Clustering Methods for Multi-Resolution Simulation Modeling [*]

C.G. Cassandras, C.G. Panayiotou
Dept of Manufacturing Eng., Boston University, Boston, MA 02215

G. Diehl
Network Dynamics, Inc., 10 Speen Street, Framingham, MA 01701

W-B. Gong, Z. Liu, and C. Zou
Dept of ECE, University of Massachusetts, Amherst, MA 01003

ABSTRACT

Simulation modeling of complex systems is receiving increasing research attention over the past years. In this paper, we discuss the basic concepts involved in multi-resolution simulation modeling of complex stochastic systems. We argue that, in many cases, using the average over all available high-resolution simulation results as the input to subsequent low-resolution modules is inappropriate and may lead to erroneous final results. Instead high-resolution output data should be classified into groups that match underlying patterns or features of the system behavior before sending group averages to the low-resolution modules. We propose high-dimensional data clustering as a key interfacing component between simulation modules with different resolutions and use unsupervised learning schemes to recover the patterns for the high-resolution simulation results. We give some examples to demonstrate our proposed scheme.

Key words: Hierarchical simulation, multi-resolution simulation, clustering.

1. INTRODUCTION

In modeling complex systems it is impossible to mimic every detail through simulation. The common approach is to divide the whole system hierarchically into simpler modules, each with different simulation resolution. In this context, the output of a module becomes an input parameter to another, as illustrated in Figure 1. The decomposed modules can be high-resolution or low-resolution models. High-resolution, e.g. the usual discrete-event simulation models, take detailed account of all possible events, but are generally time consuming. Low-resolution (or coarser) modules, perform aggregate evaluation of the module's functionality (i.e., determine what would happen "on the average"). Such modules are less time consuming and can be any of the following components: differential equations (used for example in combat[1] and semiconductor simulations[2]), standard discrete-event simulation, and fluid simulation.[3] Furthermore, the decomposed modules can also be an optimization or decision support tool such as the one described by Griggs et. al.[4]

In a hierarchical setting, the lower level simulator (typically a high-resolution model) generates output data which are then taken as input for the higher level simulator (typically a low-resolution model). Hierarchical simulation is a common practice, but the design of hierarchy is always *ad hoc*. A popular practice is to use the mean values of variables from the lower level output as the input to the higher level. This implies that significant statistical information (i.e., statistical fidelity) is lost in this process, resulting in potentially completely inaccurate results. Especially when the ultimate output of the simulation process is of the form 0 or 1 (e.g., "lose" or "win" a combat), such errors can provide the exact opposite of the real output.

A systematic design and analysis framework is definitely needed. In this paper, we present some fundamental components of such a framework. Our effort has been directed at developing an *interface* between two simulation levels to preserve statistical fidelity to the maximum extent that available computing power allows. Our research focus is to use high-dimensional clustering techniques to group the high-resolution sample paths into meaningful clusters and pass on to lower resolution module(s). In the following we explain in more detail this approach.

[*]This work was supported by AFRL under Contracts F30602-99-C-0056 and F30602-99-C-0057, by AFOSR under contract F49620-98-1-0387 and by ERPI/DoD under contract WO8333-03.

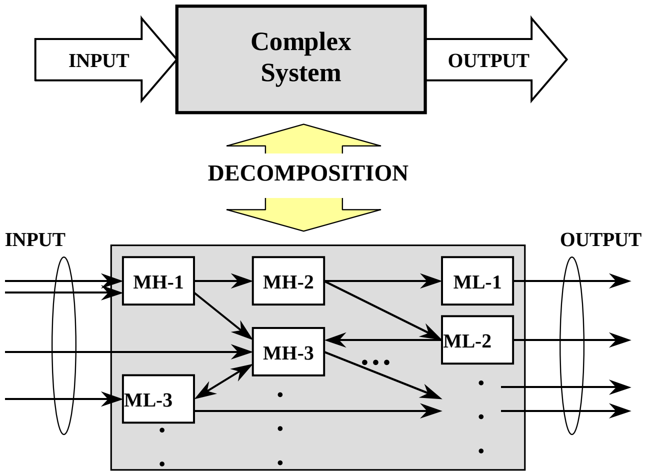

Figure 1. Decomposition of complex systems

Quite often, the system being simulated is such that the high-resolution model produces so widely divergent outputs that it does not make sense to summarize such output through a single average over the entire sample space. For example when simulating a combat, it does not make any sense to take the average of the output under different weather conditions. In such cases, we must subdivide the sample space into segments, and get the high-resolution model to produce an appropriate input to the low-resolution model for each such segment. Essentially, the low-resolution model will be broken down into a number of distinct components, one for each segment of the sample space. To carry out such a segmentation, the high-resolution paths first need to be grouped by their common features. These features then determine and feed the corresponding low-resolution model. The practice of classifying objects according to perceived similarities is the basis for much of science and engineering, since organizing data into sensible groupings is one of the most fundamental modes of understanding and learning. Clustering methods have been widely applied in pattern recognition, image processing, and artificial intelligence. In this paper, we deal with clustering methods for the preservation of statistics in hierarchical simulation.

In the following we describe the idea of using the Adaptive Resonance Theory (ART) neural network for this purpose. ART neural networks were developed by Carpenter and Grossberg[5] to understand the clustering function of the human visual system. They are based on a competitive learning scheme and are designed to deal with the stability/plasticity dilemma in clustering and general learning. It is clear that too much stability would lead to a "stubborn" mind, while too much plasticity would lead to unstable learning. ART neural networks successfully resolve this dilemma by matching the input pattern with the prototypes. If the matching is not adequate, a new prototype is created. In this way, previously learned memories are not eroded by new learning. In addition, the ART neural network implements a feedback mechanism during learning to enhance stability.

