New Generation of Data Mining Applications, Edited by Jozef Zurada and Medo Kantardzic.
ISBN 0-471-XXXXX-X Copyright © 2003 Wiley, Inc.

Chapter ??

Statistical Modeling of Large-Scale Scientific Simulation Data

Tina Eliassi-Rad, Chuck Baldwin, Ghaleb Abdulla, Terence Critchlow

Center for Applied Scientific Computing
Lawrence Livermore National Laboratory
{eliassi, baldwin5, abdulla1, critchlow}@llnl.gov

Abstract

With the advent of massively parallel computer systems, scientists are now able to simulate complex phenomena (*e.g.*, explosions of a stars). Such scientific simulations typically generate large-scale data sets over the spatio-temporal space. Unfortunately, the sheer sizes of the generated data sets make efficient exploration of them impossible. Constructing *queriable statistical models* is an essential step in helping scientists glean new insight from their computer simulations. We define queriable statistical models to be descriptive statistics that (*i*) summarize and describe the data within a user-defined modeling error, and (*ii*) are able to answer complex range-based queries over the spatio-temporal dimensions.

In this chapter, we describe systems that build queriable statistical models for large-scale scientific simulation data sets. In particular, we present our *Ad-hoc Queries for Simulation* (*AQSim*) infrastructure, which reduces the data storage requirements and query access times by (*i*) creating and storing queriable statistical models of the data at multiple resolutions, and (*ii*) evaluating queries on

these models of the data instead of the entire data set. Within AQSim, we focus on three simple but effective statistical modeling techniques. AQSim's first modeling technique (called *univariate mean modeler*) computes the "true" (unbiased) mean of systematic partitions of the data. AQSim's second statistical modeling technique (called *univariate goodness-of-fit modeler*) uses the Andersen-Darling goodness-of-fit method on systematic partitions of the data. Finally, AQSim's third statistical modeling technique (called *multivariate clusterer*) utilizes the *cosine similarity measure* to cluster the data into similar groups. Our experimental evaluations on several scientific simulation data sets illustrate the value of using these statistical models on large-scale simulation data sets.

Keywords: "queriable" statistical models, statistical modeling, multivariate clustering, large-scale scientific simulation data sets, approximate ad-hoc queries

1 Introduction

By utilizing massively parallel computer systems, scientists are able to simulate complex phenomena such as explosions of stars. Specifically, scientists encode complex equations into computer programs (known as, *simulation codes*). These computer programs produce tera-scale data sets, which contain millions of distinct spatial elements, thousands of discrete time steps, and hundreds of physical variables (such as temperature and pressure) at each point in the spatial-temporal dimensions.[1]

To gain new insight into the simulated scientific phenomena, scientists require computer applications that can rapidly analyze and display such massive data sets. However, the shear sizes of these data sets make even the best analysis and visualization tools inadequate (see Figure 1). Consequently, the need to efficiently and effectively explore such huge data sets has led to a surge of interest in scalable analysis and visualization techniques (Abdulla, et al. 2001; Acharya, et al. 1999; Agrawal, et al. 1998; Chakrabarti, et al. 2000; Eliassi-Rad, et al. 2002; Freitag and Loy, 1999; Wang, et

[1] The spatial-temporal dimensions are defined to be *x*, *y*, *z*, and *time*.

al. 1997). This chapter examines the statistical approaches that are utilized for rapid exploration of large-scale scientific simulation data.

Figure 1. Currently available analysis and visualization tools cannot efficiently process tera-scale scientific data.

The most commonly utilized techniques in building statistical models for huge data sets are sampling, histograms, and/or assumptions about the *a priori* distribution on the data. Unfortunately, such techniques are not desirable for scientific simulation data sets since they tend to be computationally expensive for high-dimensional data and can miss outliers.[2] Despite these drawbacks, systems that do not rely on such techniques are rare. In particular, our *Ad-hoc Queries for Simulation* (*AQSim*) infrastructure constructs *queriable statistical models* without the use of the aforementioned techniques. We define queriable statistical models to be descriptive statistics that (*i*) summarize and describe the data within a user-defined modeling error, and (*ii*) are able to answer complex range-based queries over the spatio-temporal dimensions.[3]

AQSim has two processors: (*i*) *model generator* and (*ii*) *query processor*. Figure 2 illustrates an overview of AQSim's architecture. AQSim's two processors enable reduction of both the data storage requirements and the query access times. The model generator builds and stores statistical queriable models of the data at multiple resolutions. Since models take less storage space than the original data set,[4] AQSim is able to reduce the data storage requirements. AQSim's query processor takes as input the generated models, a user's query, and the amount of time that the user is willing to wait for an answer. Then, while its running time is less than the user-defined time limit, the query processor searches the generated models and returns those models that satisfy the given query. AQSim's query

[2] Outliers are very important in scientific simulation data since the insight gained from them is commonly higher than from trends.

