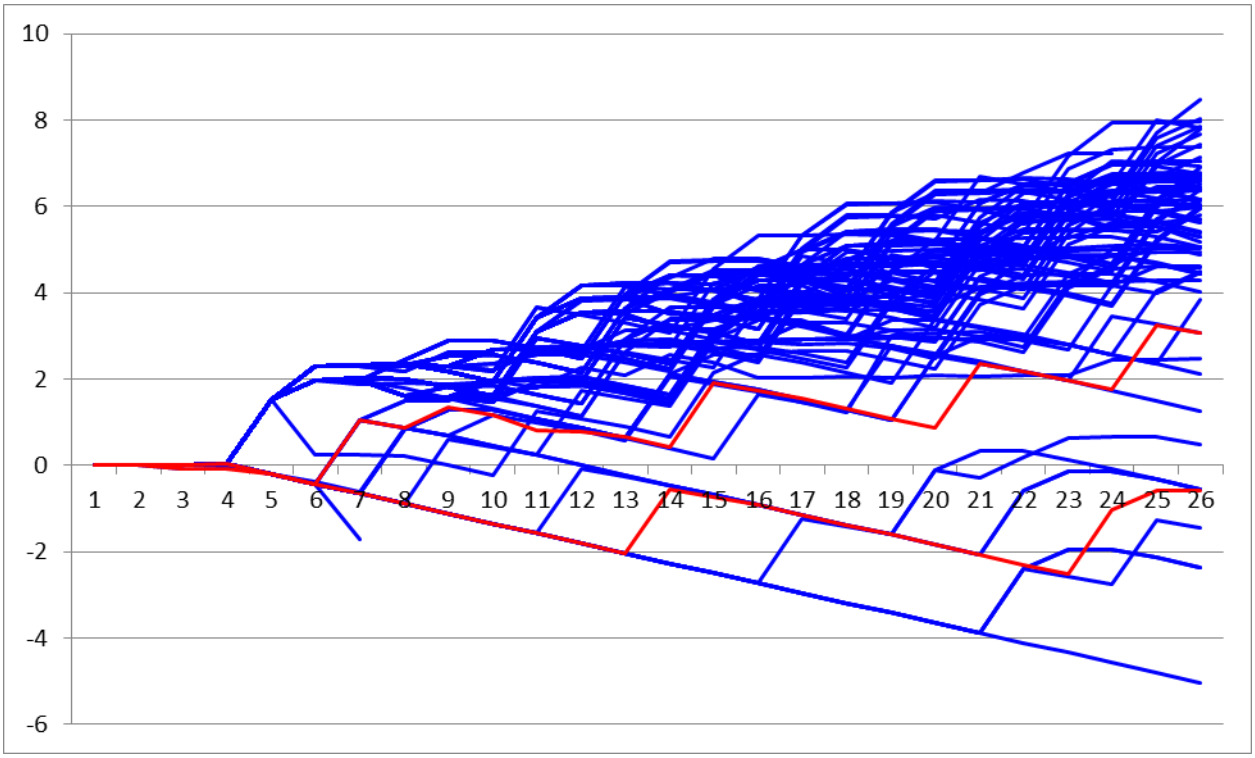

Figure D18. Order Two, Multi class, Weighted

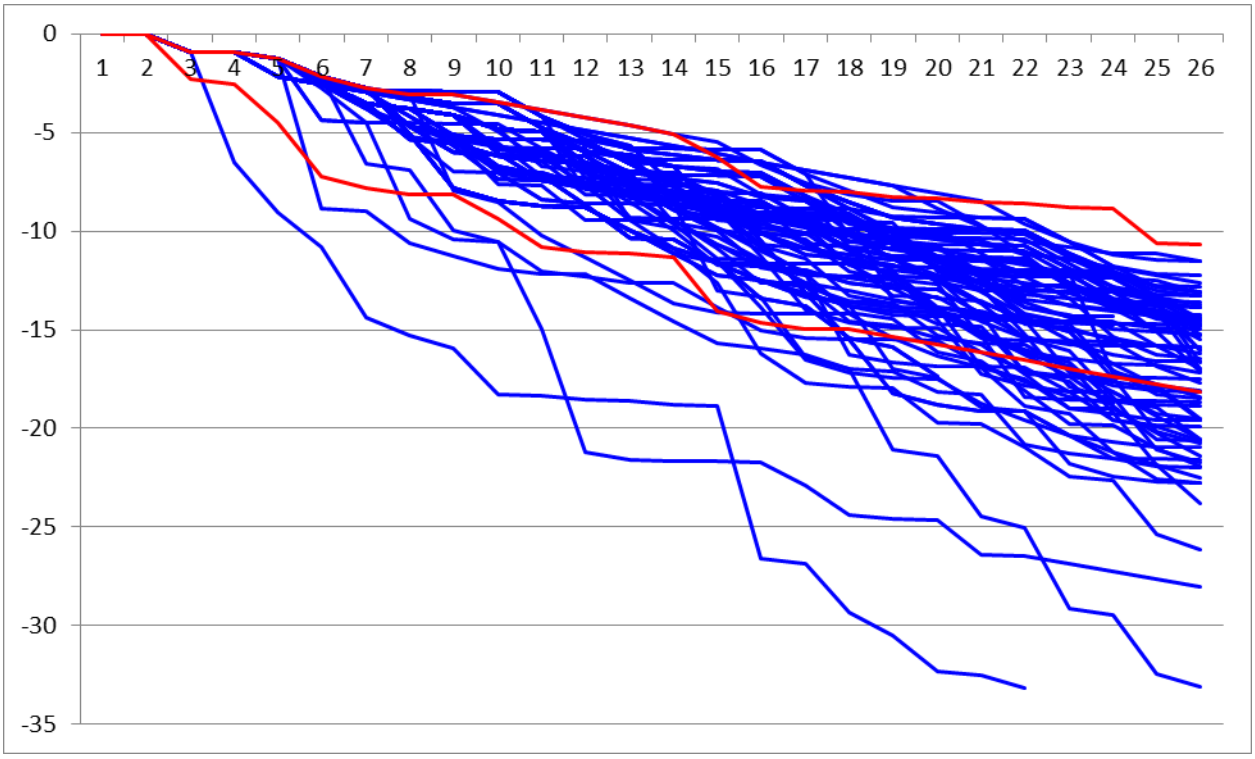

Figure D19. Order Three, One class, Unweighted

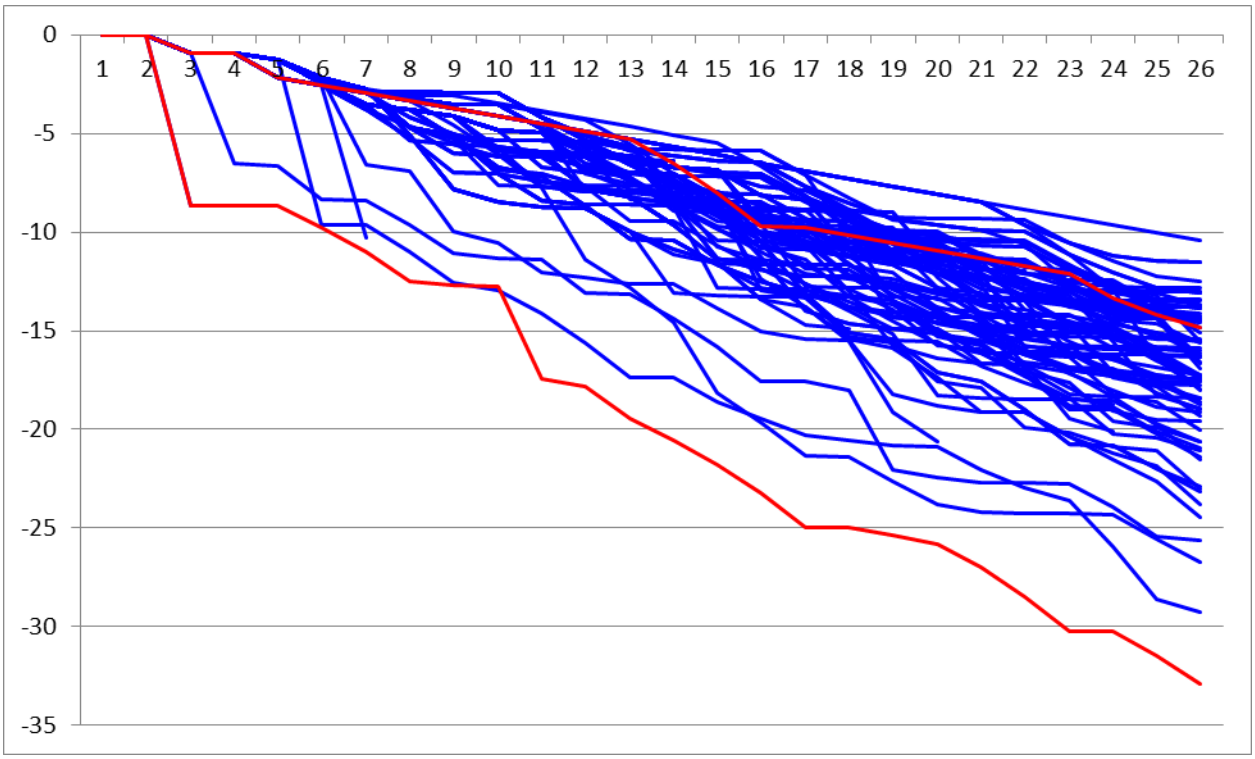

Figure D20. Order Three, One class, Weighted

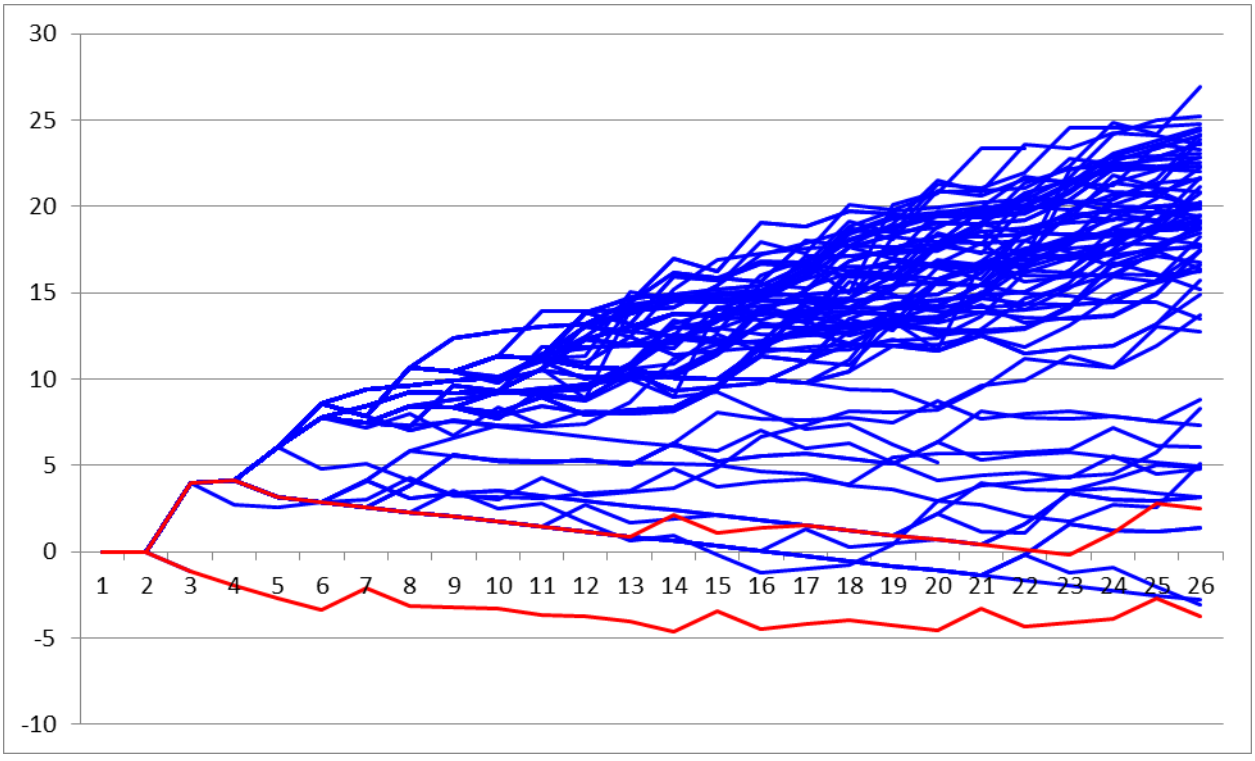

Figure D21. Order Three, Two class, Unweighted

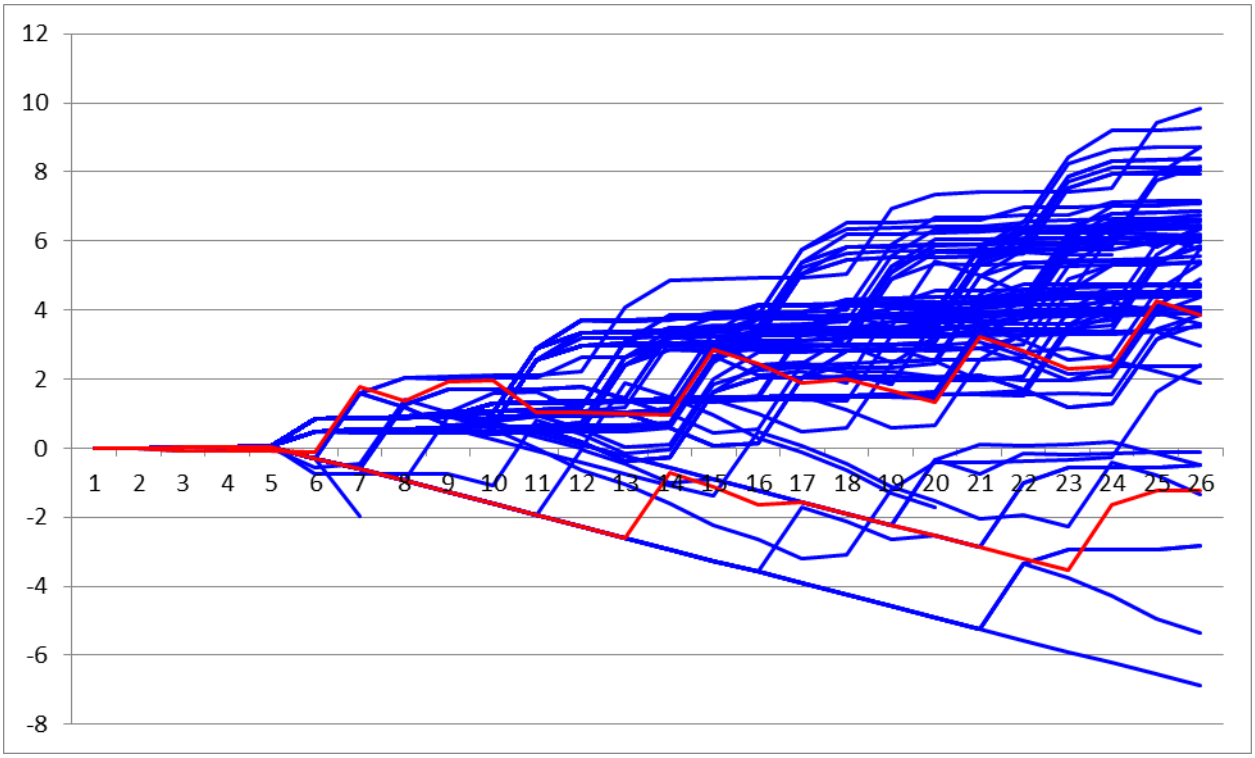

Figure D22. Order Three, Two class, Weighted

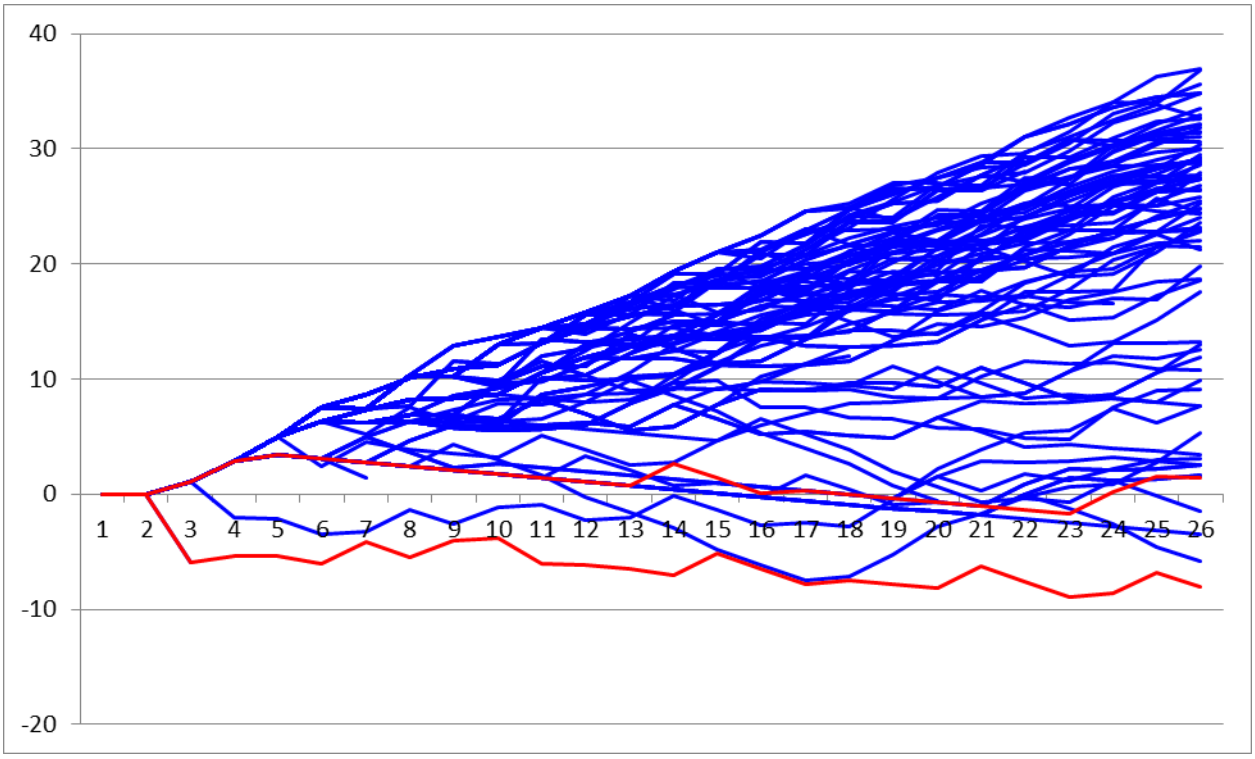

Figure D23. Order Three, Multi class, Unweighted

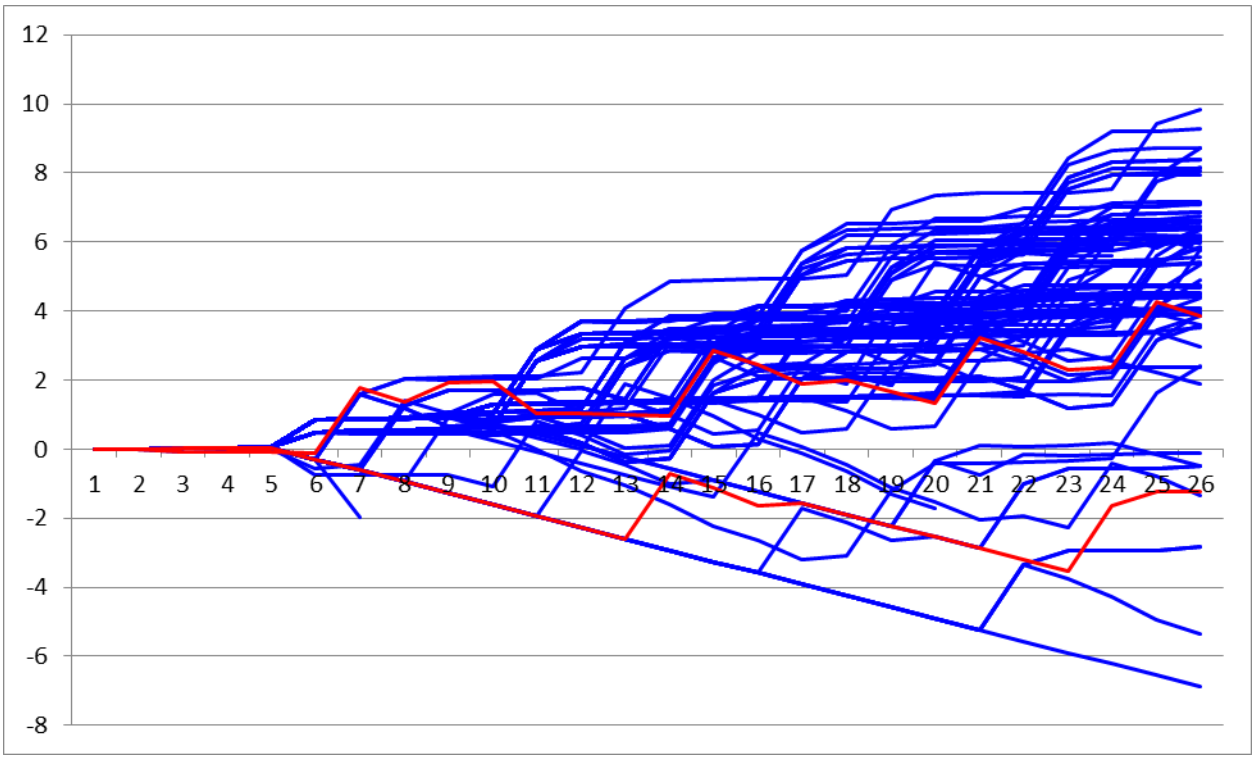

Figure D24. Order Three, Multi class, Weighted

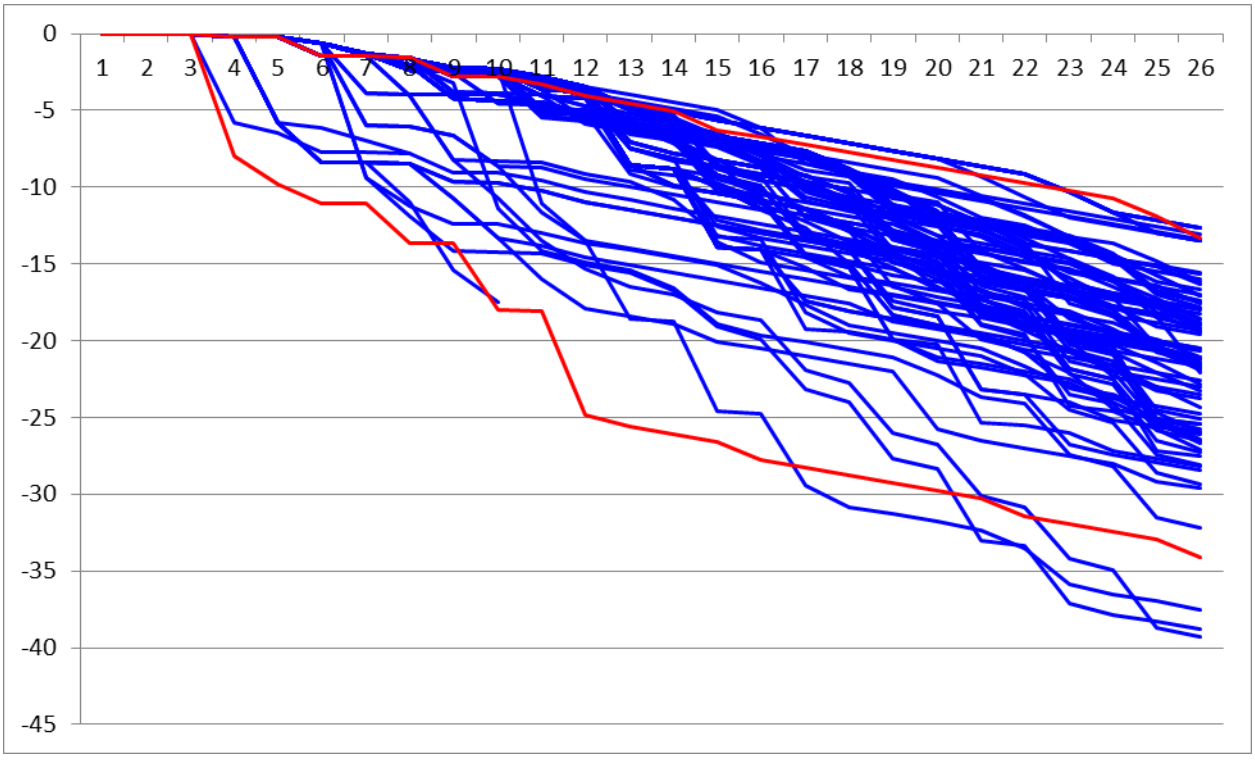

Figure D25. Order One, One class, Unweighted, Pseudo Timing