Our experiments of using ART neural networks with combat simulation paths have been quite successful.[6] We believe further improvement with the ART structure can lead to a fundamental breakthrough in large data clustering, which is needed in complex systems modeling. ART performs the clustering function based on the "angle" between the vectors that describe the various input patterns. In some cases, as it will be discussed in the sequel, this may pose a limitation of ART since the magnitude of an input pattern may contain significant information which is ignored. To alleviate this shortfall we develop a heuristic that allows the magnitude of the input pattern to play a role in the clustering function. Furthermore, we are developing a generic numerical clustering tool, based on the ART neural network, that can be used for many important problems in intelligent data analysis.

In general, the description of a typical sample path generated by a discrete-event system requires a large amount of data since such sample paths are typically quite long. This implies that the dimension of each input pattern will also be large. However for high dimensional data most of the clustering algorithms (including ART) will involve huge computational effort; thus they are not practical for simulation modeling purposes. For this reason we develop a new clustering approach where we try to take advantage of the statistical structure behind a typical sample path. For high dimensional complex systems we try to use a Hidden Markov Model (HMM), which has be successfully used in speech recognition and other areas,[7] to characterize each observed sample path. In our approach, we use an HMM to describe an arbitrary sample path and thus we cluster together all sample paths whose corresponding HMM have a high *similarity measure* (to be defined in Section 6). The advantage of this approach is that the amount of data required to describe an HMM is generally much smaller than the amount of data required to explicitly describe an observed sample path and as a result the HMM approach is more efficient.

The remaining of this paper is organized as follows. Section 2 discusses some of the issues that arise when interfacing between modules with different resolution while Section 3 presents an example where statistical fidelity is lost as a result of poor interfacing between high- and low-resolution models, i.e., passing only averages from high- to low-resolution models. Section 4 briefly presents the ART clustering algorithm and Section 5 describes an ART based tool that we have developed and its application on a complex manufacturing system. Section 6 presents a new approach for clustering sample paths based on HMMs and finally Section 7 summarizes our work.

2. INTERFACE BETWEEN HIGH- AND LOW-RESOLUTION MODELS

As mentioned above, the key issue in hierarchical simulation is the design of the interface between the hierarchies. In a typical hierarchical simulation model, the lower lever consist of a high-resolution model, such as the discrete event simulator, that generates several sample paths given some input parameters $\mathbf{u}$. The output of such simulation models is then used as input to the higher level model (typically a low-resolution model). The question that arises, the focal point of this paper, is how much and what information we need to pass from the high-resolution to the low-resolution model such that statistical fidelity is preserved.

Note that each sample path generated by the high-resolution model is also a function of some randomness ω (a random number sequence generated through some random seed). Thus, any function evaluated over an observed sample path (e.g., $h(\mathbf{u}, \omega)$) is also a random variable. Typically, we are not interested in the value of $h(\mathbf{u}, \omega)$ obtained from a single sample path but rather the expectation $E\{h(\mathbf{u}, \omega)\}$. Based on this, in hierarchical simulation it is customary to use $E\{h(\mathbf{u}, \omega)\}$ as an input parameter to the higher level model as seen in Figure 2. This is often highly unsatisfactory, since the mean often obscures important features of the high-resolution output. Said in another way, we are seeking $E\{L(h(\mathbf{u}, \omega))\}$, where $L(\cdot)$ is a function corresponding to the low-resolution model, but what we end up evaluating by passing a single average is $L(E\{h(\mathbf{u}, \omega)\})$; however, in general $E\{L(h(\mathbf{u}, \omega))\} \neq L(E\{h(\mathbf{u}, \omega)\})$.

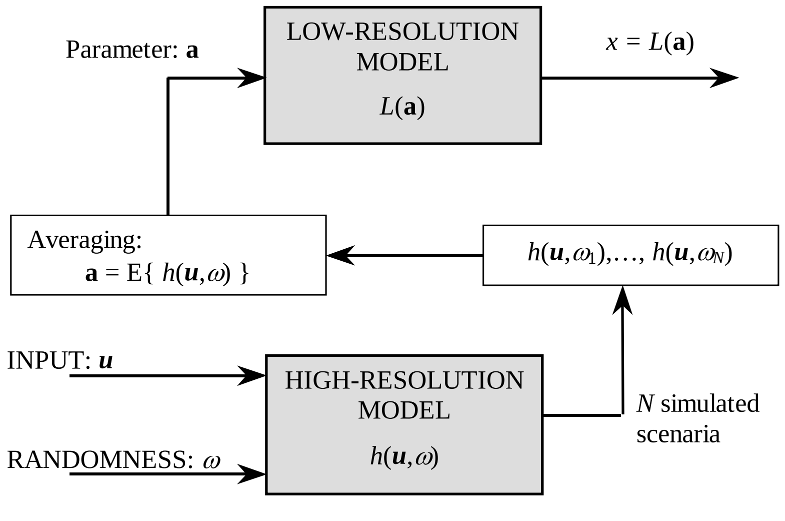

Figure 2. Hierarchical model interface: passing a simple average to the lower resolution model

To solve this problem we propose the use of clustering to identify groups of sample paths that have some "common features", and therefore, when averaged together do not cause the loss of too much information. This approach in shown in Figure 3. From the N observed sample paths we identify $m < N$ groups that share some common features and determine m input parameters $\mathbf{a}_1, \cdots, \mathbf{a}_m$ where $\mathbf{a}_i = E\{h^i(\mathbf{u}, \omega))\}$ and $h^i(\cdot)$ identifies all sample paths in cluster i. Subsequently, each parameter $\mathbf{a}_i$ is used as an input to a lower resolution model and finally we obtain $E\{L(\mathbf{a}_i)\}$ over m low-resolution components, which we claim is a better estimate of the overall system output than the one obtained using a single average.