[3] Queriable models are very useful to the scientists since the models can answer specific user's queries.

[4] The original data set typically resides on tertiary storage.

processor decreases the query response time since models of the data are queried (instead of the entire data set). This chapter focuses on AQSim's model generator.[5] In particular, Section 3 describes three simple but effective statistical modeling techniques: (*i*) *mean modeler*, (*ii*) *goodness-of-fit modeler*, and (*iii*) *multivariate clusterer*.

Figure 2. AQSim's architecture

AQSim's mean modeler is univariate and captures the true mean of systematic spatial-temporal partitions of the data. It has two major advantages. First, it makes no assumptions about the distribution of the data. Second, it calculates its model parameters through one sweep of the data. The error on the mean modeler is a variant of the *root mean square error* (RMSE).

AQSim's goodness-of-fit modeler is also univariate and captures the normality of systematic spatial-temporal partitions of the data by utilizing the *Anderson-Darling* goodness-of-fit test (D'Agostino and Stephens, 1986). Similar to the mean modeler, the goodness-of-fit modeler is able to calculate its model parameters through one sweep of the data. However, this modeler assumes that since the data describes a physical phenomenon, it probably fits a normal distribution. The error on this model is the *Type I error* associated with the goodness-of-fit test.

AQSim's multivariate clusterer captures the interrelationships between a data set's *physical variables* by finding "similar" behavior among them. Physical variables are all variables except the spatial (*i.e.*, *x*, *y*, *z*) and temporal (*i.e.*, *time*) variables. For example, *temperature*, *pressure*, and *density* are considered physical variables. The exclusion of the spatial dimensions from the clustering process is important since "similar" characteristics could be far from each other in the spatial region. For example, the temperature of the outer shell of a star is more homogeneous than its inner shell even though the outer shell spans a spatial area several times larger than that of the inner shell.

[5] For more information on AQSim's query processor, see Baldwin, et al. (2003a), Eliassi-Rad, et al. (2002), and Lee, et al. (2003).

AQSim's multivariate clusterer defines "similar" behavior by utilizing a variant of the *cosine similarity measure*, which has been used in information retrieval applications (Van Rijsbergen, 1979). We chose the cosine similarity measure instead of other metrics (such as the Euclidean distance metric) since our clusters will be used for query retrieval.

To scale AQSim's multivariate clusterer for large-scale data sets, we take advantage of the geometrical properties of the cosine similarity measure. Specifically, we utilize the user-defined clustering threshold to place a tighter upper-bound on the similarity of items within a cluster. This allows us to reduce the clustering time from $O(n^2)$ to $O(n \times g(f(u)))$, where n is the number of data points, $f(u)$ is a function of the user-defined clustering threshold, and $g(f(u))$ is the number of data points satisfying the new threshold $f(u)$. Eliassi-Rad and Critchlow (2003) empirically showed that on average $g(f(u))$ is much less than n.

Even though spatial variables do not play a role in building AQSim's clusters, it is desirable to associate each cluster with its correct spatial region. In particular, it is important for AQSim's clustering model to frame the answers to scientists' queries in the spatial region of the original data. While this information is trivially contained in AQSim's univariate models, an additional step must be performed to map clusters to their appropriate spatial representations. To accomplish this, we use a *linking* algorithm for connecting each cluster to the appropriate nodes of the data set's *topology tree*. Such a topology tree captures and stores the spatial connectivity of the data set by utilizing the intrinsic topology given in the original scientific problem (Baldwin, et al. 2003b). AQSim's linking technique is embedded into its clustering algorithm. That is, connections between a cluster and the correct nodes of the topology tree are made as the cluster is being constructed. In this way, AQSim avoids traversing the clusters after they are made, which in turn reduces its execution time. The main challenge for AQSim's

linking algorithm is to find the best *m* nodes in the topology tree for a particular cluster, *c*, where *m* is much less then the number of items in *c*.

This chapter is organized as follows. We define the format of our scientific simulation data in Section 2. In Section 3, we present AQSim's model generator and other notable statistical model generators. Section 4 describes future directions. Finally, Section 5 summarizes the material in this chapter.