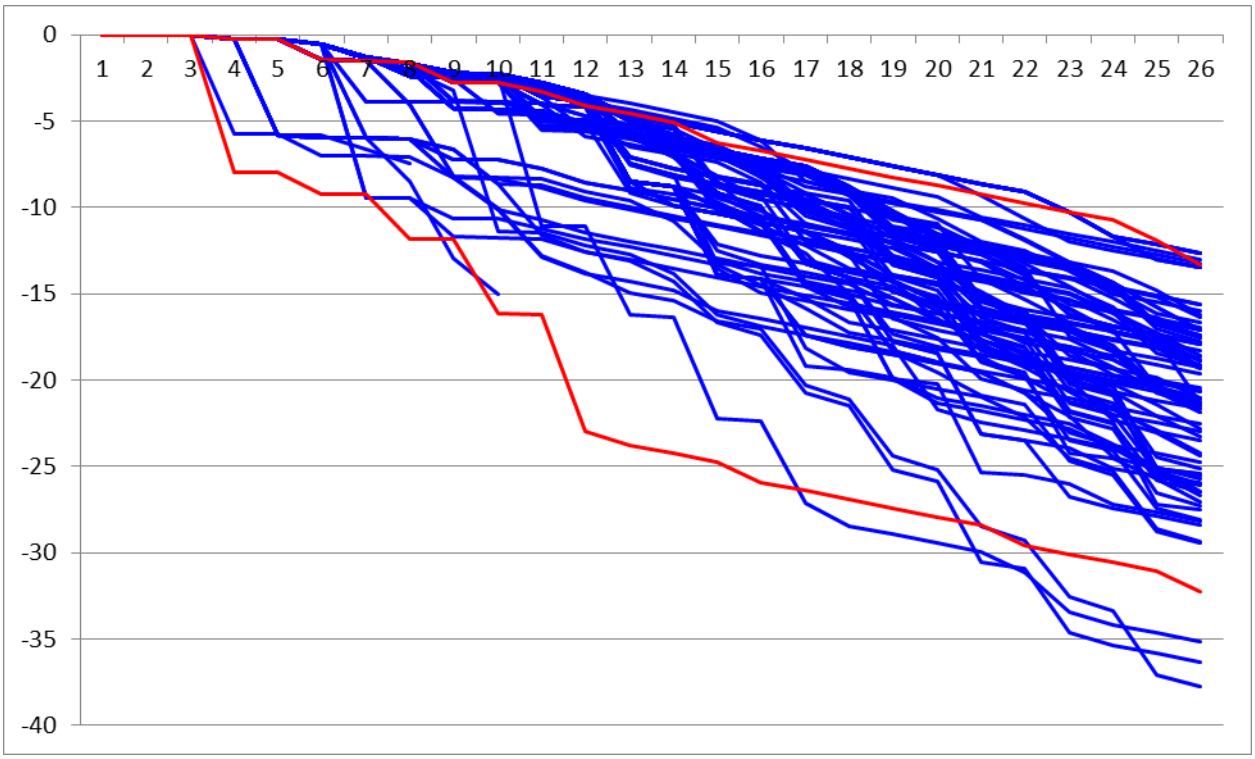

Figure D26. Order One, One class, Weighted, Pseudo Timing

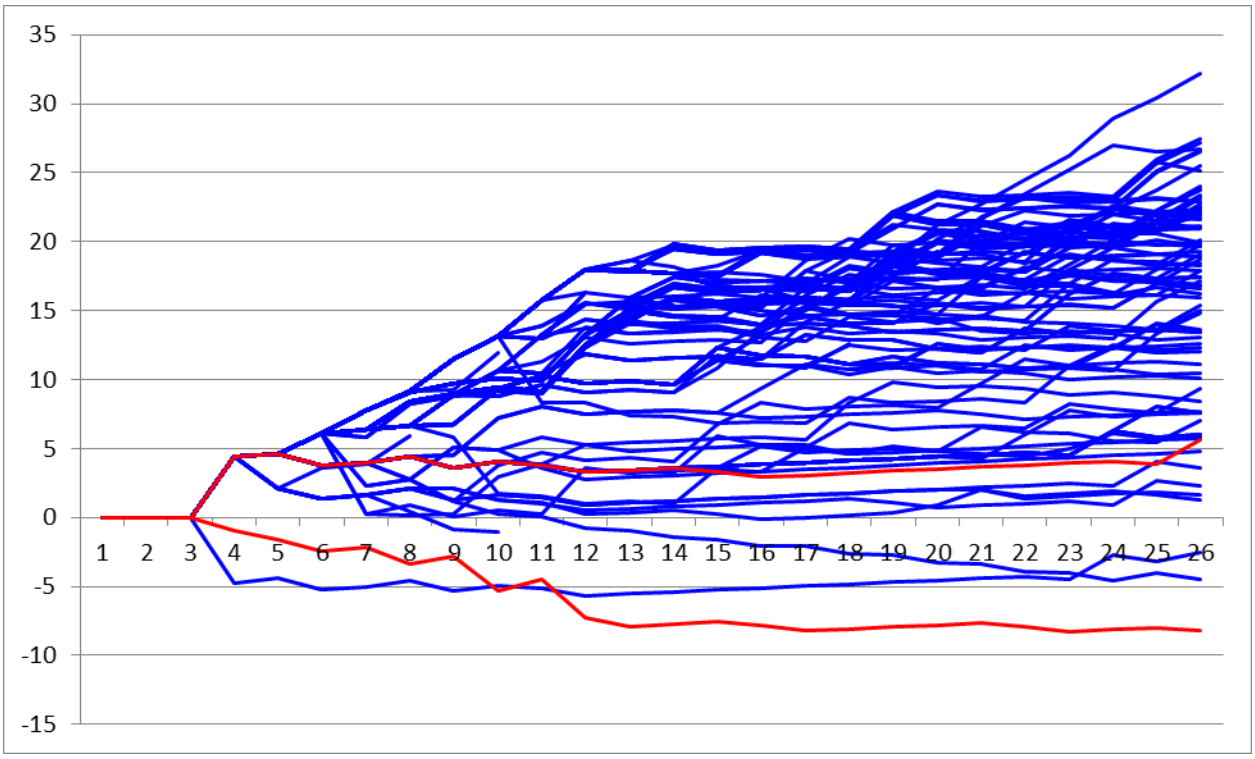

Figure D27. Order One, Two class, Unweighted, Pseudo Timing

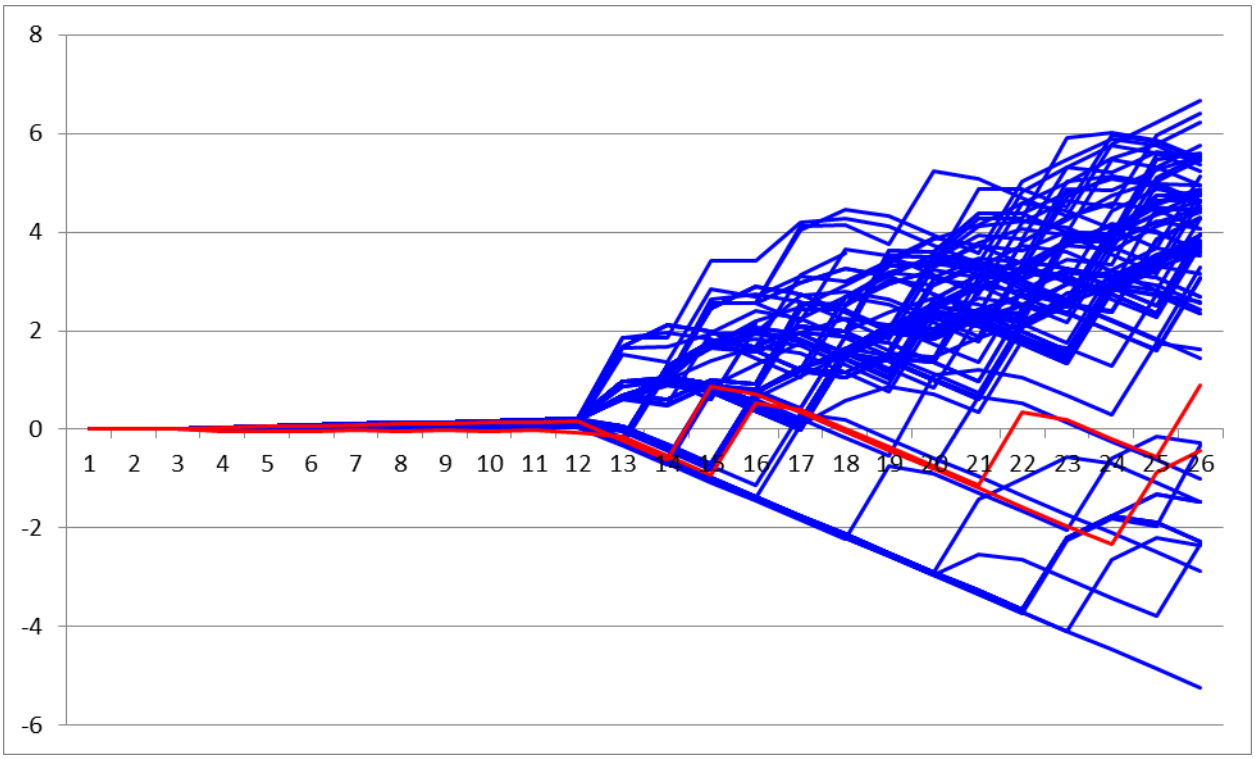

Figure D28. Order One, Two class, Weighted, Pseudo Timing

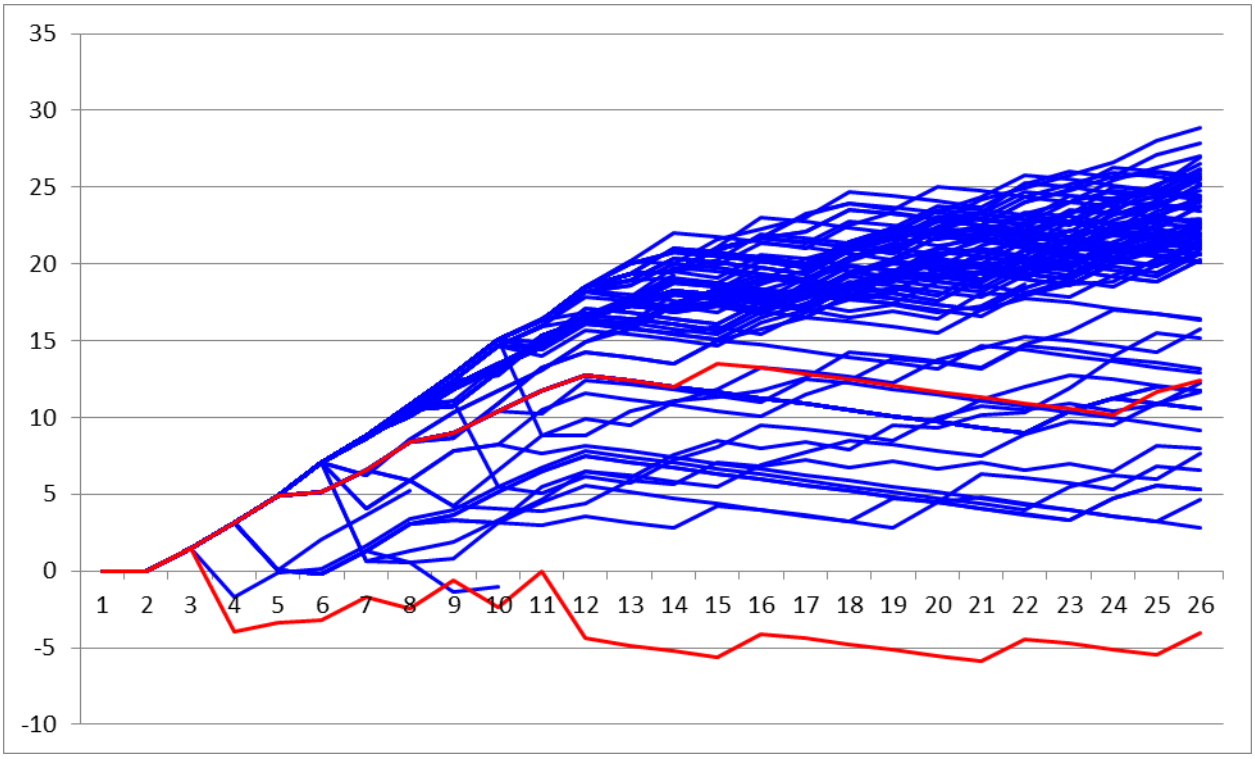

Figure D29. Order One, Multi class, Unweighted, Pseudo Timing

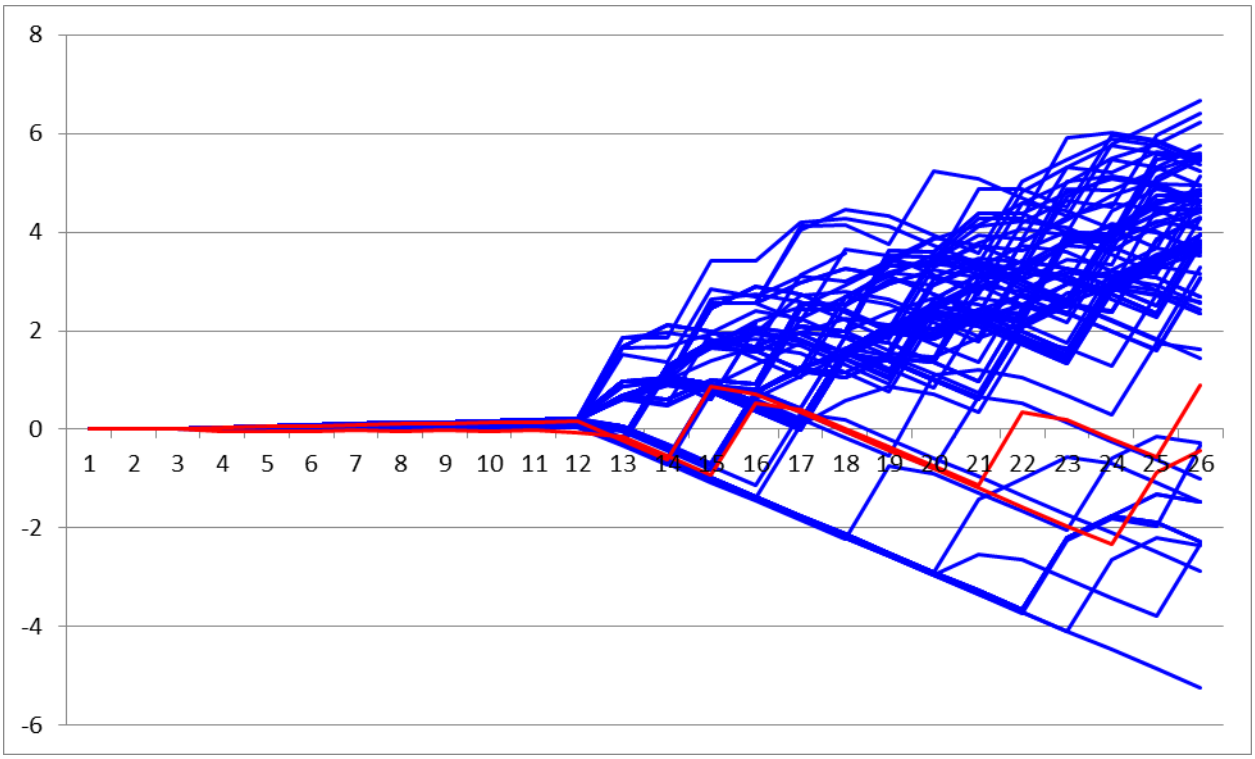

Figure D30. Order One, Multi class, Weighted, Pseudo Timing

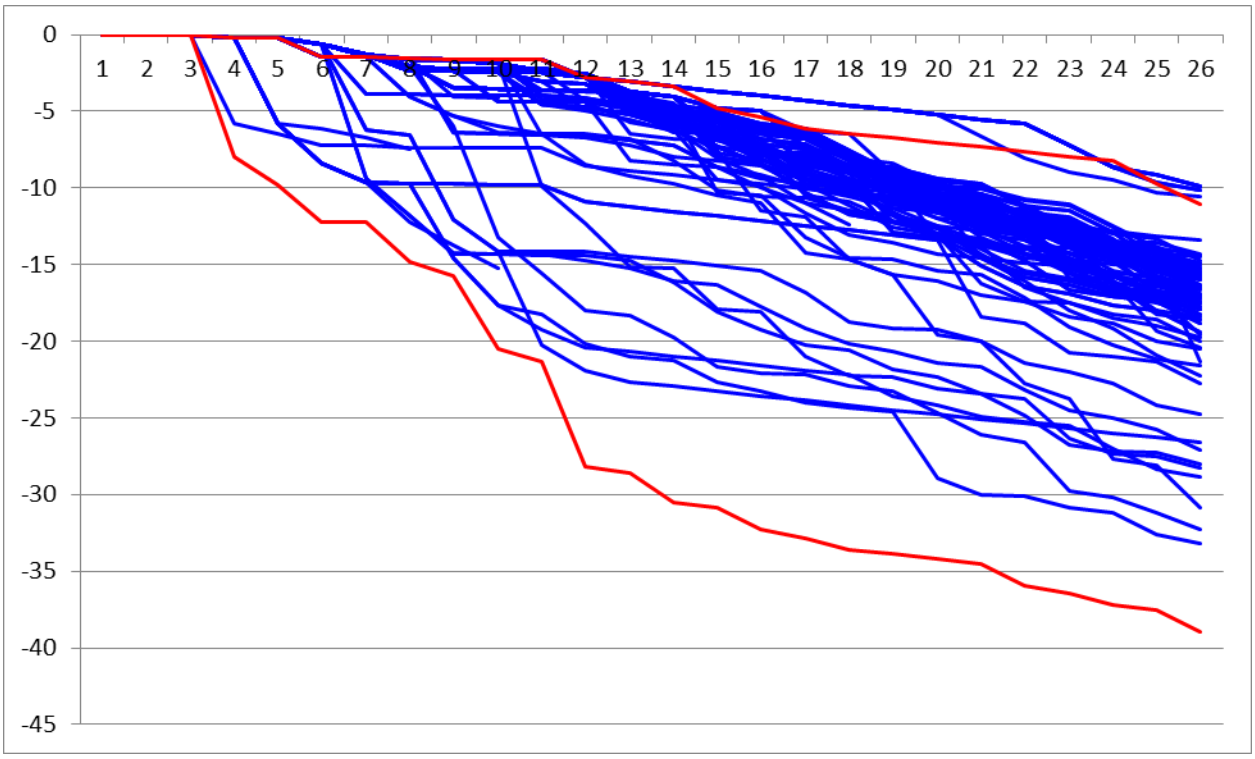

Figure D31. Order Two, One class, Unweighted, Pseudo Timing

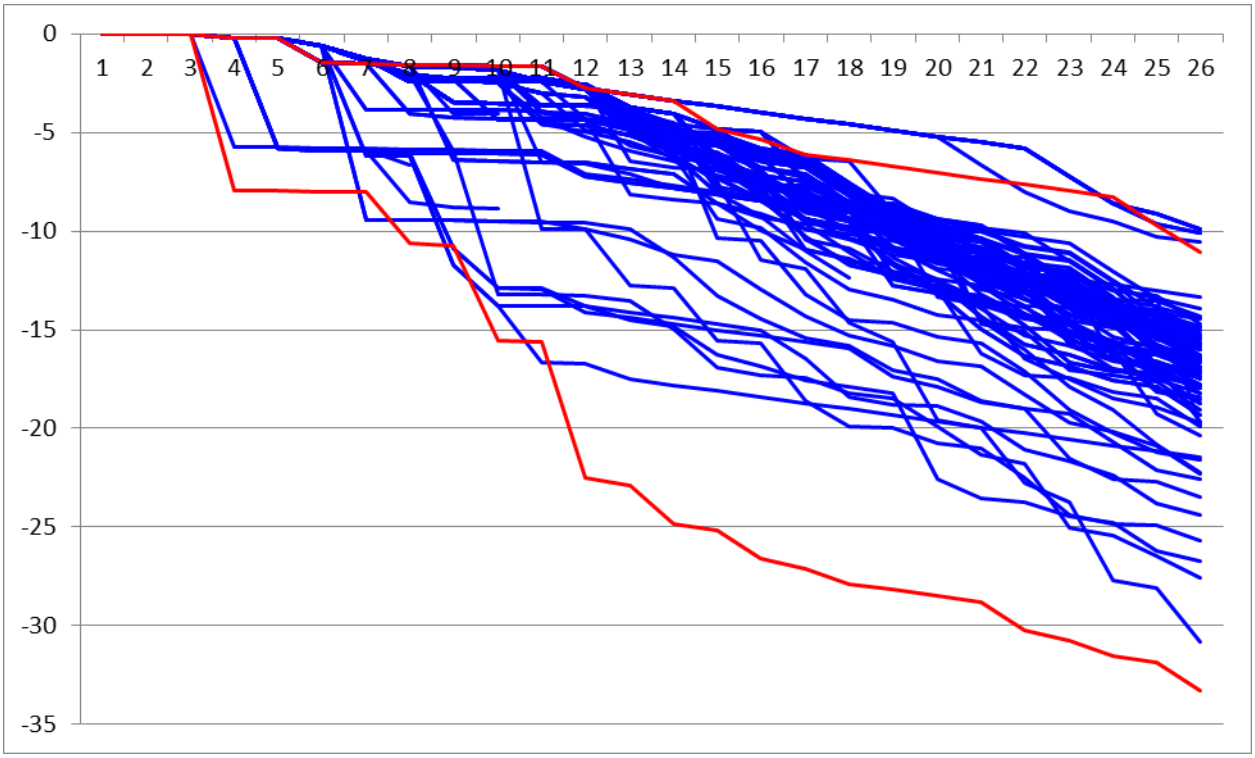

Figure D32. Order Two, One class, Weighted, Pseudo Timing