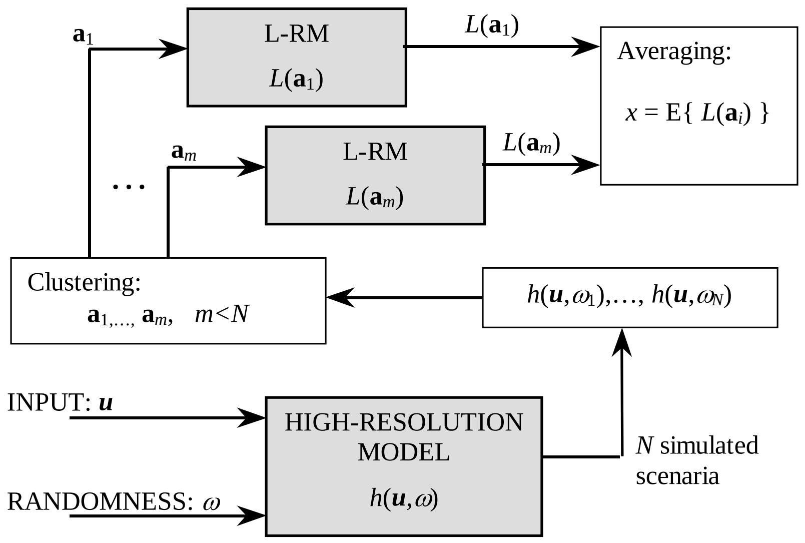

Figure 3. Hierarchical model interface: passing several averages to the lower resolution model, one for each cluster

One may pose the following question: Since the desired output is of the form $E\{L(h(\mathbf{u},\omega))\}$ why bother with clustering at all when we can evaluate $L(h(\mathbf{u},\omega))$ for *all* N obtained samples and then perform the required expectation, especially since the low-resolution models are generally easy to evaluate? The answer to this question lies in the derivation of the low-resolution model. Typically, $L(\mathbf{a})$ assumes that $\mathbf{a}$ is an expectation and therefore it would be meaningless to use some quantity obtained from a single sample path.

3. LOSING STATISTICAL FIDELITY: AN EXAMPLE

In this section we present a simple example where the loss of statistical fidelity can result in poor use of resources with possible catastrophic consequences. For this example, we assume that we are interested in planning a mission that consists of several operations $\mathbf{O}_1, \cdots, \mathbf{O}_N$ as shown in Figure 4. Due to the dependence between operations (e.g., $\mathbf{O}_4$ cannot start until $\mathbf{O}_2$ and $\mathbf{O}_3$ are completed), it is required to know the time requirements of each operation. So it is natural to ask the following question: *How much time should be allocated for each operation so that the probability of not completing an operation within the allocated time (threshold/deadline) is less than* P_0?

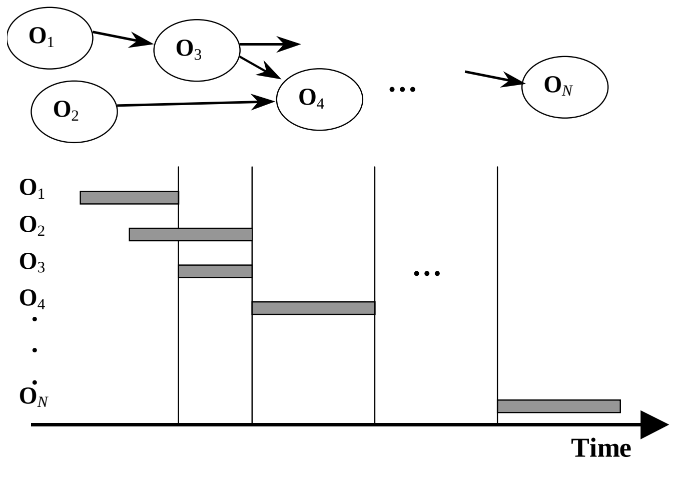

Figure 4. Operation Scheduling

The methodology for attacking this problem follows the hierarchical structure described next. For every individual operation there exists a high-resolution model that simulates the operation under different scenaria and returns the average m and standard deviation σ of the time that it takes to complete it. Subsequently, the planners use a low-resolution model to determine the time to allocate to the operation so that the probability of not meeting the deadline is less than P_0. One such low-resolution model is the Normal distribution, and the probability of not meeting the scheduled deadline is given by the corresponding cumulative distribution function. Hence, the problem is to find the threshold time T such that

$$F(T) = 1 - \int_{-\infty}^{T} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{x-m}{\sigma}\right)^2} dx \leq P_0 \tag{1}$$

A typical operation in a mission involves the Aircraft Maintenance and Refueling System (ARMS) shown in Figure 5. In this system there are Q classes $\{C_1, \cdots, C_Q\}$ of aircraft that can be refueled/maintained in one of B bases $\{B_1, \cdots, B_B\}$. Before an aircraft receives any service, it needs to travel to the corresponding base for a time τ which depends on the initial location of the aircraft with respect to the base. Once an aircraft (say aircraft a) arrives at the base, it is assigned one of the M priorities $\{P_1, \cdots, P_M\}$ and it is placed in the corresponding queue. If a token is available at the token queue, it is assigned to a which then proceeds to the next FIFO queue. Once all preceding aircraft are serviced, a enters the server where it stays for a random period of time with mean $\frac{1}{\mu_{bc}}$ which depends on the base b and the aircraft class c. For a given mission, it is required to service A aircraft before the next operation can start. Therefore, one can use a simulation model to determine the time it takes to process the A aircraft. Note that this processing time depends on several parameters: (a) The number of bases and the routing algorithm that determines what aircraft goes to what base. (b) The initial location of each aircraft and (c) The service times of each class of aircraft at each base. These parameters are considered as the *initial conditions* of each simulation (high-resolution model) and the output is going to be used as an input to the low-resolution model (i.e., the Normal distribution).