2 Scientific Simulation Data in Mesh Format

Most scientific simulation programs generate data in *mesh* format. Mesh data sets commonly contain the following information:

i. *Zones*, which are distinct spatial elements. Interconnected grids on the x, y, and z axes in the Euclidean space generate zones. The shapes of the zones can be regular (*e.g.*, rectilinear) or irregular (*e.g.*, arbitrary polygons). Each zone is identified by its *x*, *y*, and *z* coordinates. For examples, a cubic zone contains eight $\langle x, y, z \rangle$ triples, which identify its corners.

ii. *Time steps*, which are discrete steps in the temporal space. Since data changes over time, it is stored at different time steps.

iii. *Physical variables*, which denote non-spatial and non-temporal information. For each step in time, physical variables can be assigned values at a zone's corners or its center.

Figure 3 depicts two mesh data sets produced by astrophysics simulations. The first simulation represents the explosion of a star. The second simulation depicts the interactions of the various components of a star at its mid-life. AQSim operates on scientific simulation data in mesh format.

The three major factors determining the size of a mesh data set are its number of zones, time steps, and physical variables (Abdulla, et al. 2003). Musick and Critchlow (1999) provide a nice introduction to scientific mesh data.

Figure 3. Part of a mesh data set representing the explosion of a star (on the left) and a star in its mid-life (on the right)

3 Statistical Model Generators for Scientific Data

This section presents three different approaches used in AQSim's model generator and briefly describes several other notable statistical model generators.

3.1 AQSim's Model Generator

AQSim's model generator builds statistical summaries of mesh data for description. To achieve this, the original data set is systematically partitioned (by utilizing a top-down approach) or agglomerated (by utilizing a bottom-up approach). The partitions/agglomerations divide the data in the spatio-temporal region. Then, statistical models of the data are either built on top of the partitions/agglomerations or independently constructed and subsequently associated with each agglomeration. This subsection describes three of AQSim's model generation strategies: (*i*) top-down partitioning with univariate modelers, (*ii*) bottom-up agglomeration with a univariate modeler, and (*iii*) bottom-up agglomeration with a multivariate modeler.

3.1.1 Top-Down Partitioning with Univariate Modelers

AQSim's top-down partitioning algorithm employs a four-way octree-like partitioning on the spatio-temporal space (see Figure 4). In particular, a four-way bisection on the *x*, *y*, *z*, and *time* dimensions repeatedly subpartitions the data. The major advantage of this algorithm is the generation of a global decomposition of the data. However, this global partitioning comes with several drawbacks. First, it is computationally too expensive to scale to tera-byte data sets. This is largely due to its need to convert a mesh data file from its original simulation-specific format into a consistent vector-based

representation. Second, it is not able to capture a mesh data set's topology, which stores the connectivity relationships between the zones. Third, this approach is closely coupled with the models chosen to represent the data in each partition. In particular, this algorithm repeatedly divides the data until the a-priori defined model error for a partition falls below a user-specified threshold (Eliassi-Rad, et al. 2002). Thus, the bisection procedure works best when there is a uniform density throughout the whole problem domain. Despite these shortcomings, our empirical evaluations illustrate the value of this simple partitioning when combined with either the mean modeler or the goodness-of-fit modeler.

Figure 4. Top-down partitioning of the data at a particular time step

3.1.1.1 Mean Modeler

Each top-down partition of the data has a set of variables associated with it. For each variable v_i, the mean modeler is μ_i, where μ_i is the mean of the data points associated with v_i in partition p_k.

For the mean modeler, partitioning of the data stops when either one of the following two conditions is true:

1. $\forall v \in NonPartitioningVariables$ in node η, $\sigma_v = 0$.
2. $\forall v \in NonPartitioningVariables$ in node η, $(\mu_v - (c \times \sigma_v) \leq min_v)$ & $(max_v \leq \mu_v + (c \times \sigma_v))$.

The first stopping criterion represents the simple case of partitions with either 1 data point or a set of data points with standard deviation of zero. In the second stopping criterion, the partition threshold, c, is a real number greater than or equal to zero. This user-defined threshold is a scaling factor for the standard deviation of variable v. For example, $c = 1$ means that the minimum and maximum values for each non-partitioning variable must be within 1 standard deviation of the mean of the data points in the node.

Since the *true mean* (which is an unbiased estimator) is used as the model, standard deviation is the same as *RMSE* (root mean square error). We use each variable's RMSE to represent the error

associated with the variable's model. Consequently, c can be thought of as regulating the relative error allowed in each variable's model. The advantage of the above stopping criteria is that it does not assume any distribution on the data points.