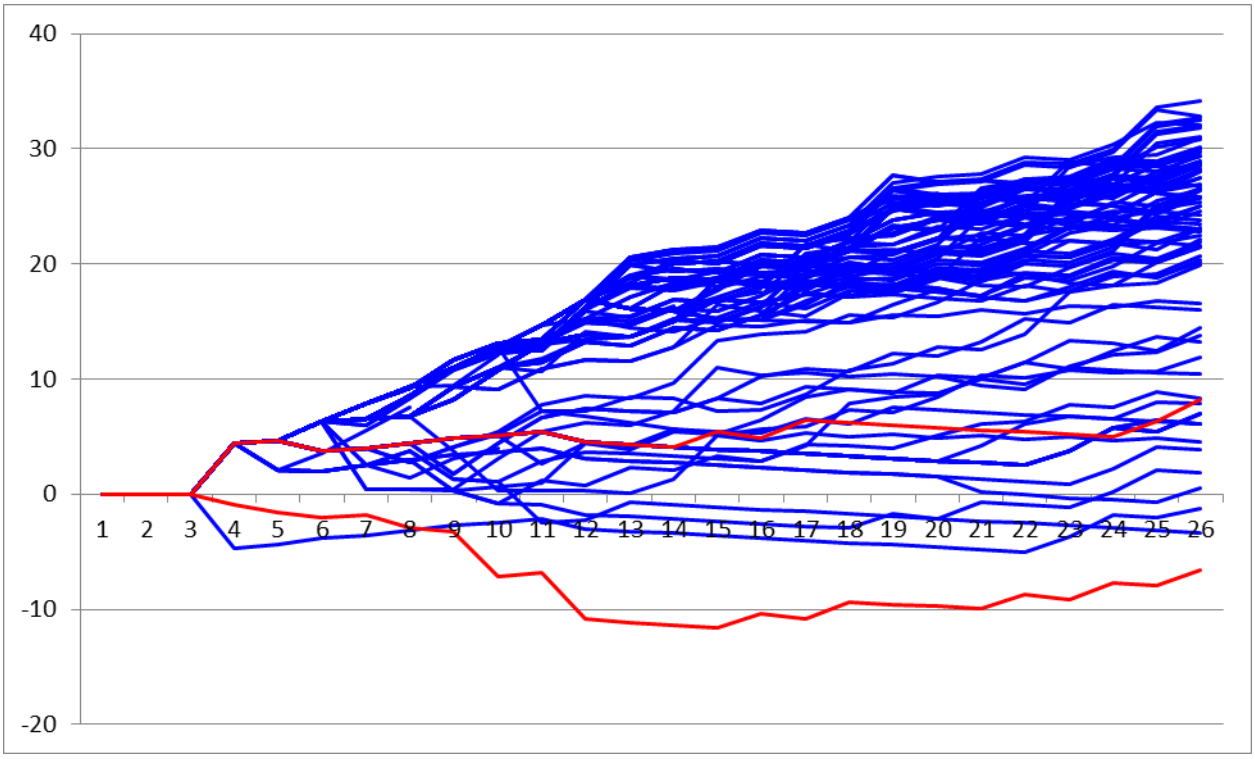

Figure D33. Order Two, Two class, Unweighted, Pseudo Timing

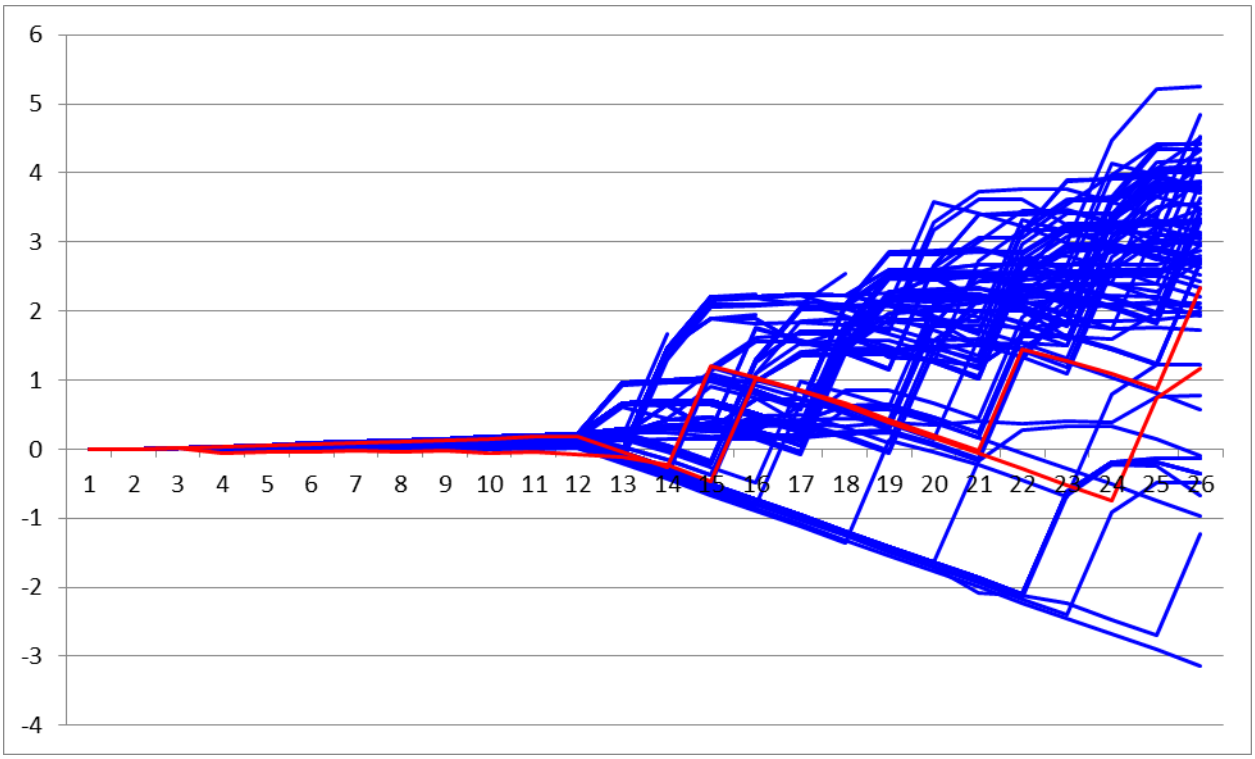

Figure D34. Order Two, Two class, Weighted, Pseudo Timing

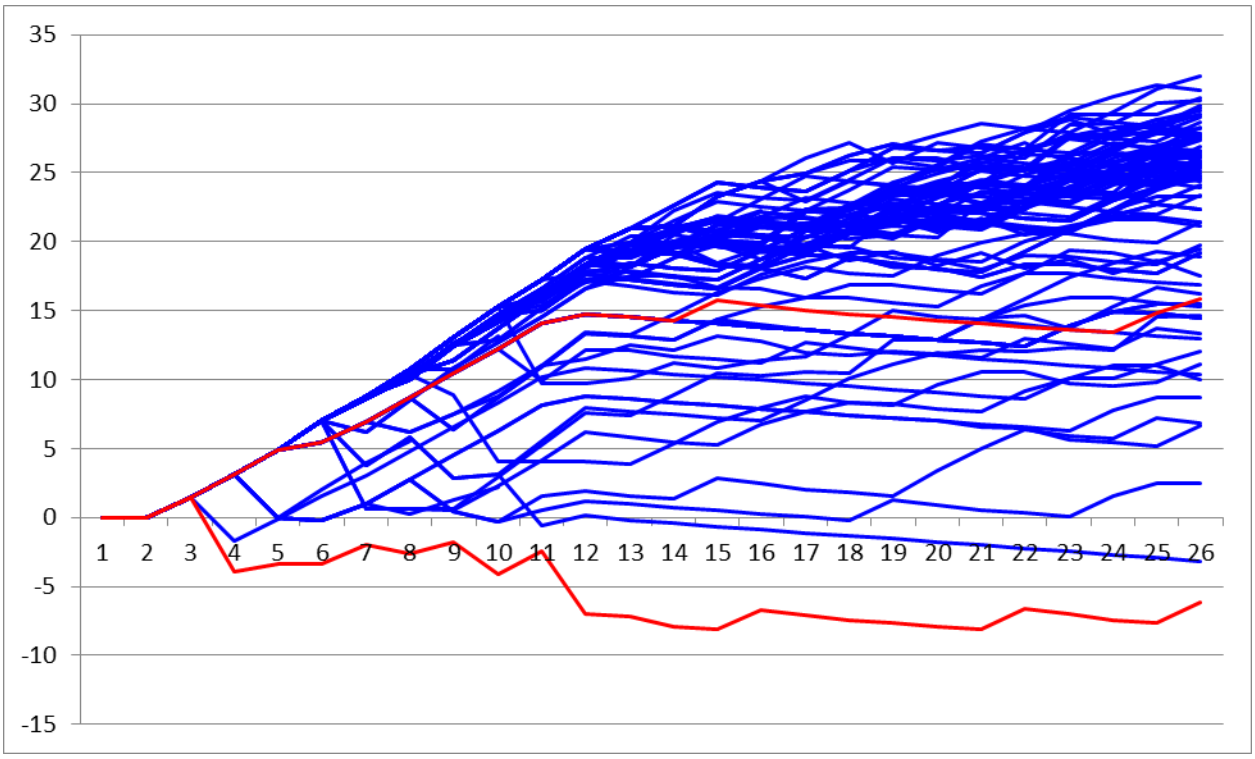

Figure D35. Order Two, Multi class, Unweighted, Pseudo Timing

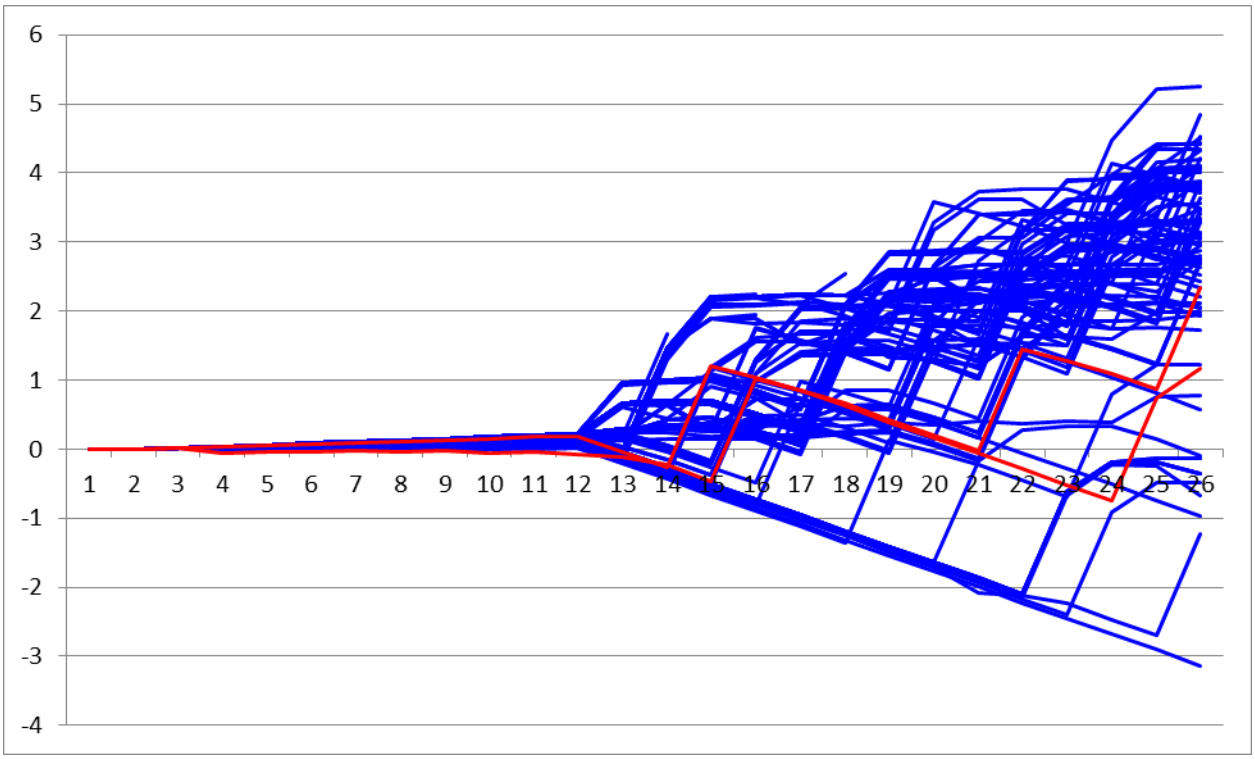

Figure D36. Order Two, Multi class, Weighted, Pseudo Timing

APPENDIX E. WAREZ ATTACK DATASET SPRT GRAPHS

Portseep is a set of attacks from the 1998 DARPA intrusion detection evaluation dataset. Included for reference is an organized listing of SPRT graphs calculated under explained Conditions.

DARPA Description: User logs into anonymous FTP site and creates a hidden directory.