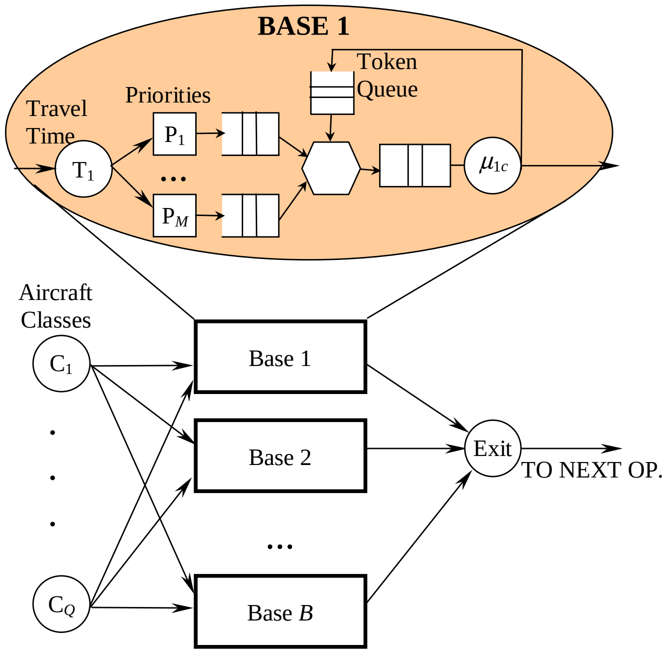

Figure 5. ARMS Model

3.1. Simulation Results

To test the clustering ideas we use the following simple problem. We assume that this operation involves $A = 100$ aircraft that can be classified in $Q = 3$ classes. An aircraft is classified as C_1 with probability 0.25, as C_2 with probability 0.25 and as C_3 with probability 0.5. Each aircraft is routed probabilistically to one of two identical bases ($B = 2$). Each base has a total of 3 tokens and each class C_1, C_2, C_3 requires service for a random period of time which is Erlang distributed with means 1.2, 1.8, 2.4 time units respectively. The routing to each base depends on the initial "state of the world" W (e.g., current weather conditions). We assume that W can only take two values 0 and 1 with probabilities 0.8 and 0.2 respectively. If $W = 0$, then the aircraft are routed to either Base 1 or Base 2 with equal probability. On the other hand, when $W = 1$, all aircraft are routed to Base 1.

We simulated the ARMS model 65,000 times and obtained the histogram shown in Figure 6 for the time it takes to process all 100 aircraft. One observation is that the average time to process all 100 aircraft is about 135 time units, however, the probability of actually processing all aircraft in 135 hours is very small. From the simulation results we also obtain an estimate of the standard deviation which was found equal to 33.9 time units.

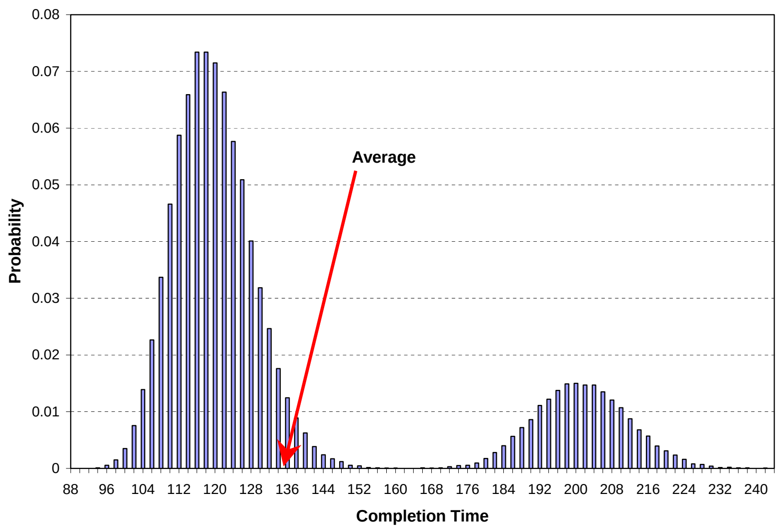

Figure 6. Completion time of the first $A = 100$ aircraft

For planning purposes now, it is required to find the smallest deadline such that the probability of missing it is less that $P_0 = 10\%$. Using the normal distribution with $m = 135$ and $\sigma = 33.9$ we find using (1) that the deadline should be set at 178.4 time units. However, using **all** simulation results it is found that in order to meet the deadline with probability equal to 90% it is necessary to set it at 200 time units. Therefore, use of the simple averages has resulted in an error of about 11%. What is more dramatic is the error in the probability of missing the deadline. If we use the deadline suggested by the previous procedure on the data obtained through simulation, it is observed that the actual probability of missing the deadline is not 10% as was originally desired but about 19.7%, therefore there is an error of about 98%. What is also interesting is that this error was obtained even though we passed both, first and second order statistics to the low-resolution model. One might expect the errors to be even larger in cases where only first order statistics are passed to the low-resolution model.

To solve this problem, we can use a simple clustering approach. Rather than using a single average and standard deviation, we can form groups of data, determine the average and standard deviation of each group and use those estimates to drive the low-resolution model. For our simulation example it is natural to cluster the obtained data based on the initial conditions of each simulation. Therefore, all data that were obtained when $W = 0$ form cluster 0 and data that were obtained when $W = 1$ form cluster 1. Using this grouping, we obtain the following results

	Cluster 0	**Cluster 1**
Samples	52,014	12,986
m	118.5	200.3
σ	8.7	10.4

Using these results, we form the weighted sum of two Normal distributions

$$\acute{f}(x) = \frac{52,014}{65,000} N(m_0, \sigma_0) + \frac{12,986}{65,000} N(m_1, \sigma_1) \tag{2}$$

Finally, using trial and error we find that in order to meet the deadline with probability 10%, it is necessary to set the deadline at 200 time units which is in agreement with the simulation results as well.

If clustering is used as an approach to preserve statistical fidelity, we need a systematic way of grouping sample paths into clusters, especially when there is no apparent way to classify sample paths like the "world state" W we used in the above example. Such approaches are presented in the next two sections.