3.1.1.2 Goodness-of-Fit Modeler

For each variable v_i in partition p_k, the goodness-of-modeler is $N(\mu_i, \sigma_i)$. That is, the model for v_i is a normal distribution with mean, μ_i, and standard deviation, σ_i.

For the goodness-of-fit modeler, partitioning stops when the hypothesis test for normality is not rejected. For our goodness-of-fit test, we use the *Anderson-Darling test for normality*, which is considered to be the most powerful goodness-of-fit test for normality (D'Agostino and Stephens, 1986).

The Anderson-Darling test involves calculating the A^2 *metric* for variable $v_i \sim N(\mu_i, \sigma_i)$, which is defined to be

$$A^2 = -\frac{1}{n}\left(\sum_{j=1}^{n}(2j-1)\left(\ln(z_j)+\ln(1-z_{n+1-j})\right)\right)-n$$

where n = number of data points for v_i and $z_j = \Phi(\frac{x_j - \mu_i}{\sigma_i})$. $\Phi(\bullet)$ is the standard normal distribution function. We reject H_0 if $A^2\left(1+\frac{0.75}{n}+\frac{2.25}{n^2}\right)$ exceeds the *critical value* associated with the user-specified error threshold (D'Agostino and Stephens, 1986). Otherwise, we accept H_0.

For each variable v_i, the error on this model is defined to be $Pr(reject\ H_0 \mid H_0\ is\ true)$, where H_0 is the null hypothesis and states that the distribution of the variable v_i is normal. In other words, the model error is equal to the Type I error.

3.1.1.3 Experiments with AQSim's Univariate Modelers

AQSim's univariate modelers were used to build statistical models a star in its mid-life. The data set, called *Djehuty-5*, has 18 variables, 16 time steps, and 1,625,000 zones. The variables associated with this data set are: time, *x* axis, *y* axis, *z* axis, distance, grid vertex values, grid movement along the *x* and *y* axes, d(energy)/d(temperature), density, electron temperature, temperature due to radiation, pressure, artificial viscosity, material temperature, material velocity along the *x*, *y*, and *z* axes. Figure 5a depicts this data set in its first time step when all the 1,625,000 zones are plotted.

Table 1 lists the compression results on Djehuty-5 for the mean modeler. Recall that the partition threshold for this modeler restricts the difference between the minimum and maximum values of a variable and its mean value with respect to RMSE.

For AQSim's mean modeler experiments, Figure 5b shows Djehuty-5 at its first time step when the query *time > 0* is posed with no constraint on execution time and with partition threshold of 3.00. As expected, we get better compression as the partition threshold for the mean modeler increases since we are allowing the range of values for a variable to also increase. However, as you see in Figure 5b, even with 92.1% compression, we are able to return a highly precise answer.

Figure 5. (a) The actual simulation of the star its first time step. (b) The mean modeler's representation of the first time step with partition threshold of 3.00. (c) The goodness-of-fit modeler's representation of the first time step with partition threshold of 99.99%

Table 1. Mean Modelers' Compression Results on the Djehuty-5

Table 2 lists the compression results on Djehuty-5 for the goodness-of-fit modeler. Recall that the partition threshold in this table represents the confidence region of our normality test, which is equal to 100 × (1 – Type I error).

Table 2. Goodness-of-Fit Modeler's Compression Results on Djehuty-5

For our goodness-of-fit modeler experiments, Figure 5c shows Djehuty-5 at its first time step when the query *time > 0* is posed with no constraint on execution time and with partition threshold of 99.99%. Again not surprisingly, we get better compression as the partition threshold for the goodness-of-fit modeler increases since the confidence region shrinks. However, as you see in Figure 5c, even with 94.3% compression, we are able to return a highly precise answer

Our experimental results illustrate the value of using simple statistical modeling techniques on scientific simulation data sets. Both of our approaches require only one sweep of the data and generate models that compress the data up to 94%.

The goodness-of-fit modeler performed better than the mean modeler on Djehuty-5. This is not surprising since Djehuty-5 describes a physical phenomenon and the goodness-of-fit modeler is biased towards such normally distributed data sets. In general, we prefer the mean modeler since it makes no assumption on the data set.