Data Counts

Table E1. Data Counts

	Training	Testing
Flows	21	40
Transitions	50512	179084
Packets	131299	312619

Training Dataset

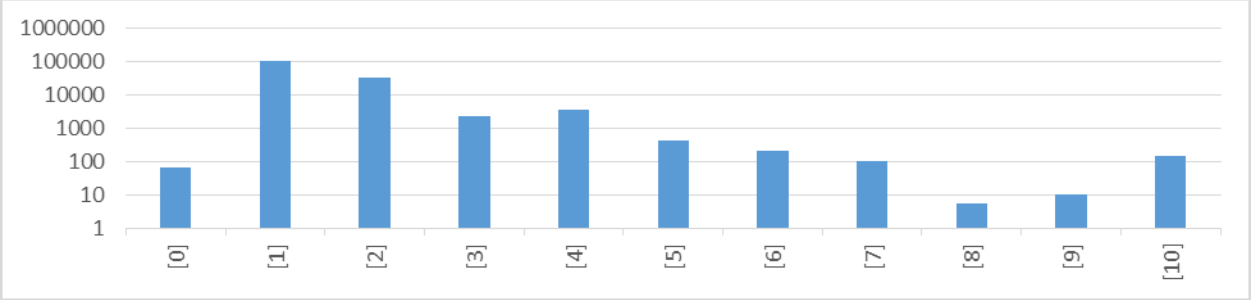

Figure E1. Warez Order 1 Training Dataset Histogram

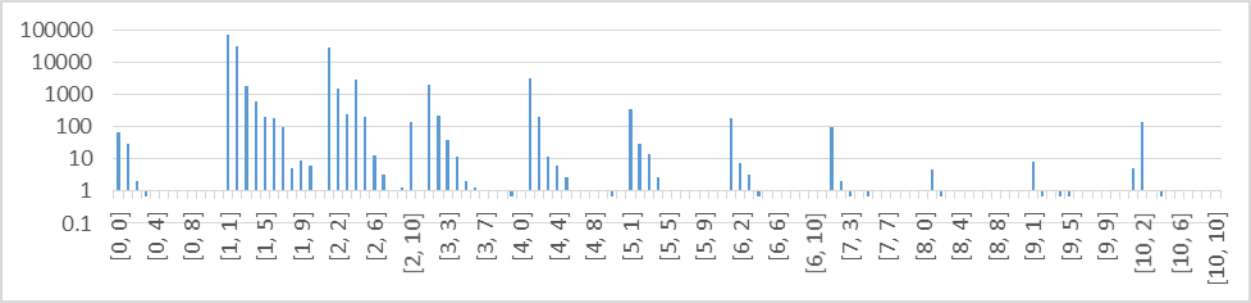

Figure E2. Warez Order 3 Training Dataset Histogram

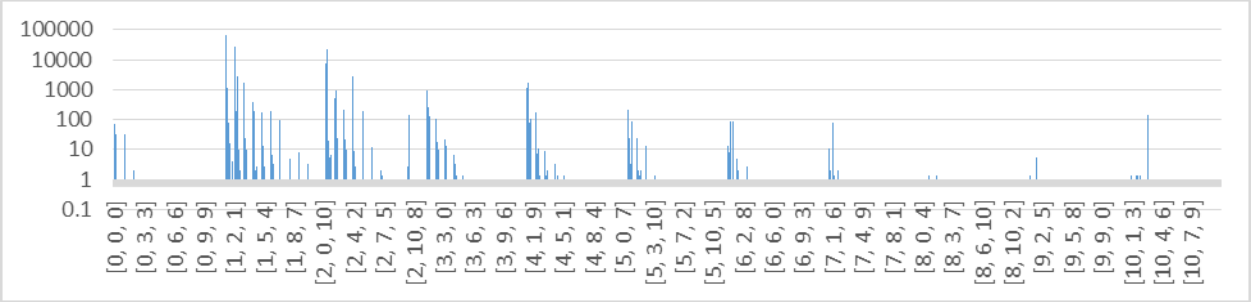

Figure E3. Warez Order 3 Training Dataset Histogram

Testing Dataset Histograms

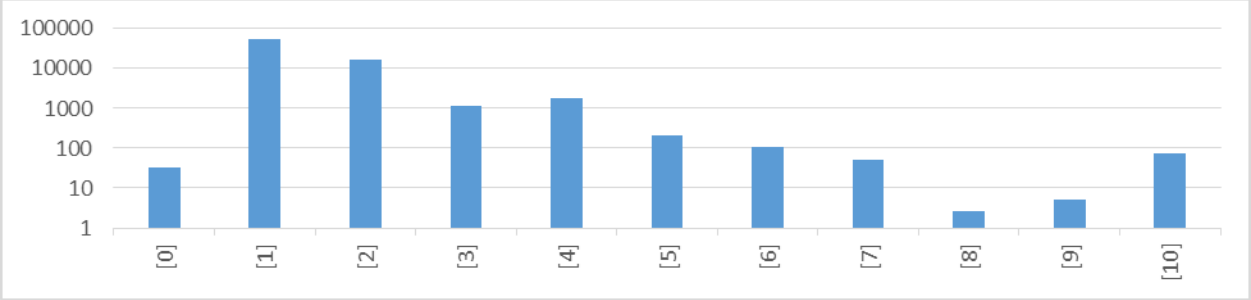

Figure E4. Warez Order 1 Testing Dataset Histogram

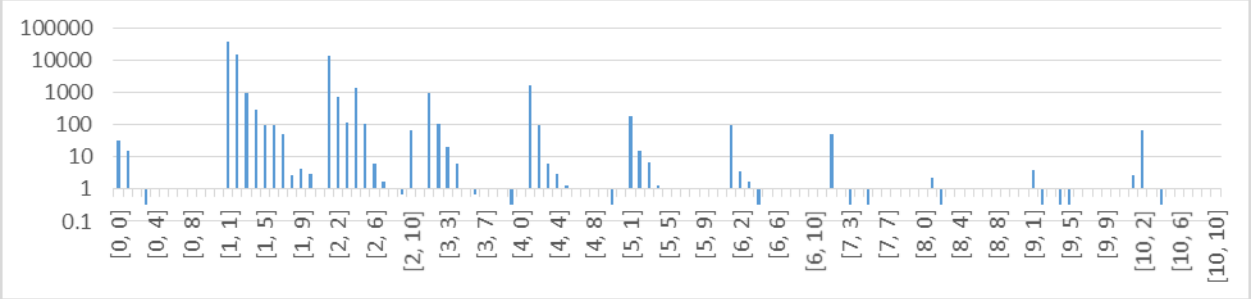

Figure E5. Warez Order 2 Testing Dataset Histogram

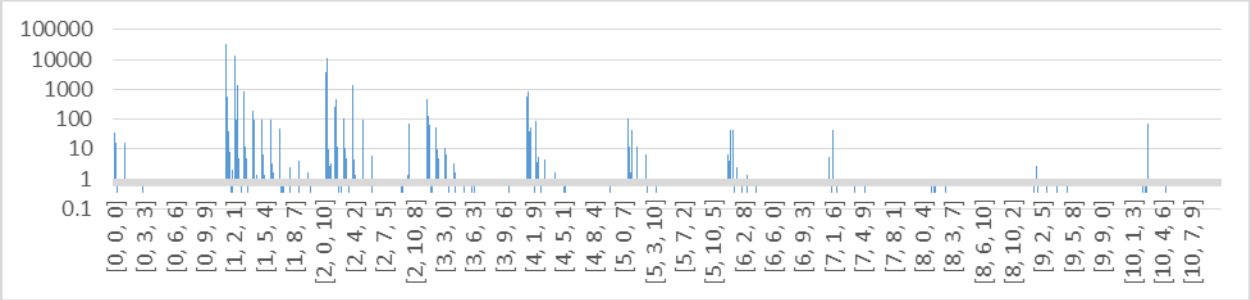

Figure E6. Warez Order 3 Testing Dataset Histogram