4. CLUSTERING USING ADAPTIVE RESONANCE THEORY (ART)

Our work is motivated by our earlier path bundle grouping approach in hierarchical combat simulation. In dealing with hierarchical simulation models one needs to consider grouping sample paths generated from high-resolution models so as to provide appropriate input statistics to the lower resolution model. This requires clustering very high dimensional data vectors (the sample paths from high-resolution simulators). We have used this approach in the Concept Evaluation Model (CEM) of the Concept Analysis Agency in order to group the sample paths from the high-resolution Combat Sample Generator (COSAGE) and generate the input to the lower resolution Attrition Calculation (ATCAL). Concrete numerical results are reported in Guo et. al.[8]

The algorithm used for clustering in the ART framework is closely related to the well-known k-means clustering algorithm. Both use single prototypes to internally represent and dynamically adapt clusters. The k-means algorithm clusters a given set of input patterns into k groups. The parameter k thus specifies the coarseness of the partition. In contrast, ART uses a minimum required similarity between patterns that are grouped within one cluster. The resulting number of clusters depends on the distances (in terms of the applied metric) between all input patterns, presented to the network during training cycles. This similarity parameter is called vigilance and is denoted by ρ.

The basic ART architecture consists of the input layer F_1, the output/cluster layer F_2 and the reset mechanism which controls the degree of similarity required among patterns that are placed in the same cluster. F_1 has n input units, thus each input pattern must have dimension n, while the output layer consists of m units, therefore the maximum number of clusters that can be generated by the network is m. Note that F_2 is a competitive layer, in other words only the unit with the largest input has an activation other than zero, and therefore each unit corresponds to a different cluster. Furthermore, the input layer F_1 is subdivided into two sub-layers $F_1(a)$ the input portion and $F_1(b)$ the interface portion which receives input from both the input layer $F_1(a)$ and the output layer F_2.

In the ART algorithm, every input pattern, after some preprocessing, is compared to each of the m prototypes which are stored in the network's weights. If the degree of similarity between the current input pattern I and the best fitting prototype J is at least as high as a given vigilance ρ, prototype J is chosen to represent the cluster containing I; The vigilance ρ defines the minimum similarity between an input pattern and the prototype of the cluster it is associated with and is typically limited to the range $[0, 1]$. If the similarity between I and the best fitting prototype J does not fit into the vigilance interval $[\rho, 1]$, then a new cluster has to be installed, where the current input is most commonly used as the first prototype or "cluster center". Otherwise, if one of the previously committed clusters matches I well enough, its prototype is adapted by being slightly shifted towards the values of the input pattern I. For a detailed description and analysis of the ART algorithm refer to Carpenter and Grossberg[5] and Fausett.[9]

Often, the level of detail of the input patterns may be different; some input patters may have less than n non-zero components. In ART this has motivated the use of a *similarity measure* that is independent of the magnitude of the input pattern and so input patterns are first normalized before presented to the neural network. Rather, the similarity measure is based on the *angle* between the vector that corresponds to an input pattern and the cluster's prototype vector as illustrated in Figure 7 for a two-dimensional input vector ($n = 2$). This figure shows the jth cluster prototype vector $\mathbf{W}_j$ and the range of the cluster which extends β radians above and below the angle of $\mathbf{W}_j$. Note that β is a function of the vigilance parameter ρ; the larger the value of ρ the higher similarity is required among the vectors that are clustered in the same cluster and thus, the smaller the angle β. For this example, the angle between the input pattern I and $\mathbf{W}_j$ is $\phi > \beta$ therefore I will not be assigned to cluster j.

5. AN APPLICATION TO A "REAL-WORLD" COMPLEX SYSTEM

As mentioned earlier, we are also developing a generic clustering tool, the Intelligent Clustering Interface (ICI) and we encountered an interesting opportunity to test it in the case of a complex manufacturing system. In particular, in working with a large metal manufacturer we were faced with the issue of supplying a low-resolution model of a

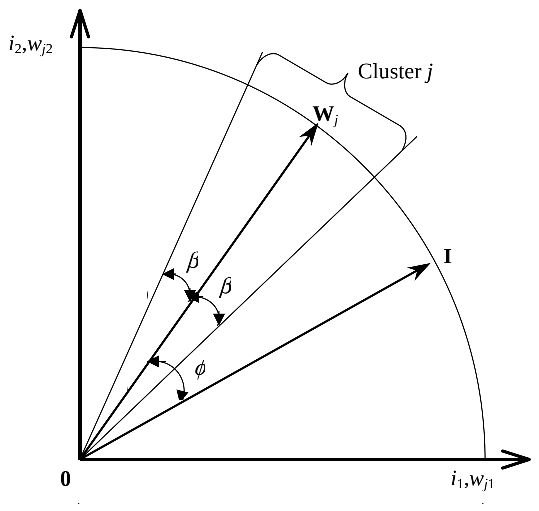

Figure 7. Similarity in ART is measured by the angle

large plant with the necessary parameters for running it. These parameters are to be obtained from detailed (high-resolution) models of the process plans (or flowpaths) for over 10,000 products manufactured in the plant. A flowpath is a specific sequence of Production Centers (PCs) with different processing characteristics at each PC (there are over 100 such PCs). Thus, each flowpath may be thought of as corresponding to a unique product; however, since the low-resolution model cannot possibly handle input data for over 10,000 flowpaths, the objective is to group products with similar flowpaths. For purposes such as forecasting, capacity planning, and lead-time estimation (among others) it is in fact indispensable to have such product groups available: not only it is conceptually infeasible to work with over 10,000 distinct products, it is also practically impossible to input such high-dimensional data for over 10,000 products and 100 PCs into modeling and decision support tools. Moreover, even if there were an automated way to accomplish this, it would be unrealistic to expect anyone to manipulate or interpret output data with information such as inventory levels and lead times for many thousands of distinct products.

In the effort to establish groups (or clusters) of products based on similarities in flowpaths and processing characteristics, an initial project was set up with plant experts given the task to "manually" create such groupings. The project was quickly abandoned: in addition to the sheer product volume which makes this task prohibitive, it is also difficult to rationally quantify "similarities" in flowpaths and processing data without some systematic means of doing so. We were able to accomplish this task using the clustering techniques we have developed and obtained a "compression" of over 10,000 products to 25-100 product clusters (depending on the aggregation accuracy required, which is completely controlled by the analyst). Of particular interest is the fact that the plant experts who reviewed the results we obtained found "by hindsight" the clusters defined by our method consistent with their expectations.