3.1.2 Bottom-up Agglomeration with a Univariate Modeler

AQSim's bottom-up partitioning algorithm overcomes the aforementioned shortcomings of the top-down approach by utilizing the intrinsic topology of the data given in the original scientific problem. The topology of a mesh data is the (true physical) connectivity information of its zones. For AQSim's bottom-up algorithm, we remove the time dimension from the partitioning space and redefine our partitions on the three-dimensional spatial structure of the data. This new partition space allows us to produce agglomerations that can easily be parallelized for data access.

AQSim's bottom-up algorithm starts at the mesh data's initial grid configuration (*i.e.*, at the zones). From this fine level collection of grid cells, it iteratively produces *coarse* level collections of cells (see Figure 6). In particular, it uses a two-pass approach. In the first pass, each coarse cell is assigned the "best" neighborhood configuration (with respect to its rectilinear cell shape). This

operation is a local search on the 2^N possible neighborhood configurations of a coarse cell, where N is the number of dimensions. For instance, in two dimensions, the grey boxes in Figure 7 denote the four possible locations for a given cell within a coarse agglomeration.

Figure 6. AQSim's bottom-up algorithm generates a *topology tree*.

Figure 7. Four possible locations for a cell within a coarse agglomeration in two dimensions

Figure 8 illustrates the first step in the agglomeration of a collection of fine cells. The first pass of the bottom-up algorithm starts at fine cell $^{\#}1$ and agglomerates the fine cells $^{\#}1$, $^{\#}2$, $^{\#}5$, and $^{\#}6$ into coarse cell *C1*. Then, it investigates the neighborhood configuration around fine cell $^{\#}3$ (which is the next unassigned fine cell) and builds coarse cell *C2* from fine cells $^{\#}3$, $^{\#}4$, $^{\#}7$, and $^{\#}8$. Finally, it explores the neighborhood configuration around fine cell $^{\#}9$ and groups fine cells $^{\#}9$, $^{\#}10$, $^{\#}11$, and $^{\#}12$ into coarse cell *C3*.

Figure 8. A non-quad tree coarse cell arrangement

The first pass of the bottom-up algorithm is a local operation on cells. That is, when creating coarse cells from fine cells, no information about the past and future agglomerations in other regions of the domain is taken into consideration. For this reason, some coarse agglomerations can result in trees that are non-binary, non-quad, or non-octree. For instance, it is easy to be in a situation where the coarse cells are arranged as shown in Figure 8 after the first pass.

The coarse cells (*C1*, *C2*, and *C3* given by solid lines) have been arranged in such a way that indeterminate behavior for neighbors exists for the coarse cells. For example, *C2* has two neighbors to its right. A second pass corrects such structural problems associated with indeterminate behavior for neighbors of coarse cells. In particular, the second pass has N-dimensional subphases. Each subphase, s, uses information from all the previous subphases to correctly place the $(N - s)$ dimensional structures, planes, lines, and points. Conceptually, this corresponds to sliding the coarse cells so they line up

nicely. It is important to note that the second pass only adjusts each coarse cell's neighbor relations (if needed). For example, in two dimensions, the problem illustrated in Figure 8 can be fixed by (*i*) adjusting the face neighbors so that cell *C2* "slides" down half of a coarse grid cell and (*ii*) making sure the neighbors for all local coarse cells reflect this slide (see Figure 9). The *C1*, *C2*, and *C3* cells can then be combined at the next level in the agglomeration to form the coarse cell *C4* (which in this case is also the root of the topology tree).

Figure 9. A fix for a non-quad tree coarse cell arrangements

A heuristically complex procedure is used to compute the corrections in the bottom-up algorithm's second pass. Our correction procedure utilizes the information about the (faces, edges, and corners of) neighbors of the coarse cells' descendents to establish neighbors at the coarse level. For instance, to find the neighbors for *C2* (shown in Figure 8), we utilize the information for neighbors of fine cells $^{\#}1$, $^{\#}2$, $^{\#}5$, $^{\#}6$, $^{\#}9$, $^{\#}10$, $^{\#}11$, and $^{\#}12$.