SPRT Graphs

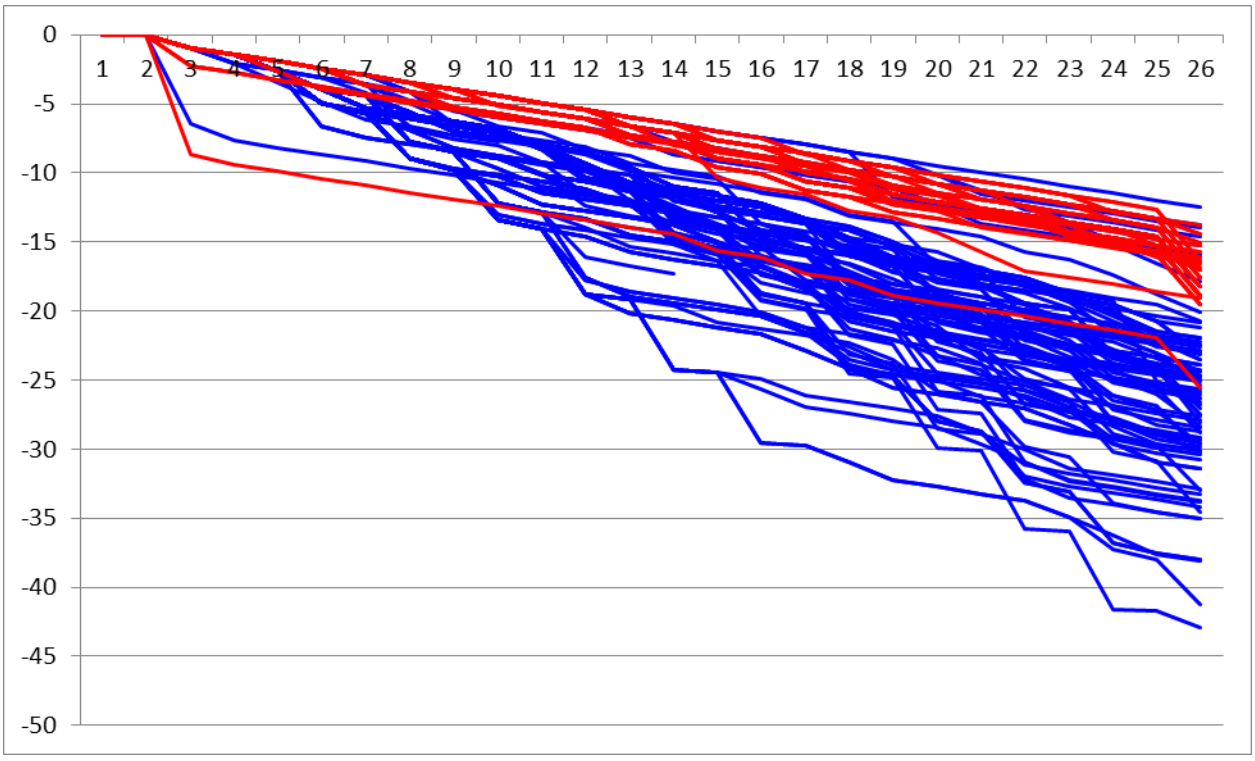

Figure E7. Order One, One class, Unweighted

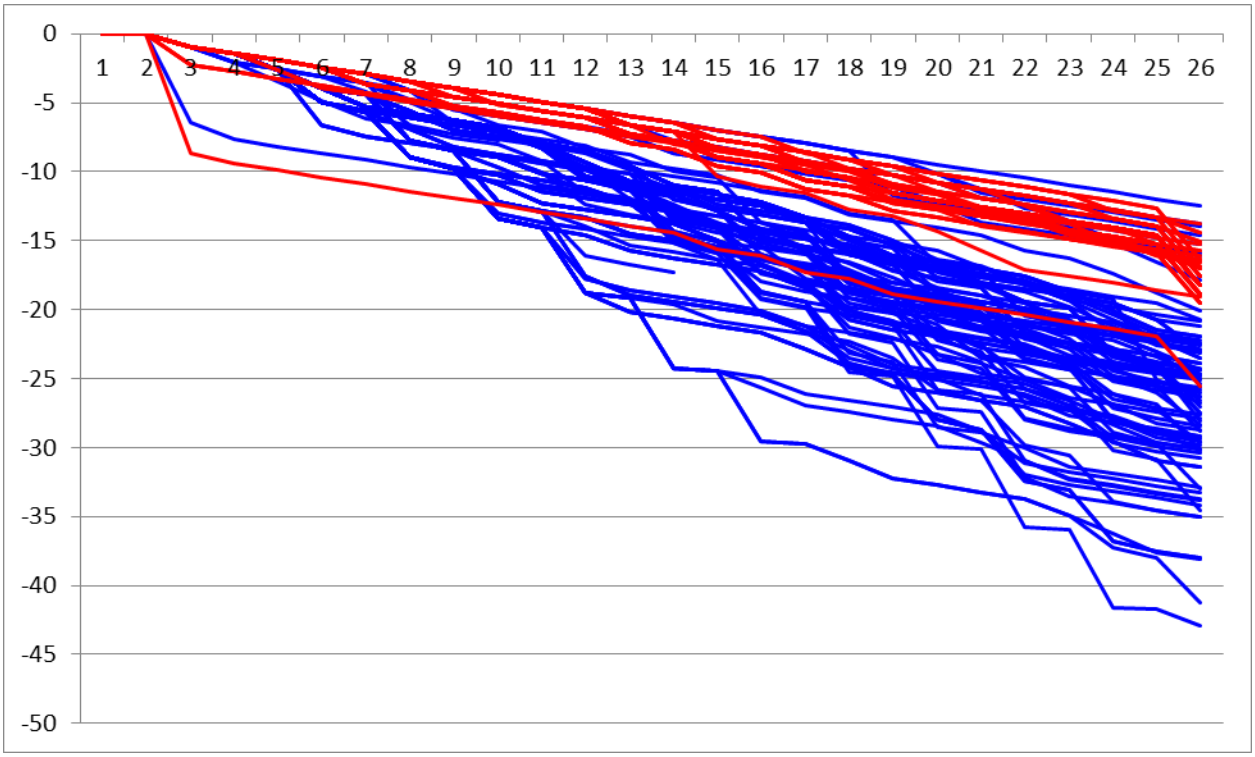

Figure E8.Order One, One class, Weighted

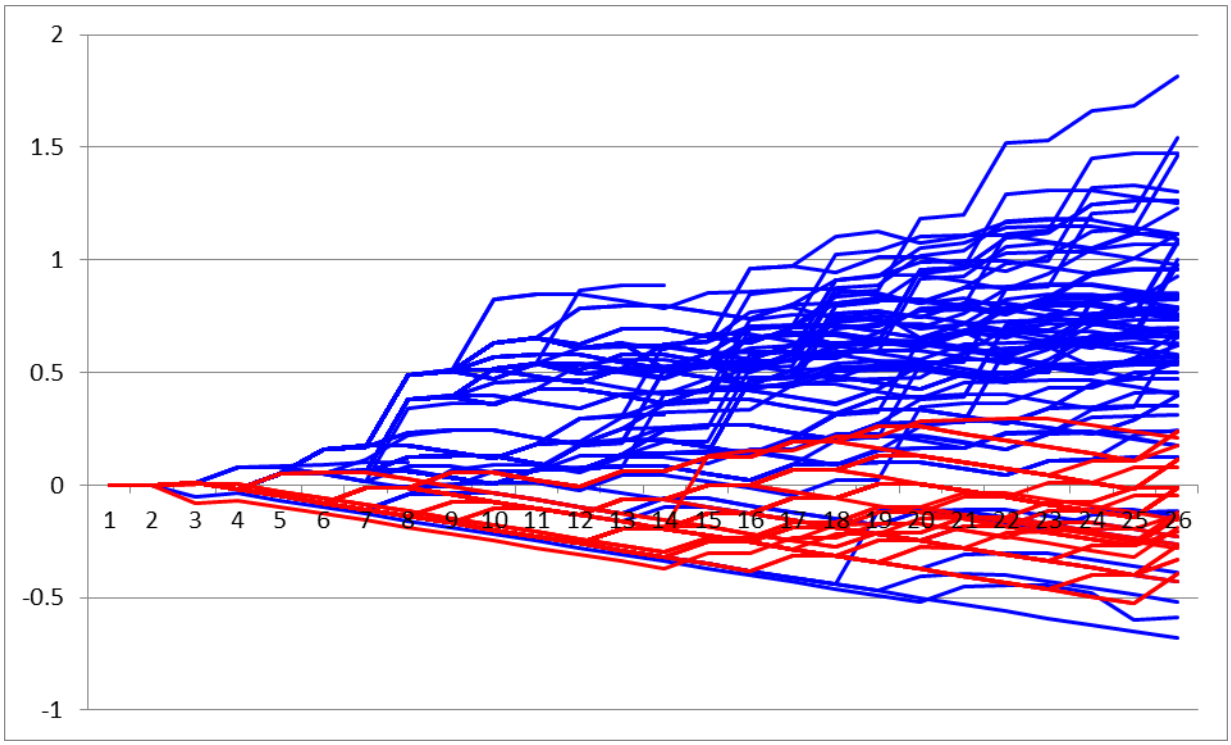

Figure E9. Order One, Two class, Unweighted

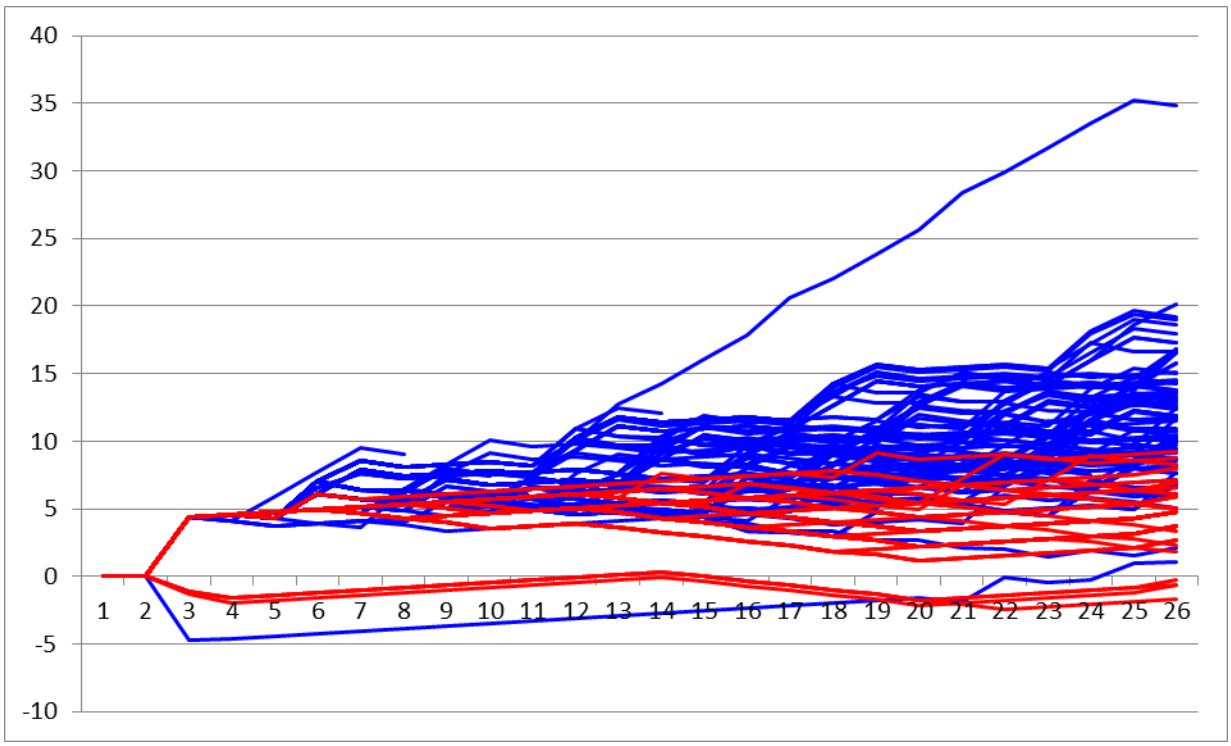

Figure E10. Order One, Two class, Weighted

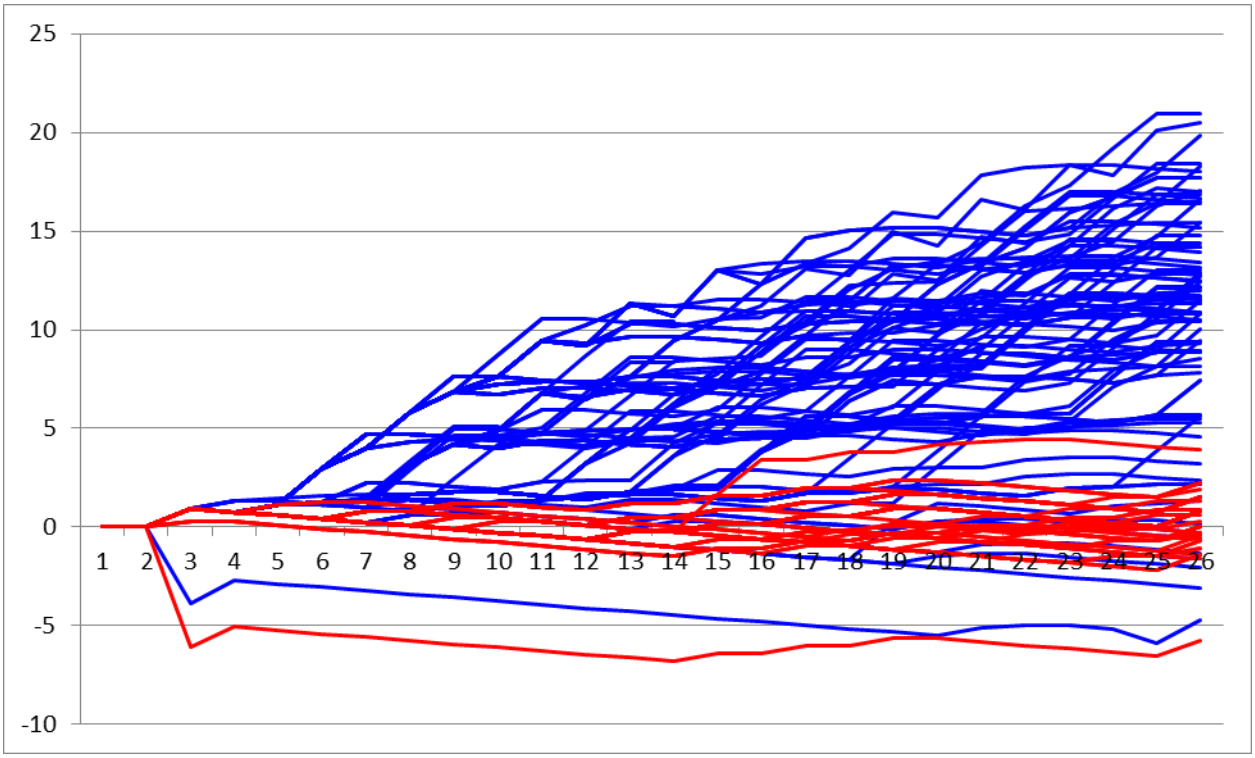

Figure E11. Order One, Multi class, Unweighted

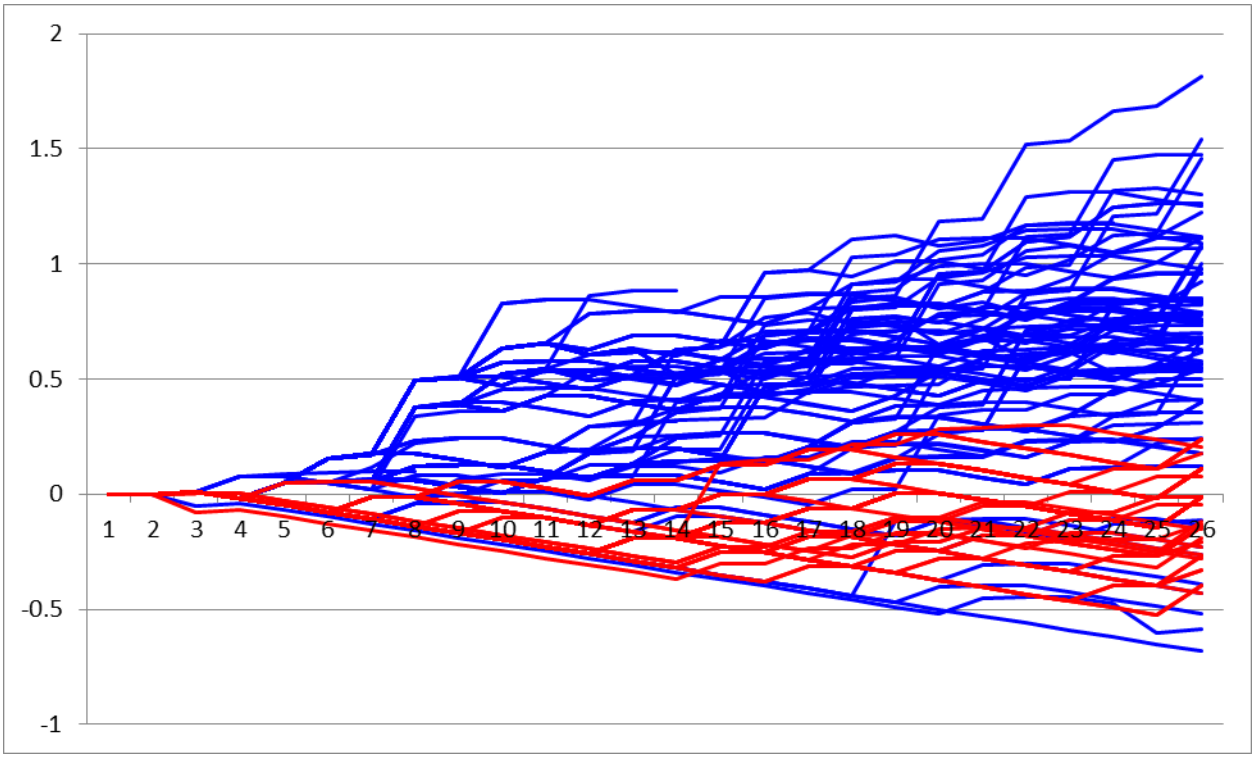

Figure E12. Order One, Multi class, Weighted

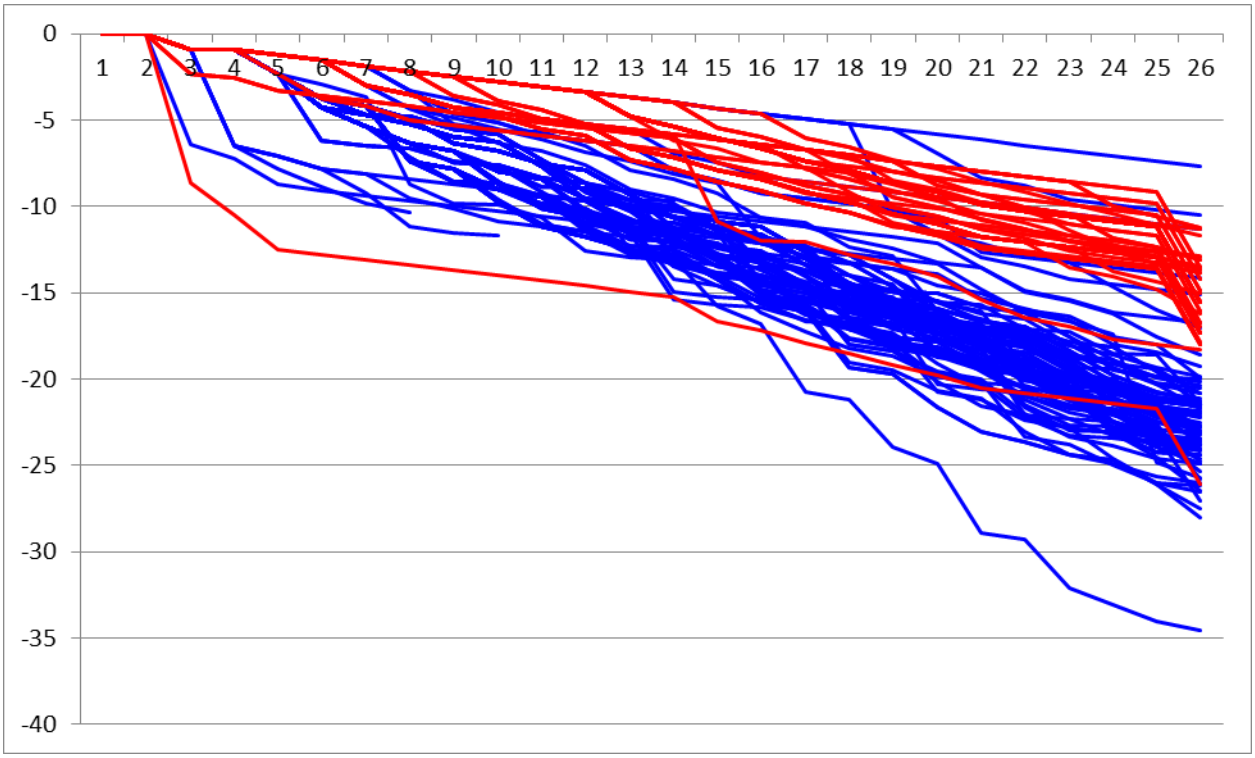

Figure E13. Order Two, One class, Unweighted

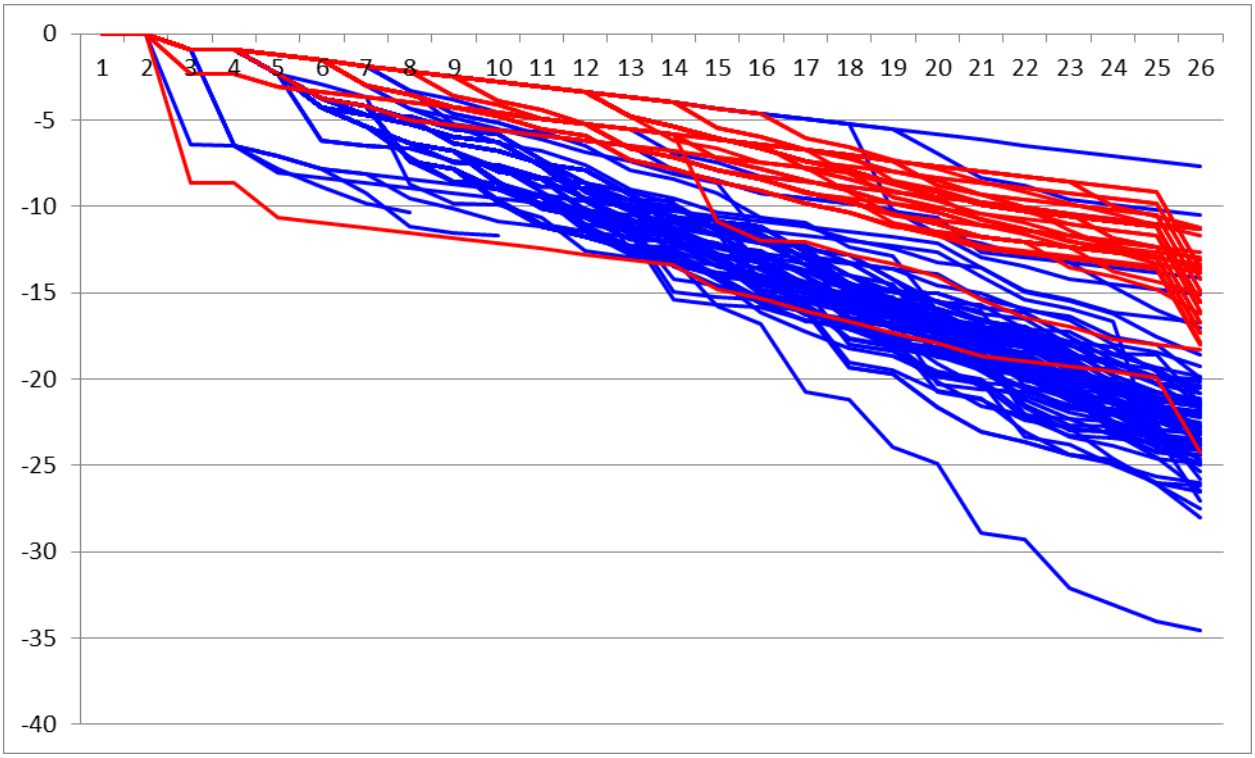

Figure E14. Order Two, One class, Weighted