Unlike other applications of ART, in this case, the *magnitude* of the vector corresponding to an input pattern contains important information that should be used when grouping products into clusters. For example, the input vectors corresponding to two products may have the same orientation (same flowpath) but differ in their magnitude (the time spent at PC's differ by orders of magnitude). For this reason we developed an enhancement to the ART algorithm that allows us to include magnitude information in the clustering process as described next.

5.1. ART Enhancements

As already pointed out, the basic mechanism through which the ART neural network performs clustering is by grouping them using an "angle" criterion. This is illustrated in Figure 8 where we used the ICI tool to cluster 200 2-dimensional vectors. For a vigilance parameter value of $\rho = 0.99$, three clusters were obtained as seen in the figure.

It is reasonable to expect, however, that data vectors with almost identical orientation but significantly different magnitudes (prior to normalization) should be distinguishable. In order to introduce this capability into the ART setting, we have introduced the following enhancement: Each data vector is provided with an additional component (thus enlarging its dimensionality from n to $n+1$) which is the Euclidian norm of the n-dimensional vector $(x_1, \ldots, x_n)$, i.e., $(x_1^2 + \ldots + x_n^2)^{1/2}$. This forces the ART neural network to include magnitude information in its

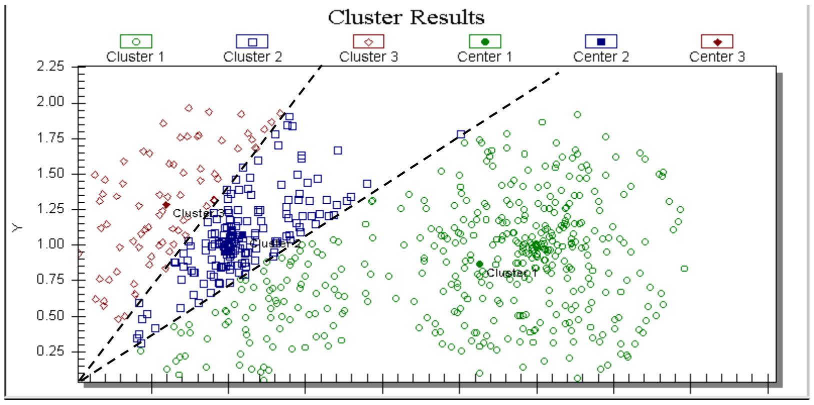

Figure 8. 200 2-dim. vectors, no extra dimension, $\rho = 0.99$

clustering algorithm. This is clearly seen in Figure 9, where the same data vectors as in Figure 8 have been clustered with this additional component. There are still three clusters, but one can see that the contents and features of the clusters have changed.

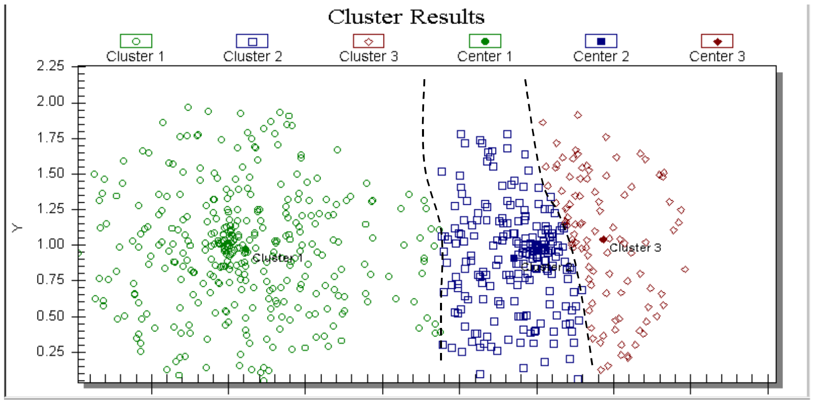

Figure 9. Same 200 2-dim. vectors, WITH extra dimension, $\rho = 0.99$

An alternative is to apply the same idea by individually reclustering each of the clusters originally obtained. Thus, within each cluster we may now distinguish between data vectors in terms of their magnitude, as illustrated in Figuer 10 where 10 clusters are now obtained by reclustering each of the three clusters in Figure 8 using a magnitude component.

Finally, note that within the ICI tool there is the capability of providing a "weight" to each component of the n-dimensional data vectors. Thus, by controlling the value of this weight for the extra component added, we can adjust the importance to be attributed to the magnitude of a vector as opposed to just its orientation in n dimensions.

6. USING HIDDEN MARKOV MODEL FOR SAMPLE PATH CLUSTERING

A sample path generated by a discrete-event system consists of a sequence $\{(e_k, t_k)\}$, $k = 1, 2, \cdots, K$, where e_k denotes the kth event and t_k its corresponding occurrence time. For typical systems, the number of observed events K is very large and thus clustering sample paths "directly" by making explicit use of the entire event sequence $\{(e_k, t_k)\}$, requires that the input vectors corresponding to such sample paths should be of a very large dimension. This impose a significant computational burden on any clustering algorithm including ART. To solve this problem we try to take advantage of the structure of discrete-event sample paths. In our experiments, we observe a sample path that is generated by an arbitrary system and try to describe it by some Markov Chain, thus we use the theory of Hidden Markov Models[7] (HMMs) to identify its parameters. Once we identify the HMM parameters we define