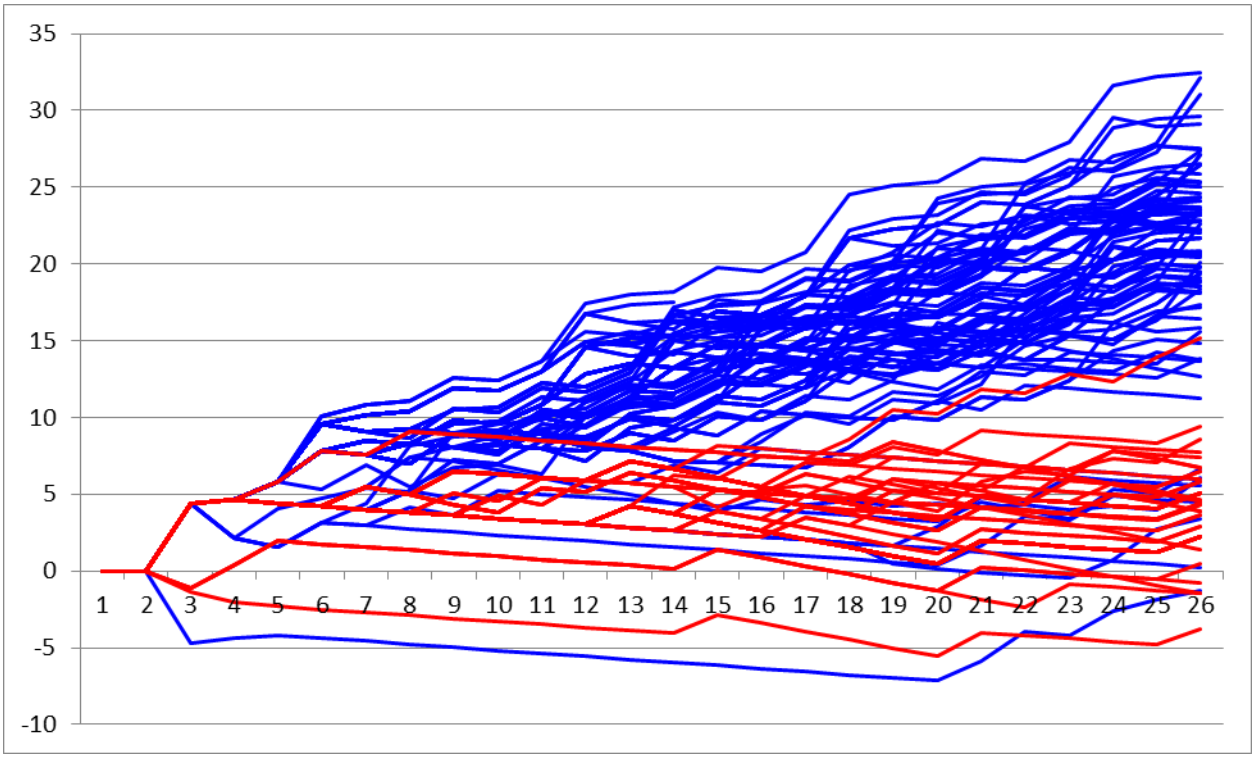

Figure E15. Order Two, Two class, Unweighted

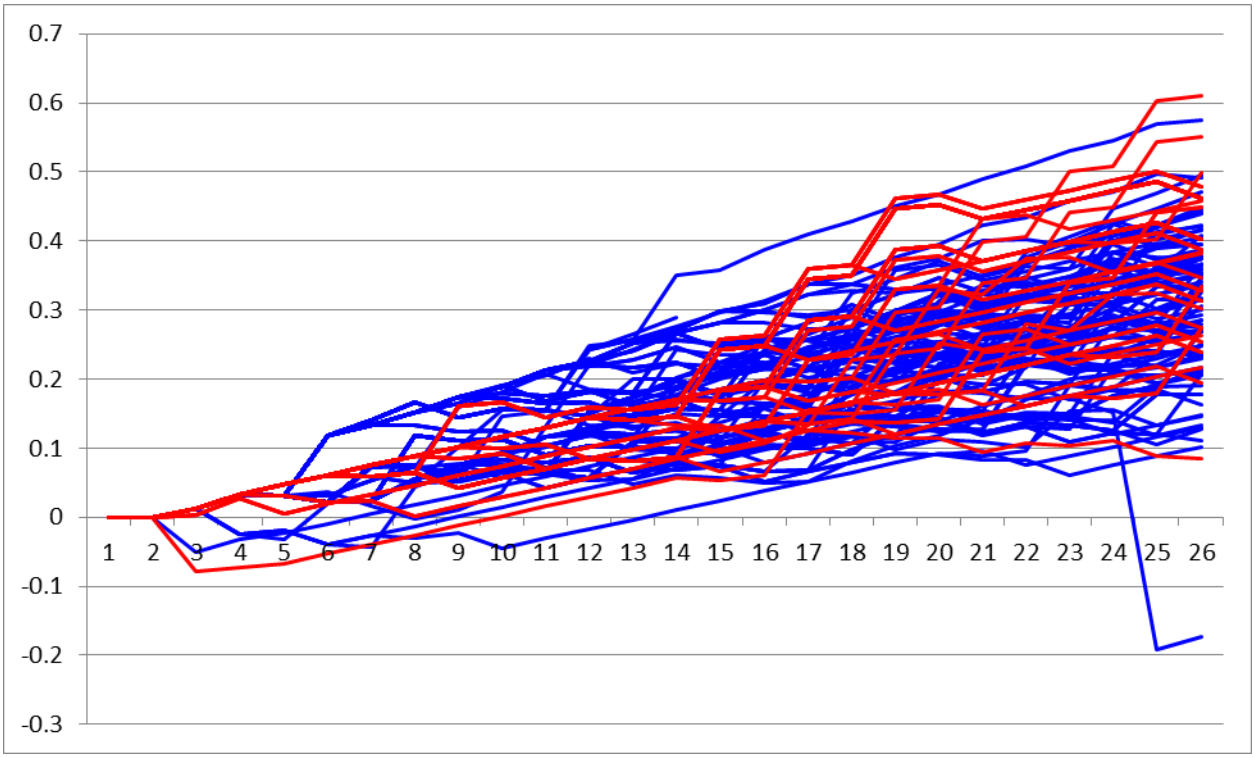

Figure E16. Order Two, Two class, Weighted

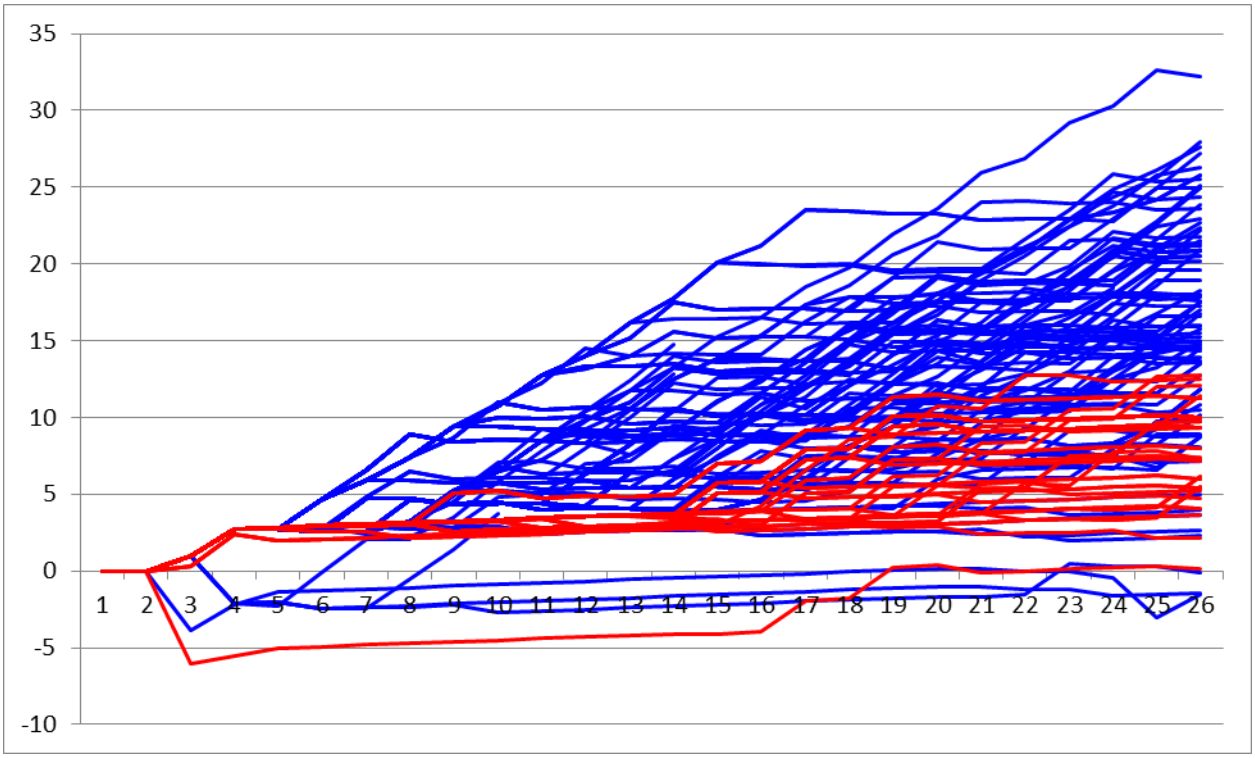

Figure E17. Order Two, Multi class, Unweighted

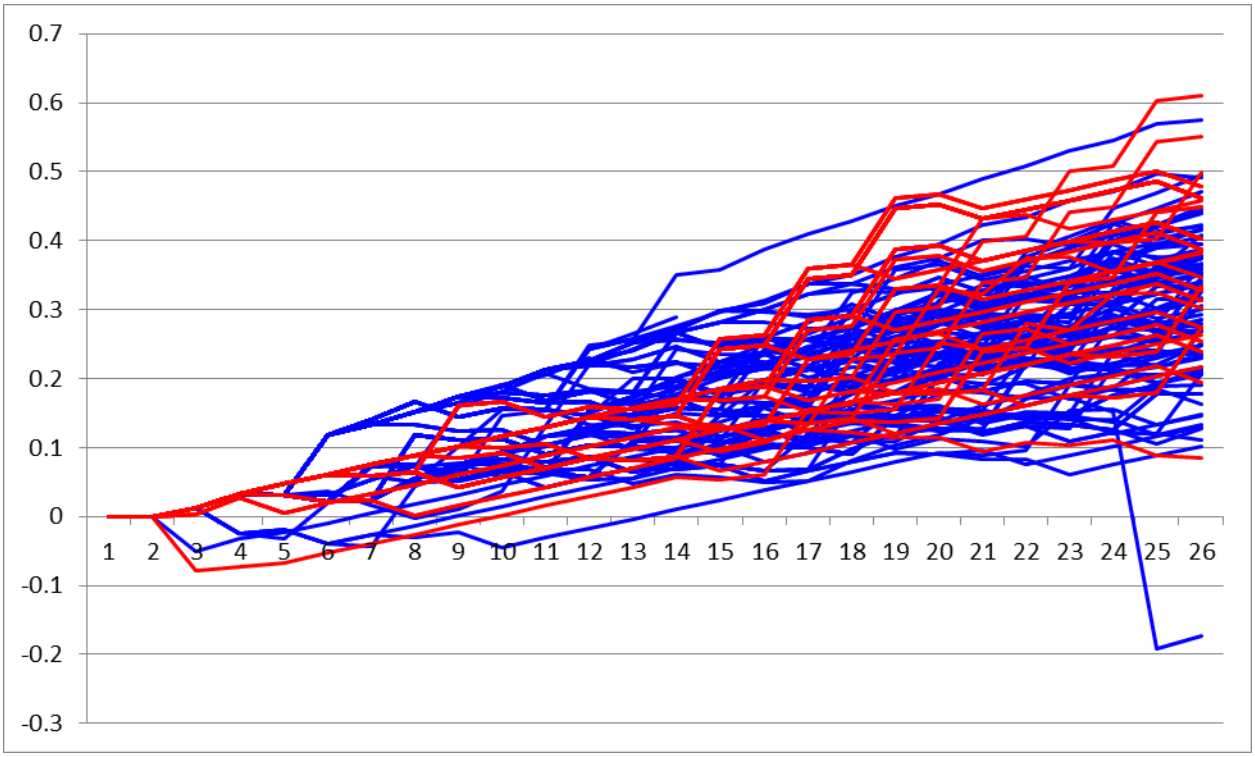

Figure E18. Order Two, Multi class, Weighted

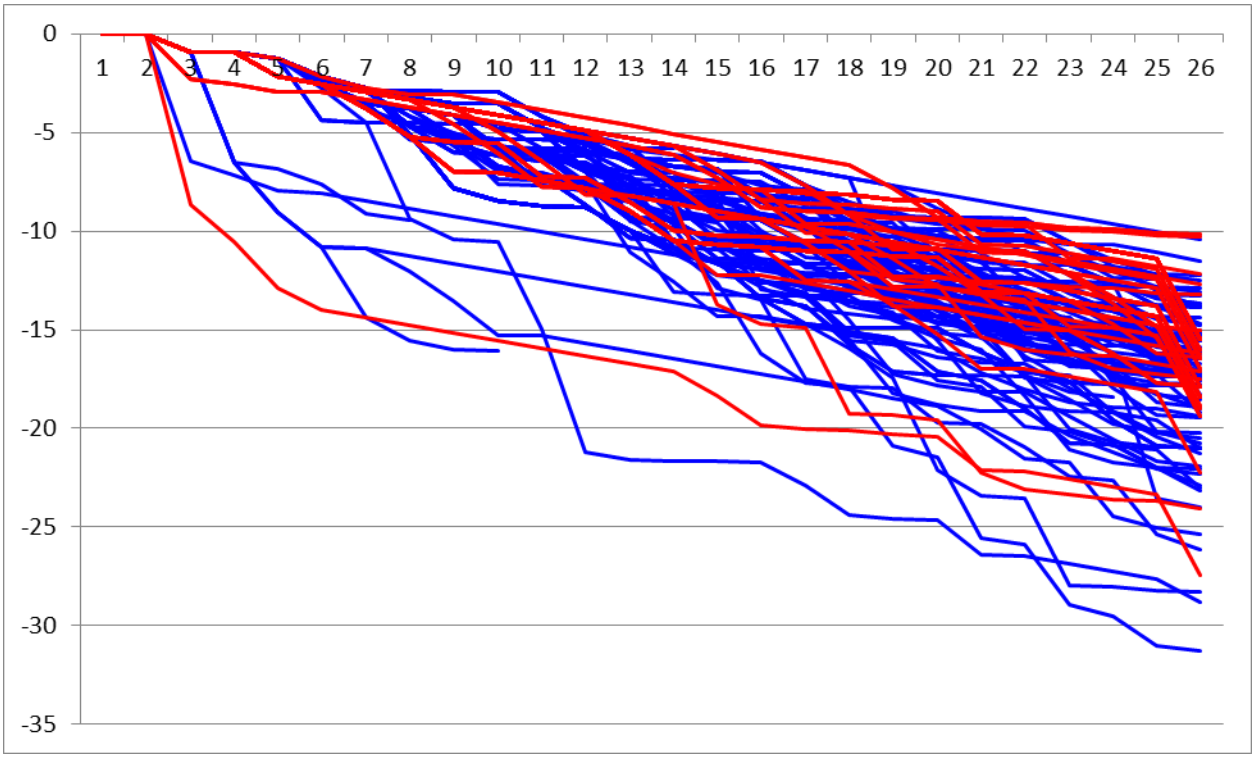

Figure E19. Order Three, One class, Unweighted

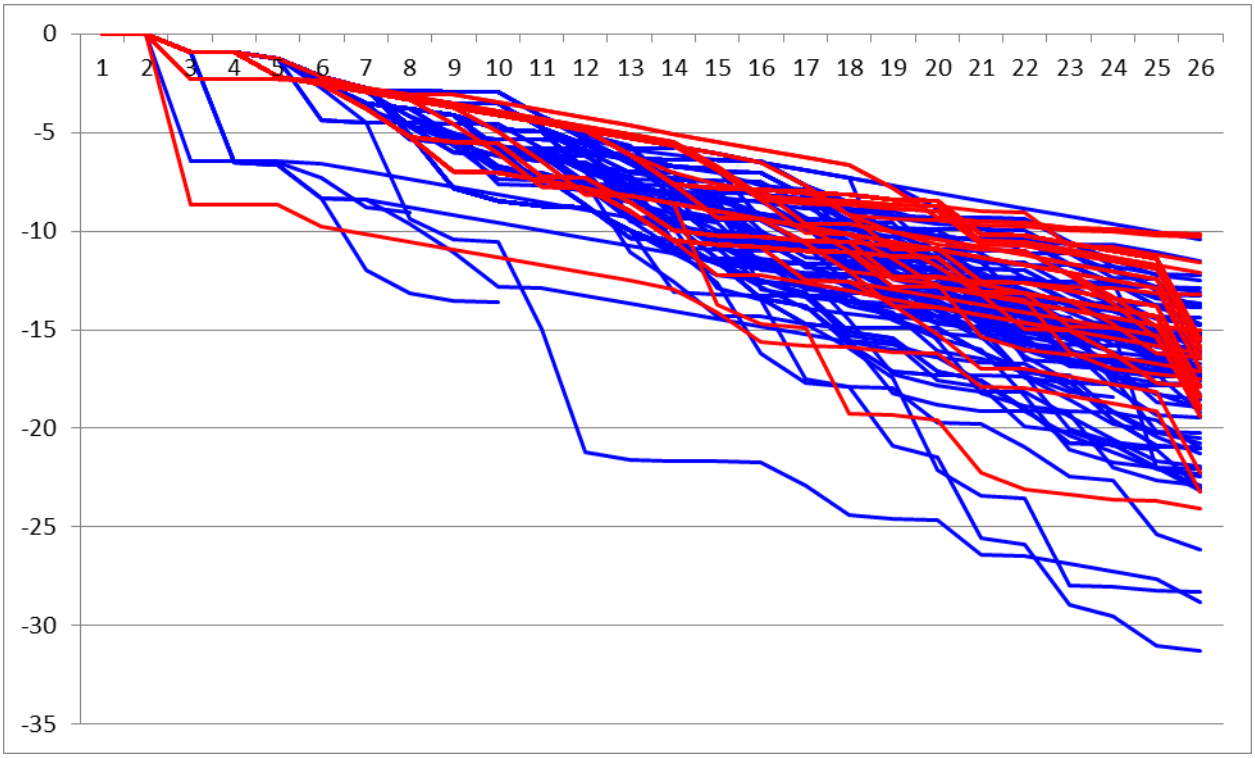

Figure E20. Order Three, One class, Weighted

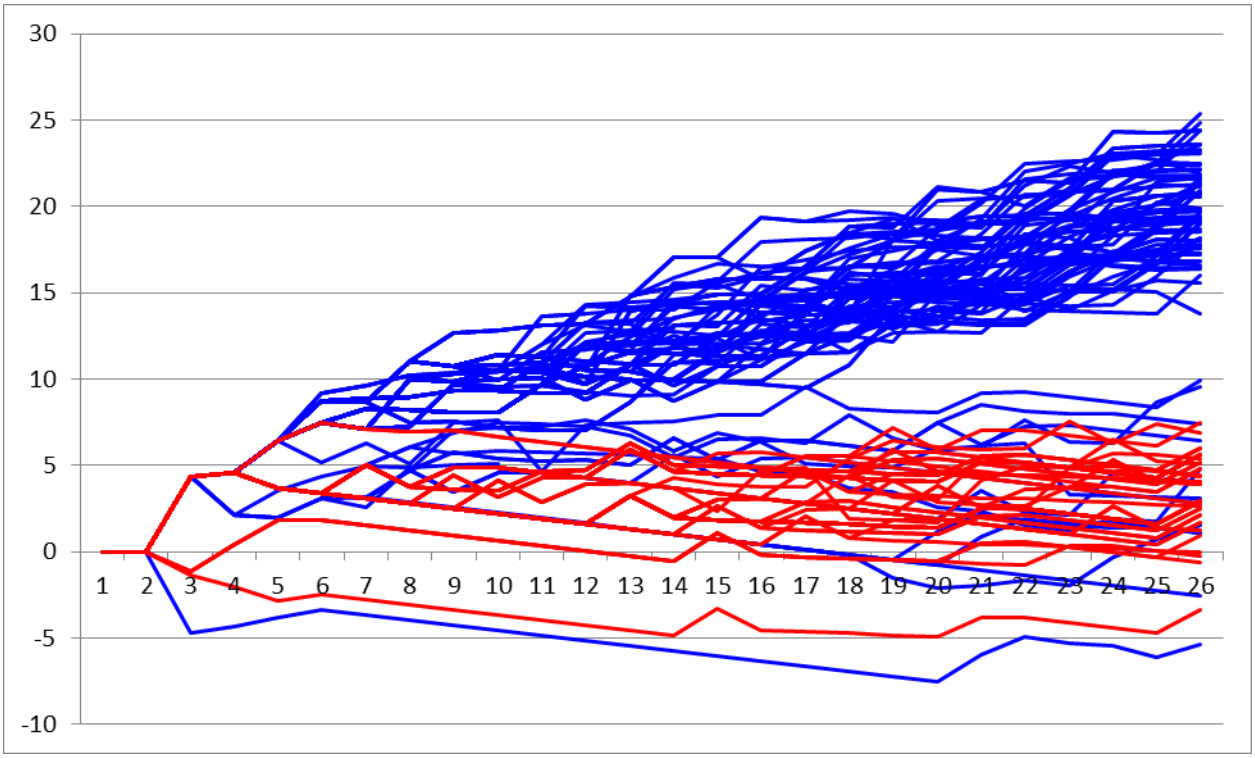

Figure E21. Order Three, Two class, Unweighted

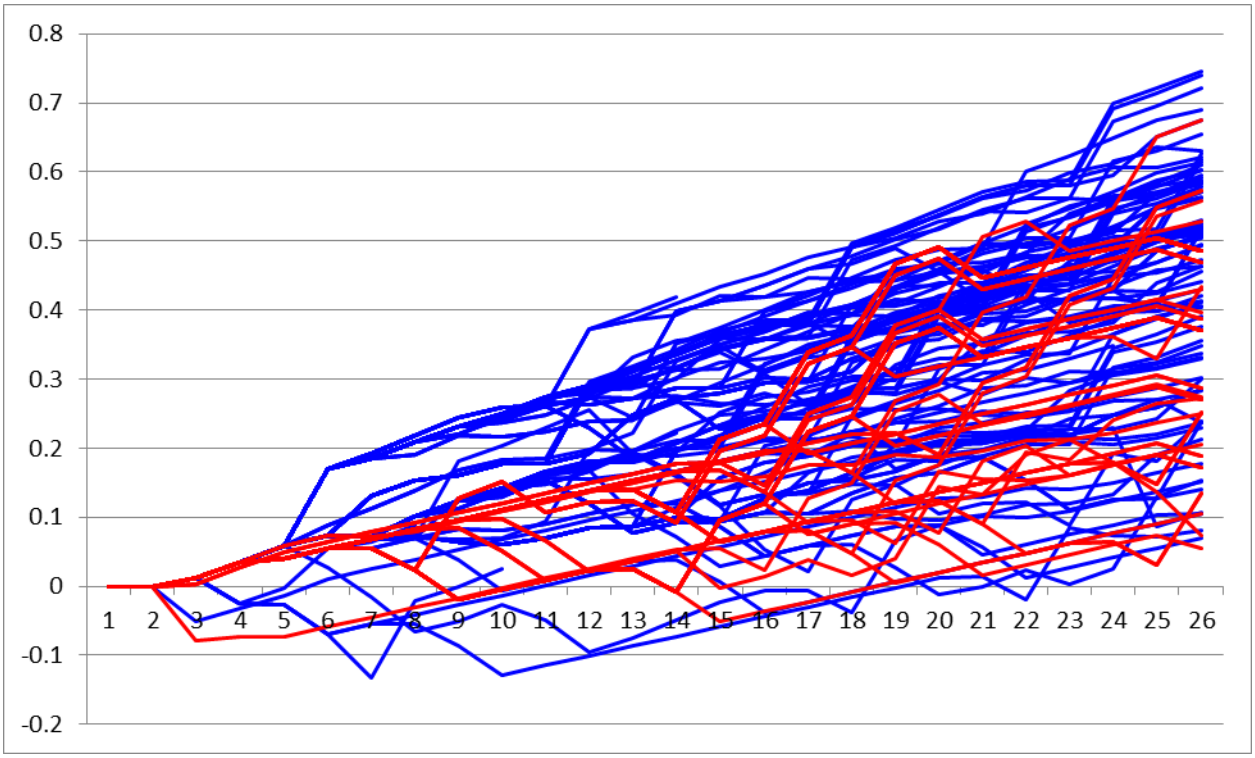

Figure E22. Order Three, Two class, Weighted

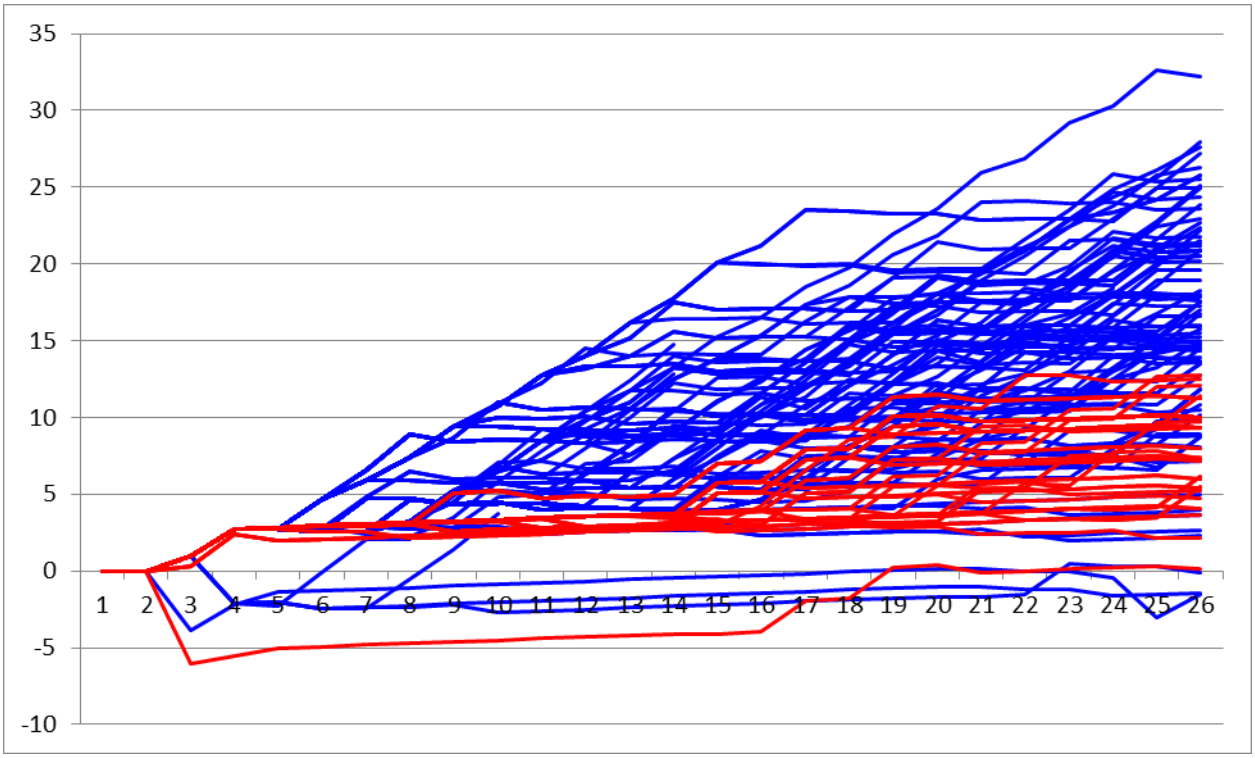

Figure E23. Order Three, Multi class, Unweighted

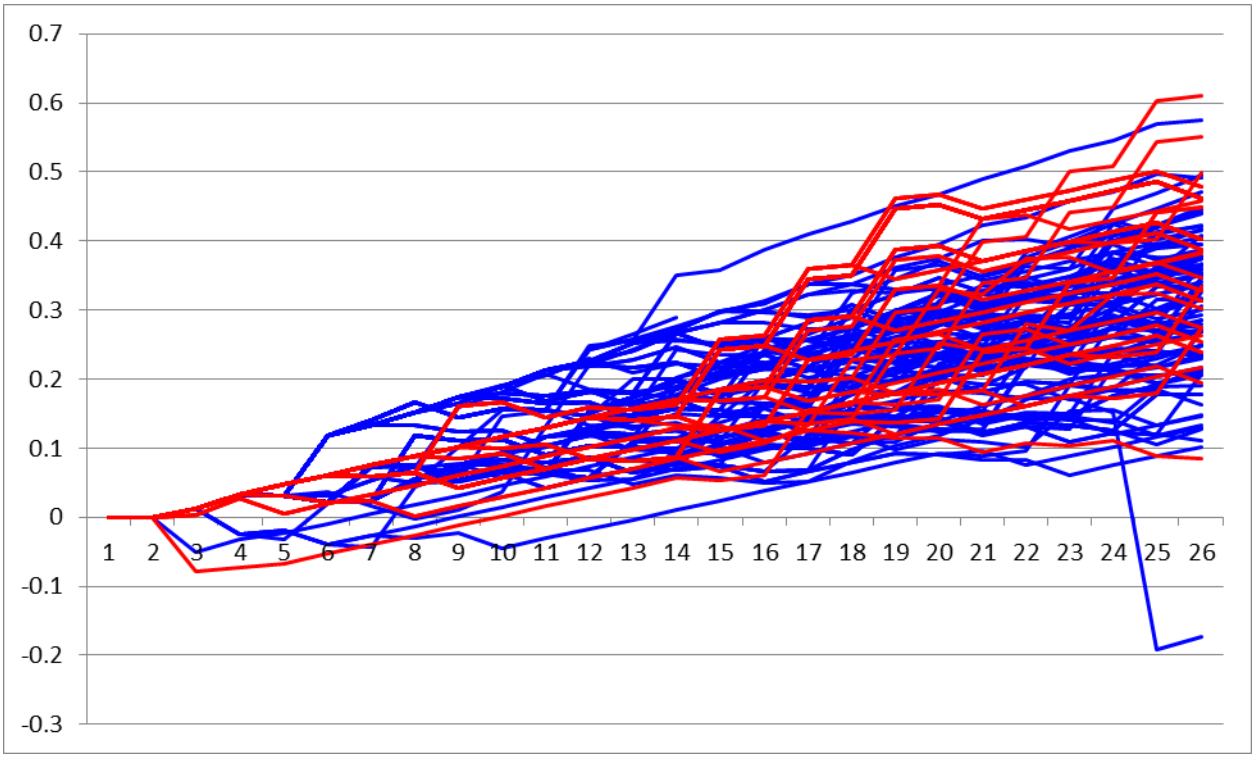

Figure E24. Order Three, Multi class, Weighted

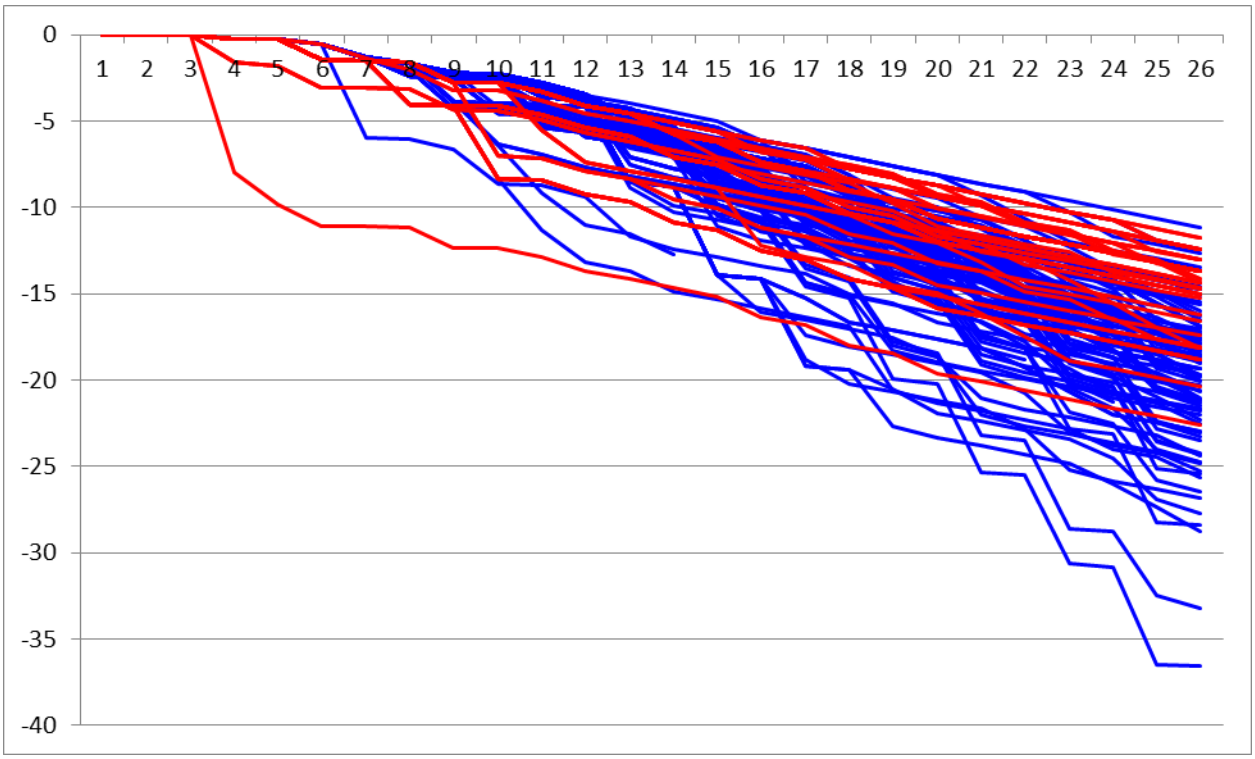

Figure E25. Order One, One class, Unweighted, Pseudo Timing

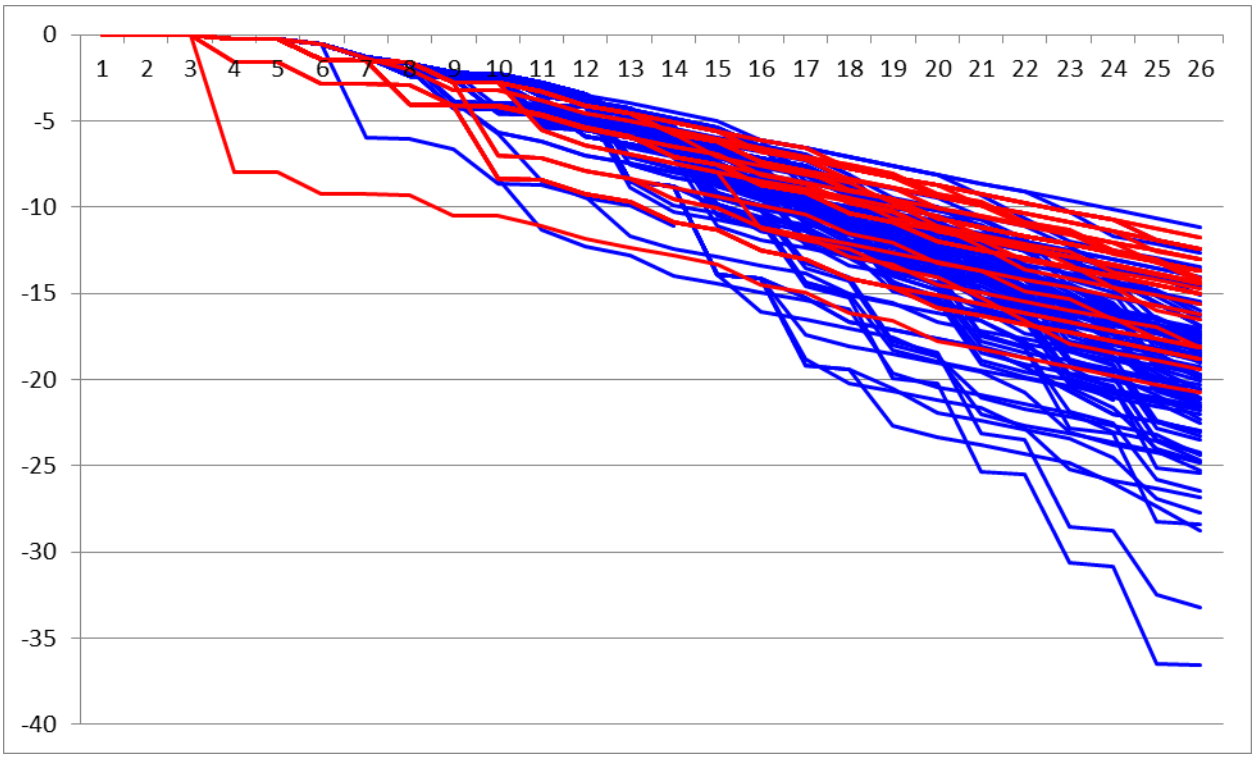

Figure E26. Order One, One class, Weighted, Pseudo Timing

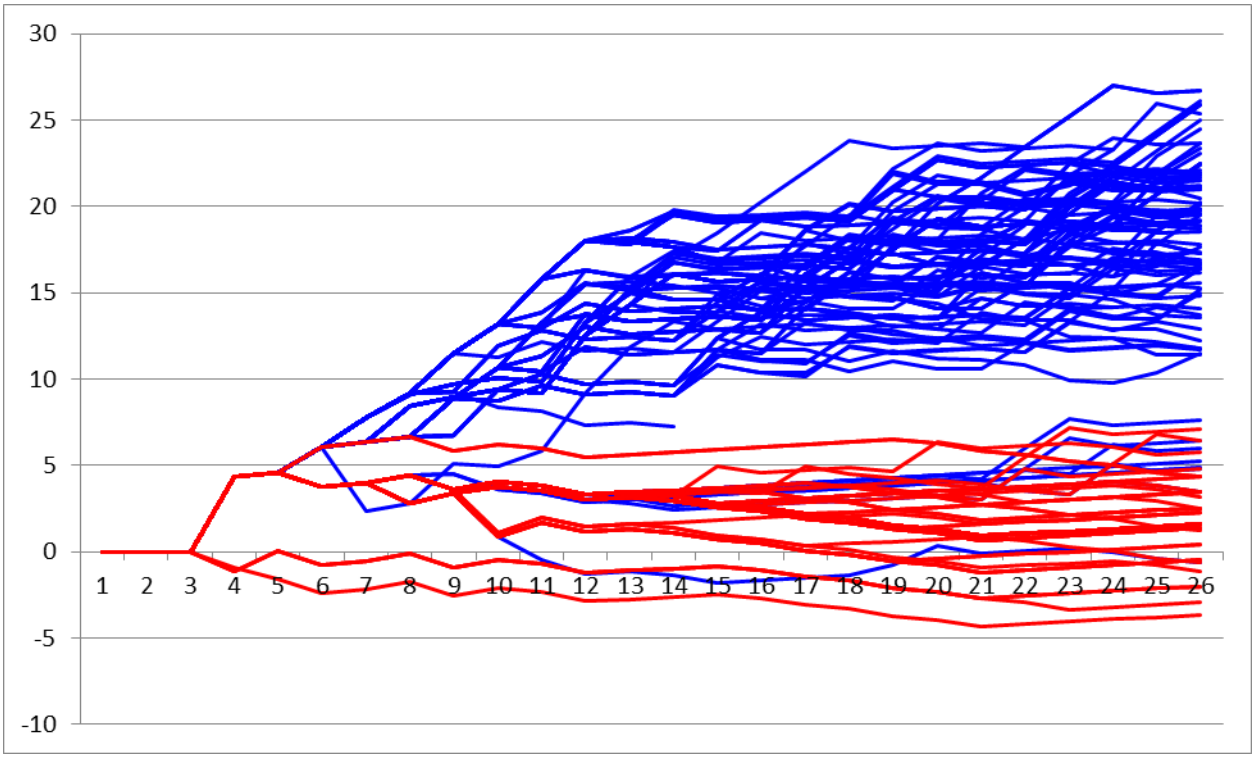

Figure E27. Order One, Two class, Unweighted, Pseudo Timing

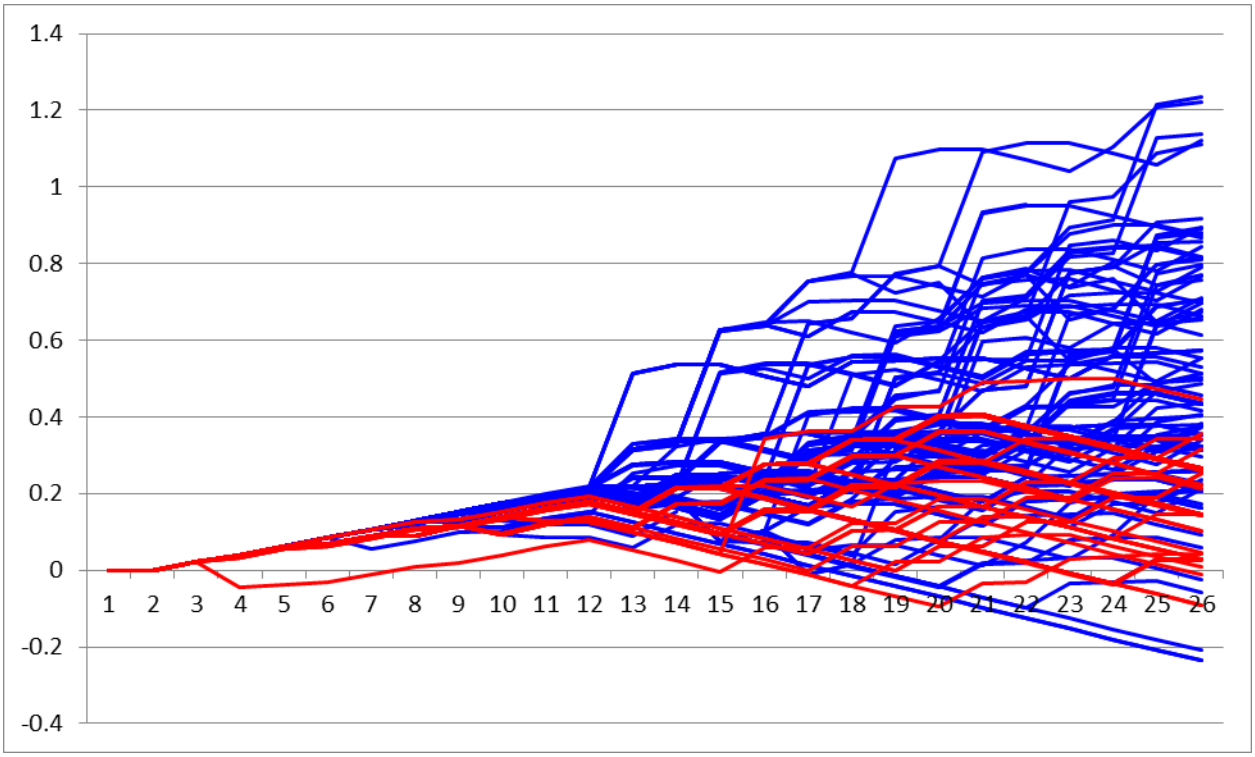

Figure E28. Order One, Two class, Weighted, Pseudo Timing

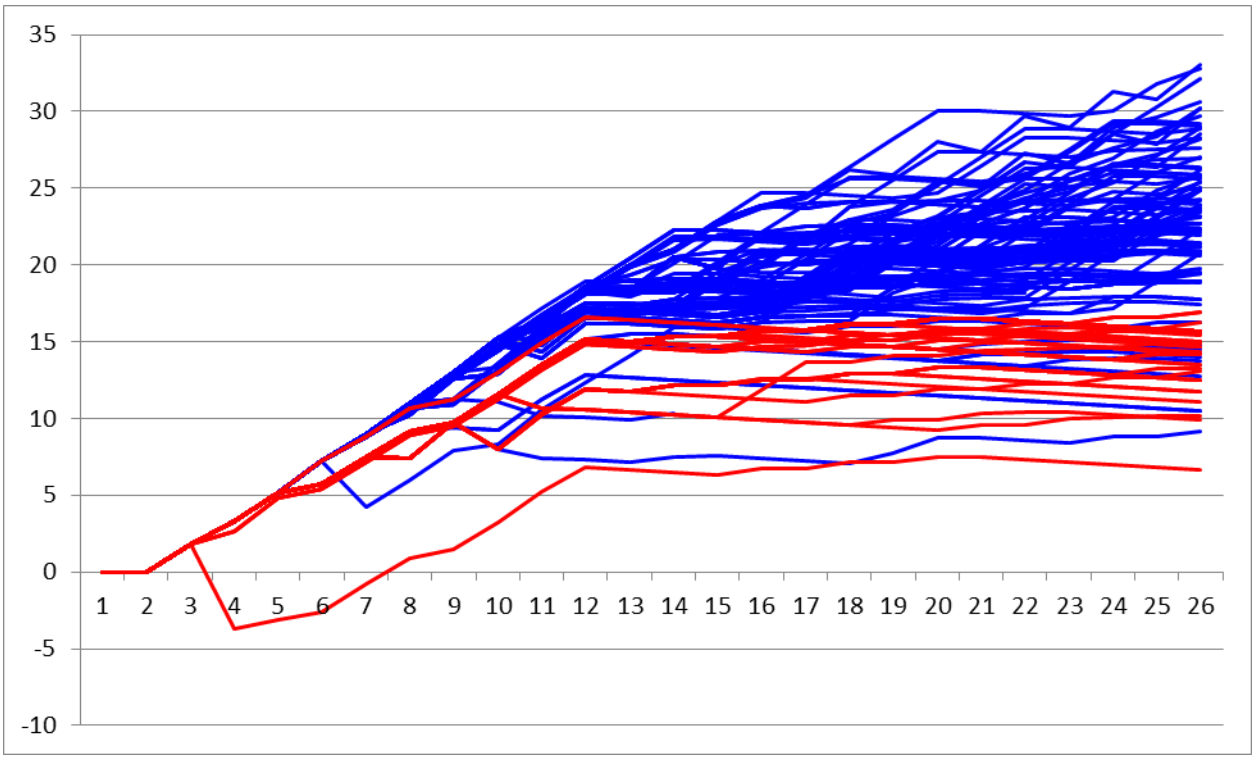

Figure E29. Order One, Multi class, Unweighted, Pseudo Timing