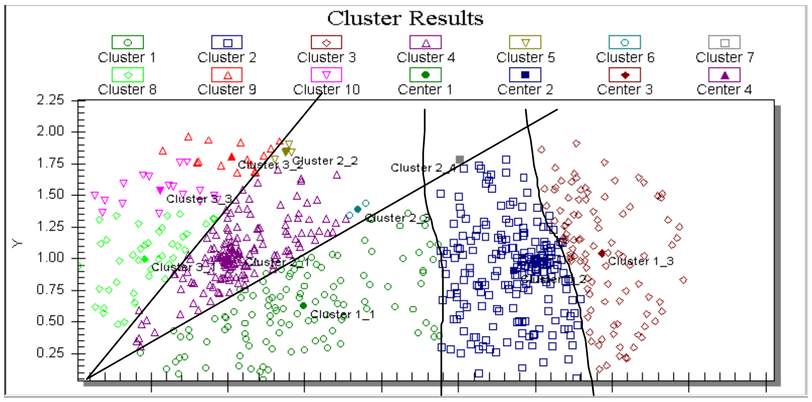

Figure 10. Same 200 2-dim. vectors, WITH extra dim. reclustered within each of the 3 original clusters, $\rho = 0.99$

a *similarity measure* among each obtained HMM and cluster together all sample paths with the largest similarity. The advantage of this approach is that the amount of information required to describe an HMM is considerably less than the amount of information required to describe a sample path. Even though the identification of the HMM parameters require some additional computational overhead, our experiments have shown that overall, the HMM approach is considerably faster than direct clustering approaches. Incidentally, we point out that this approach makes no a priori assumptions about the statistical distribution of the data to be analyzed.

Next, we demonstrate the HMM clustering approach through an example. For the purposes of our example, we assume that we have three systems S_1, S_2 and S_3. When simulated, each system generates sample paths Q_{ij}, $i = \{1, 2, 3\}$, $j = 1, 2, \cdots$ where Q_{ij} corresponds to the jth sample path generated by system S_i. When clustering sample paths, it would be reasonable to expect that sample paths generated from the same system are grouped in the same cluster. In this example, we generate 9 sample paths, 3 from each system and develop a way to distinguish between sample paths obtained from different systems. To achieve this, we first associate an HMM $\lambda_i = (A_i, B_i, \pi_i)$ to each sample path $i = 1, \cdots, 9$, where we use the notation of Rabiner.[7] A_i denotes the state transition probability, B_i denotes the observation symbol probability at every state and π_i denotes the initial state distribution. To complete the definition of the HMM, we assume the it consists of N states and that at every state we can observe any of the M possible symbols.

To construct the three systems S_i, $i = 1, 2, 3$ we assume that they consist of a Markov Chain with N_i states ($N_1 = 20$, $N_2 = 10$ and, $N_3 = 10$) and randomly generate a state transition probability matrix $\mathbf{P}_i = [p^i_{kl}]$, $k, l = 1, 2, \cdots, N_i$. Furthermore, we randomly generate the parameters μ^i_k, $k = 1, 2, \cdots, N_i$, so that the exponentially distributed sojourn time for state k has mean $1/\mu^i_k$. However, not all real systems are memoryless, therefore, to make our example more interesting we introduce some "special states" where state transitions out of such states are not made according to the state transition probability but rather through the set of rules we describe next.

For S_1, we assume that states $1, \cdots, 5$ are special states and state transitions out of these states are made according to the following rules:

State 1: System stays at state 1 for 3 consecutive, exponentially distributed sojourn time intervals, each with mean $1/\mu^1_1$. Then it jumps to state 10.

State 2: System stays at state 2 a deterministic sojourn time interval of length $2/\mu^1_2$. Then it jumps to state n according to the state visited before arriving at state 2, denoted by S_{-1}:

$$n = \begin{cases} 15 & \text{if } S_{-1} \in \{1, \cdots, 10\}, \\ 4 & \text{if } S_{-1} \in \{10, \cdots, 20\}. \end{cases}$$

State 3: System stays at state 3 for 5 consecutive sojourn time intervals, each exponentially distributed with mean $1/\mu^1_3$. Then it transfers according to state transition probability matrix $\mathbf{P}_1$. Note that after the system has spent 5 sojourn intervals at state 3, it may return to state 3 with probability p^1_{33} for 5 more intervals.

State 4: System stays at state 4 for 1 exponentially distributed sojourn time interval with mean $1/\mu_4^1$. Then it jumps to state n according to the state visited before arriving at state 4:

$$n = \begin{cases} 6 & \text{if } S_{-1} \in \{1, \cdots, 5\}, \\ 11 & \text{if } S_{-1} \in \{6, \cdots, 10\}, \\ 16 & \text{if } S_{-1} \in \{11, \cdots, 15\}, \\ 1 & \text{if } S_{-1} \in \{16, \cdots, 20\}, \end{cases}$$

State 5: System stays at state 5 for a deterministic sojourn interval of length $1/\mu_5^1$, and then transfers according to the state transition probability matrix $\mathbf{P}_1$.

Both S_2 and S_3 have only 2 special states, states 1 and 2 and the rules out of these states are the following:

State 1: System stays at state 1 for 2 sojourn time interval, each of which is exponentially distributed with mean $1/\mu_1^i$, $i = 2, 3$. Then it transfers to state n according to the previous state visited:

$$n = \begin{cases} 7 & \text{if } S_{-1} \in \{1, 2\}, \\ 9 & \text{if } S_{-1} \in \{3\}, \\ 5 & \text{if } S_{-1} \in \{4, \cdots, 10\}, \end{cases}$$

State 2: System stays at state 2 for a deterministic amount of time equal to $1/\mu_2^i$, $i = 2, 3$, and then transfers to state 5.

For each of the 9 sample paths we generate by S_1, S_2, and S_3, we adjust the HMM parameters λ_j, $j = 1, \cdots, 9$, to maximize the probability that the jth observed sample path was obtained from λ_j. This is referred to as the *training* problem and is tackled by repeatedly solving what is described as "Problem 3" in Rabiner.[7] For the purposes of this experiment, we assume that each HHM consists of $N = 6$ states. In addition, we assumed that the actual state visited by each of the systems is not observable. Rather, the observation symbols at each state are the state holding times. These can generally take any positive value so to determine B_i, the observed symbol probability, we quantize all possible values into $M = 64$ intervals.