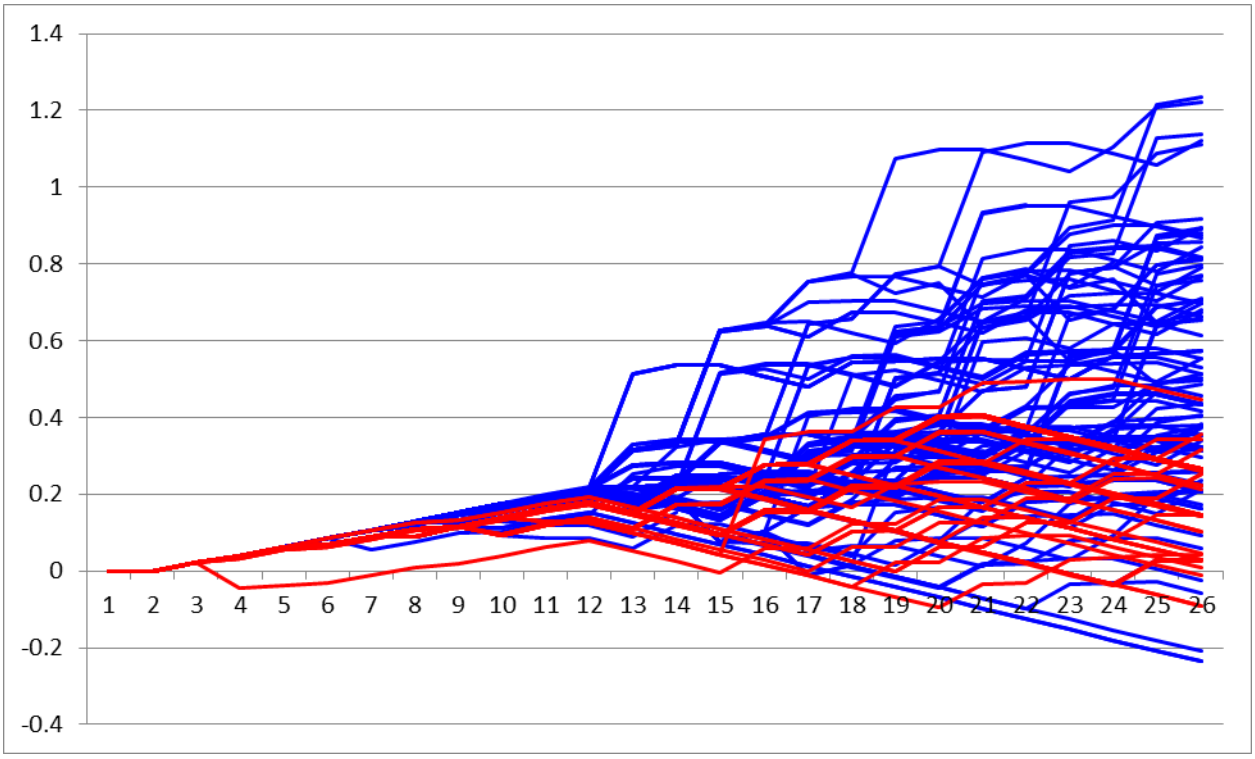

Figure E30. Order One, Multi class, Weighted, Pseudo Timing

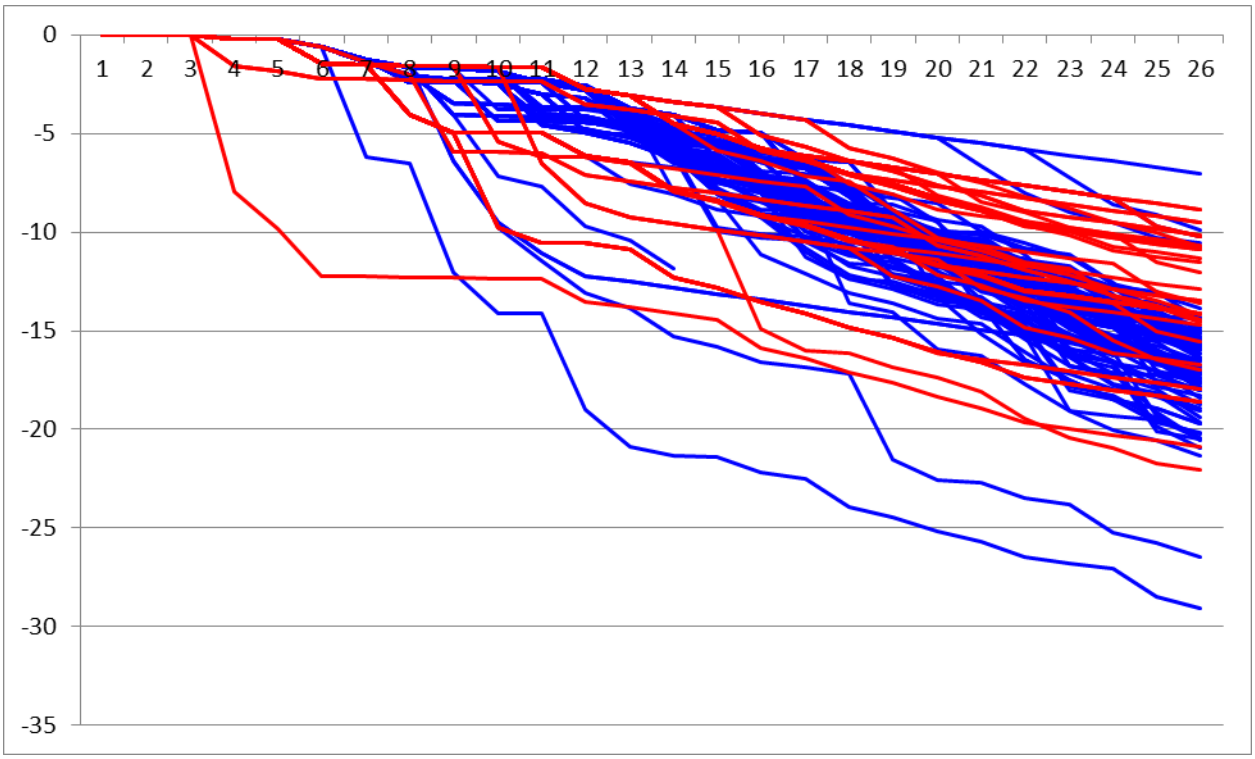

Figure E31. Order Two, One class, Unweighted, Pseudo Timing

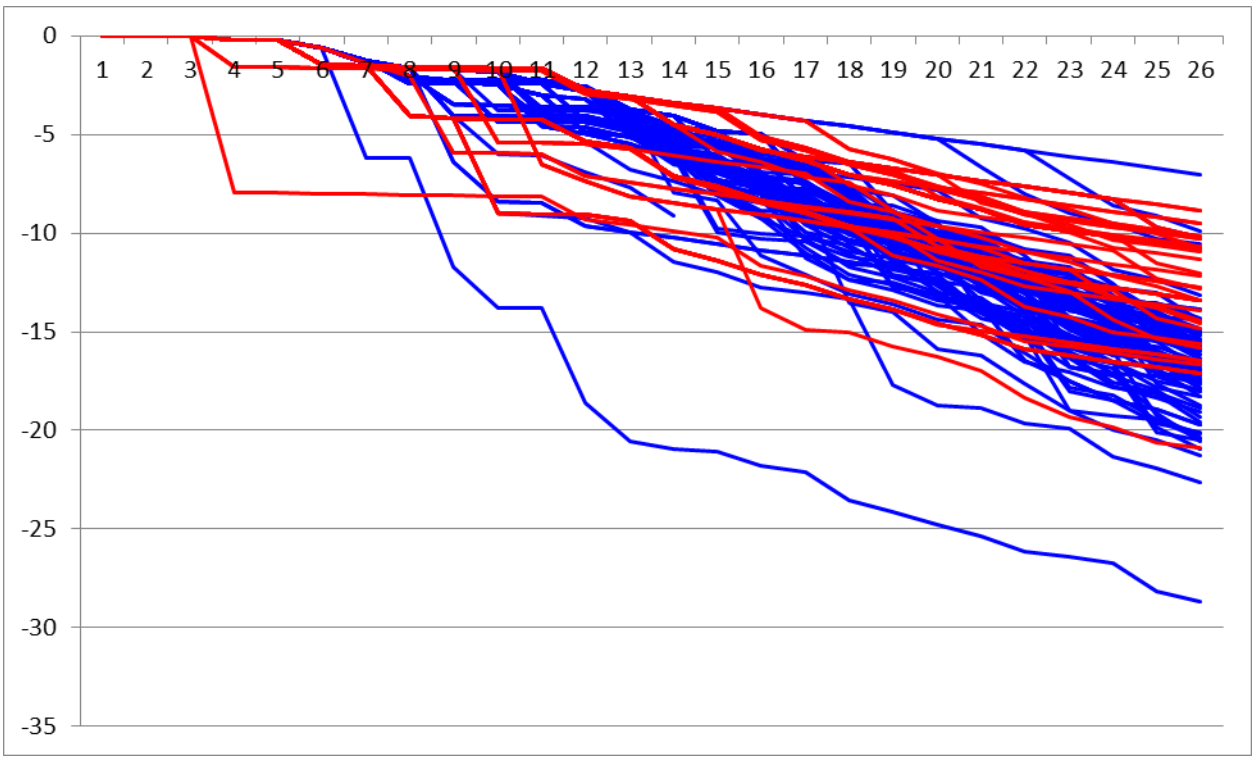

Figure E32. Order Two, One class, Weighted, Pseudo Timing

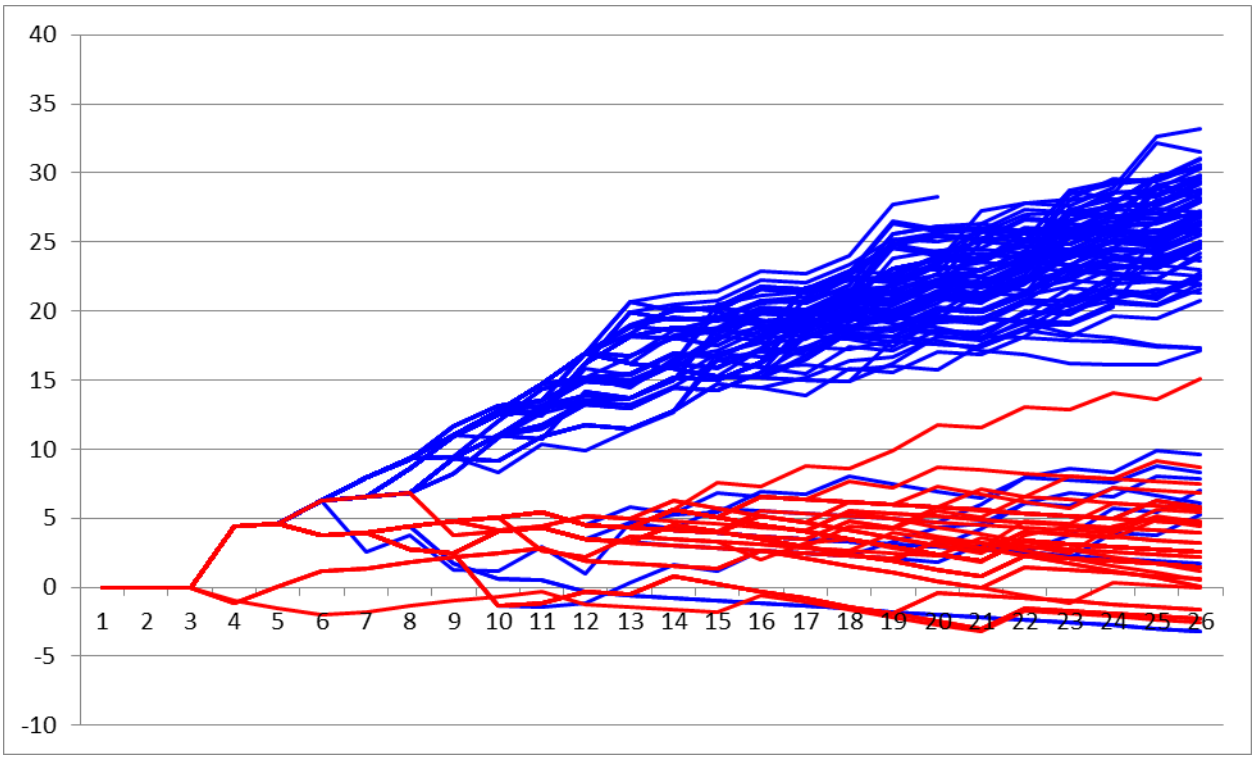

Figure E33. Order Two, Two class, Unweighted, Pseudo Timing

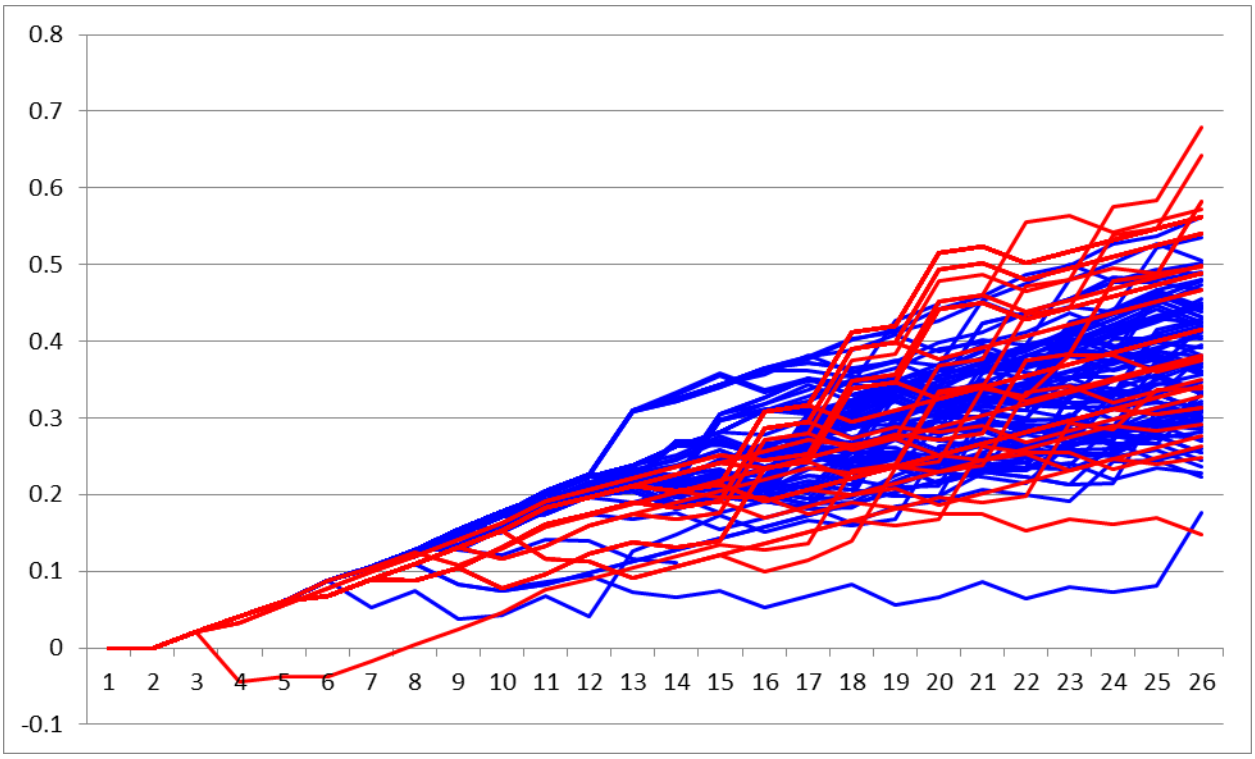

Figure E34. Order Two, Two class, Weighted, Pseudo Timing

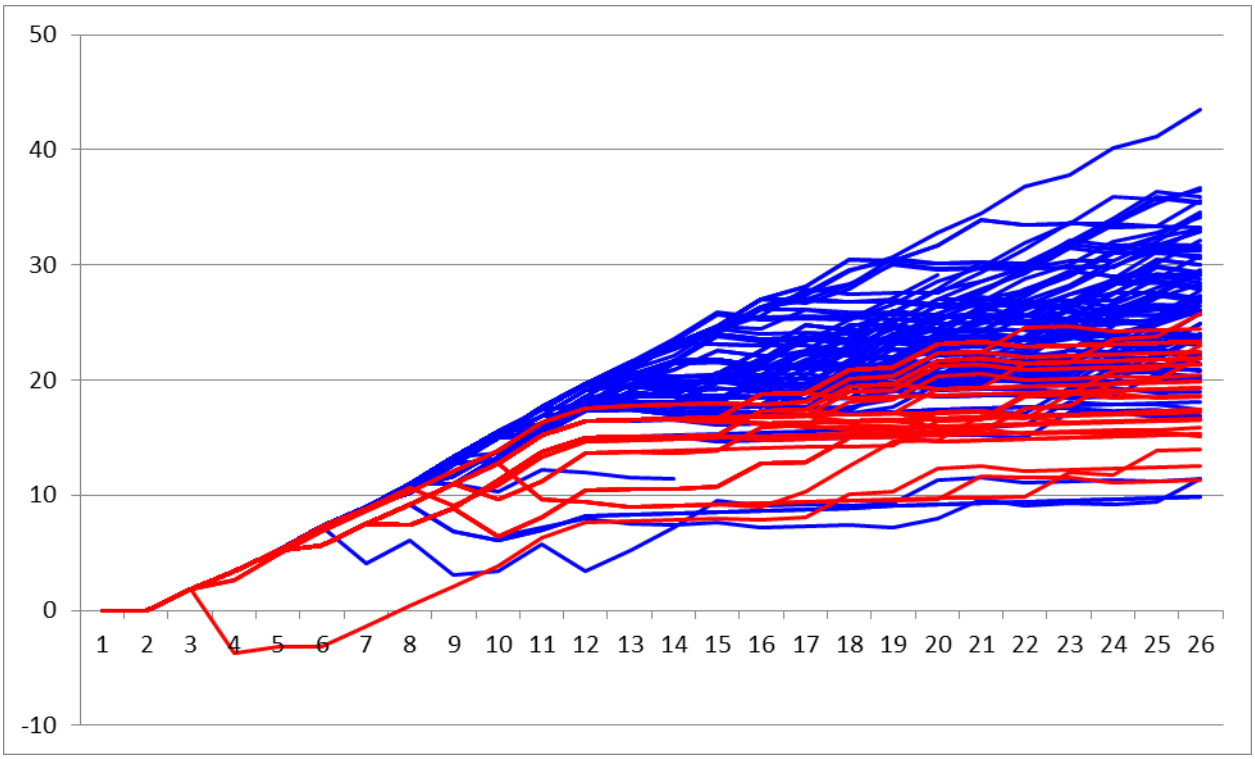

Figure E35. Order Two, Multi class, Unweighted, Pseudo Timing

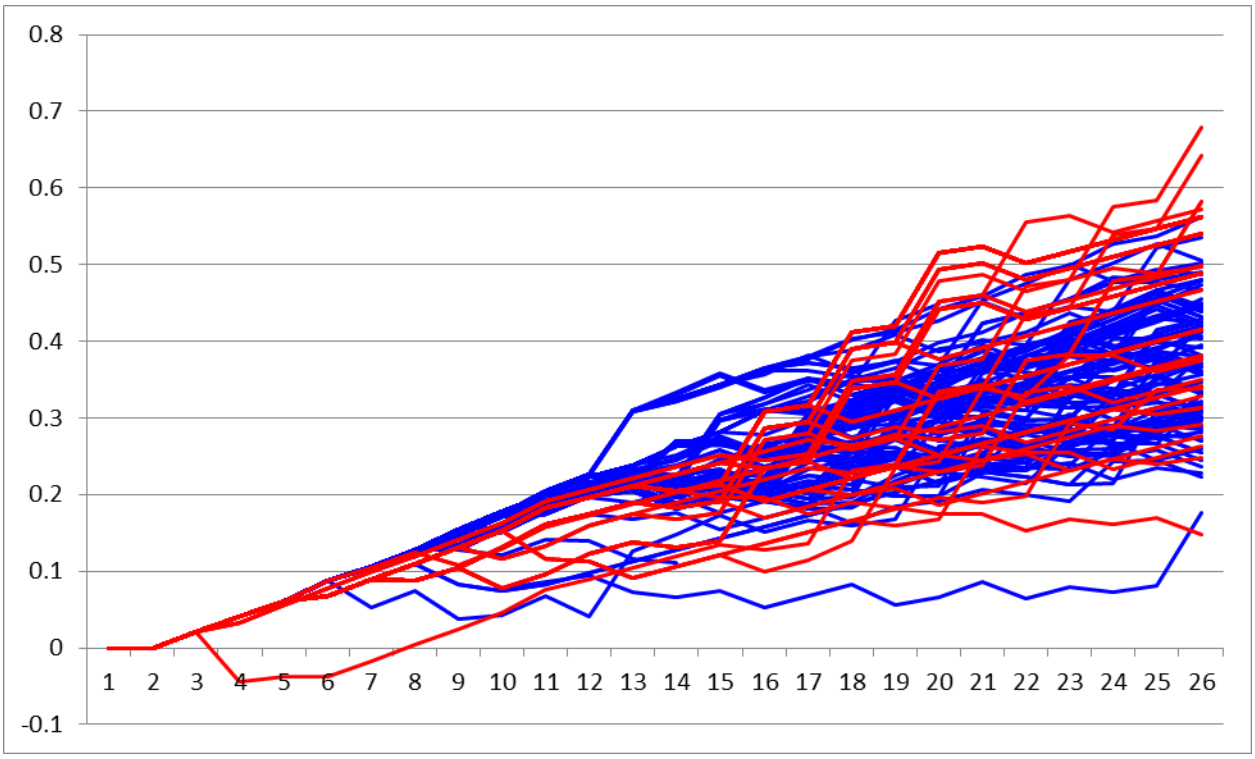

Figure E36. Order Two, Multi class, Weighted, Pseudo Timing