Once we determine the HMM parameters for all sample paths, λ_i, $i = 1, \cdots, 9$, we use the *similarity measure* also defined in Rabiner[7] to determine which HMMs and consequently which sample paths are sufficiently similar so that they can be clustered together. The similarity measure is defined for any pair of HMMs λ_i and λ_j as:

$$\sigma(\lambda_i, \lambda_j) = \exp\{D(\lambda_i, \lambda_j)\} \tag{3}$$

where

$$D(\lambda_i, \lambda_j) = \frac{\log \Pr(O^j|\lambda_i) + \log \Pr(O^i|\lambda_j) - \log \Pr(O^i|\lambda_i) - \log \Pr(O^j|\lambda_j)}{2TK} \tag{4}$$

is what Rabiner[7] called the *distance measure*. $\Pr(O^i|\lambda_j)$ is the probability of the observation sequence O^i, i.e., the sequence of state holding times that correspond to the sample path Q_i, was generated by HMM λ_j. For computational convenience, we break any sample path into K smaller sample paths of length T and thus compute

$$\log \Pr\left(O^i|\lambda_j\right) = \sum_{k=1}^{K} \log \Pr\left(O_k^i|\lambda_j\right)$$

where $\Pr\left(O_k^i|\lambda_j\right)$ is the probability of the kth sequence of sample path i was generated by HMM λ_i. Also, note that the similarity measure is symmetric, that is $\sigma(\lambda_i, \lambda_j) = \sigma(\lambda_j, \lambda_i)$; a desired property for a good similarity measure.

For the similarity results shown in Table 1, the length of each of the 9 sample path is $10,000$ events. In addition, the parameters μ_j^i, $i = 1, 2, 3$, $j = 1, \cdots, N_i$ are generated such that $1/\mu_j^i$ are uniformly distributed between 4 and 50. Sample paths Q_1, Q_5 and Q_6 are generated by S_1. Q_2, Q_3, and Q_4 are generated by S_2 while Q_7, Q_8 and Q_9 are generated by S_3.

Finally, we cluster together all sample paths that correspond to HMMs with similarity measure greater than a threshold value V. Note that V corresponds to the required degree of similarity for two sample paths to be clustered

Table 1. Similarity measure among the HMMs corresponding to each of the 9 sample paths.

$\sigma(i,j)$	HMM 1	HMM 2	HMM 3	HMM 4	HMM 5	HMM 6	HMM 7	HMM 8	HMM 9
HMM 1	1	0.760	0.769	0.776	**0.950**	**0.950**	0.798	0.804	0.794
HMM 2	0.760	1	**0.940**	**0.949**	0.772	0.776	0.837	0.839	0.835
HMM 3	0.769	**0.940**	1	**0.947**	0.777	0.785	0.850	0.847	0.847
HMM 4	0.776	**0.949**	**0.947**	1	0.787	0.793	0.847	0.844	0.844
HMM 5	**0.950**	0.772	0.777	0.787	1	**0.951**	0.799	0.804	0.799
HMM 6	**0.950**	0.776	0.785	0.793	**0.951**	1	0.815	0.820	0.809
HMM 7	0.798	0.837	0.850	0.847	0.799	0.815	1	**0.945**	**0.943**
HMM 8	0.804	0.839	0.847	0.844	0.804	0.820	**0.945**	1	**0.950**
HMM 9	0.794	0.835	0.847	0.844	0.799	0.809	**0.943**	**0.950**	1

together, like the vigilance parameter ρ of ART. For example, if $V = 0.9$, then the similarity measures exceeding V are:

Cluster 1: $\sigma(1,5)$, $\sigma(1,6)$, $\sigma(5,6)$
Cluster 2: $\sigma(2,3)$, $\sigma(2,4)$, $\sigma(3,4)$
Cluster 3: $\sigma(7,8)$, $\sigma(7,9)$, $\sigma(8,9)$

therefore, the HMM method has successfully classified all sample paths.

7. CONCLUSION

In this paper, we discuss the basic concepts involved in multi-resolution simulation modeling of complex stochastic systems and demonstrate that, using the average over all available high-resolution simulation results as the input to subsequent low-resolution modules is inappropriate and can lead to erroneous final results. Instead high-resolution output data should be clustered into groups that match underlying features of the system behavior before feeding group averages to low-resolution modules. In addition, we propose two approaches for performing high-dimensional data clustering based on neural networks and Hidden Markov Models.

REFERENCES

1. U.S. Army Concept Analysis Agency, *Concept Evaluation Model*, 1983.
2. S. Lee and T. Tang, "Transport coefficients for a silicon hydrodynamic model extracted from inhomogeneous Monte Carlo calculations," *Solid-State Electronics* **35**(4), pp. 561–569, 1992.
3. A. Yan and W. Gong, "Fluid simulation of high speed networks," *To appear: IEEE Transactions on Information Theory* , Jun. 1999.
4. B. Griggs, G. Parnell, and L. Lehmkuhl, "An air mission planning algorithm using decision analysis and mixed integer programming," *Operations Research* **45**(5), 1997.
5. G. Carpenter and S. Grossberg, "ART 2: Self-organization of stable category recognition codes for analog input patterns," *Applied Optics* , Dec 1987.
6. C. Cassandras, W.-B. Gong, C. Liu, C. Panayiotou, and D. Pepyne, "Simulation-driven metamodeling of complex systems using neural networks," in *Proceedings of 19th SPIE Conference*, Apr. 1998.
7. L. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of IEEE* **77**(2), pp. 267–293, 1989.
8. Y. Guo, X. Yin, and W. Gong, "ART2 neural network clustering for hierarchical simulation," in *Proceedings of 19th SPIE Conference*, (Orlando), Apr. 1998.
9. L. Fausett, *Fundamentals of Neural Networks: Architecture, Algorithms and Applications*, Prentice Hall, 1994.