After the topology tree is constructed, the statistical information associated with the mean modeler is propagated upwards through each level of the tree. The information about the number of data points (*n*), minimum (*min*), and maximum (*max*) values of the data points are calculated as follows:

$$n = \sum_i n_i \qquad \min = \min_i(\min_i) \qquad \max = \max_i(\max_i)$$

When calculating the mean and standard deviation values, we do not use the number of data points, instead we rely on the volume of the coarse cell and its associated fine cells. The volume of a cell is the amount of non-empty space that it occupies in the three-dimensional Euclidean space. By making this switch, we are able to avoid any assumptions about the distribution of the data in the fine cells. Therefore, the probability (or weight) of each fine cell's mean and standard deviation measure is

based on $\frac{vol_i}{vol}$, where $vol = \sum_i vol_i$. Thus, the mean (μ) and standard deviation (σ) measures of a coarse cell are calculated by using the following equations:

$$\mu = \frac{\sum_i (\mu_i \times vol_i)}{vol} \qquad \sigma = \sqrt{\frac{\sum_i ((\sigma_i^2 + \mu_i^2) \times vol_i)}{vol} - \mu^2}$$

Note that by using the sum of the volumes of the fine cells, we do not inflate the coarse cell's volume artificially in cases where the coarse cell is not "full" (*e.g.*, *C4* in Figure 9).

3.1.3. Bottom-up Agglomeration with a Multivariate Modeler

In this section, we continue utilizing the bottom-up agglomeration algorithm of the previous section but switch from a univariate modeler to a multivariate one. Our motivation for using a multivariate modeling algorithm is to capture the interrelationships among a mesh data set's physical variables in one metric.[6] In this way, we are able to collectively measure the similarity between zones.

Table 3 describes AQSim's multivariate clustering algorithm. The inputs to our algorithm are (*i*) the list of zones in the mesh data set and (*ii*) a user-defined clustering threshold in [0,1]. A clustering threshold of 0 indicates complete dissimilarity between zones. On the other hand, a clustering threshold of 1 shows total similarity among zones. The output of our algorithm is a list of clusters, where each cluster is represented by its (zonal) population, *N*, and the following four vectors:

$$\vec{\mu} = \begin{pmatrix} \mu_1 \\ \mu_2 \\ \cdots \\ \mu_m \end{pmatrix} \quad \vec{\sigma} = \begin{pmatrix} \sigma_1 \\ \sigma_2 \\ \cdots \\ \sigma_m \end{pmatrix} \quad \overrightarrow{max} = \begin{pmatrix} max_1 \\ max_2 \\ \cdots \\ max_m \end{pmatrix} \quad \overrightarrow{min} = \begin{pmatrix} min_1 \\ min_2 \\ \cdots \\ min_m \end{pmatrix}$$

[6] Recall that physical variables are all variables except the spatial (*i.e.*, *x*, *y*, *z*) and temporal (*i.e.*, *time*) variables.

$\vec{\mu}$, $\vec{\sigma}$, $\vec{max}$, and $\vec{min}$ are the mean, standard deviation, maximum, and minimum values of the zones represented in a cluster. The variable *m* is the number of physical variables in the mesh data set.

Initially, we assume that all zones are available for clustering (*i.e.*, they are assigned the color **GREEN**). Then, as each zone is placed in an appropriate cluster, it becomes unavailable (*i.e.*, its color changes from **GREEN** to **RED**). Our clustering algorithm iterates over all **GREEN** zones. At each iteration, a variant of the cosine similarity measure (Van Rijsbergen, 1979) is used to find an appropriate cluster for each **GREEN** zone.

Table 3. AQSim's Multivariate Clustering Algorithm

The standard cosine similarity measure is a function of two vectors $\vec{\alpha}$ and $\vec{\beta}$ of equal length. In particular, it is defined as follows:

$$CosSim(\vec{\alpha},\vec{\beta}) = \begin{cases} \cos(\vec{\alpha},\vec{\beta}) = \dfrac{\vec{\alpha}\bullet\vec{\beta}}{\|\vec{\alpha}\|_2 \times \|\vec{\beta}\|_2}, \text{if } \vec{\alpha}\neq 0 \text{ and } \vec{\beta}\neq 0 \\ 1, \text{if } \vec{\alpha}=\vec{\beta} \\ 0, \text{if } (\vec{\alpha}=0 \;\&\; \vec{\beta}\neq 0) \text{ or } (\vec{\alpha}\neq 0 \;\&\; \vec{\beta}=0) \end{cases}$$

AQSim utilizes the aforementioned *CosSim* metric by representing a zone's physical data points as a vector. In particular, such a vector contains the mean values of a zone's physical variables. For example, if a mesh data set contains only two physical variables (say *temperature* and *pressure*), then each zone is represented by a vector of two elements (namely, one value for temperature and another for pressure). AQSim's *CosSim* normalizes the elements of its vectors such that all the values of physical variables are between 0 and 1. This normalization step is important since the ranges of values for our physical variables differ considerably.

Suppose, the vectors $\vec{\alpha}$, $\vec{\beta}$, and $\vec{\gamma}$ represent normalized mean values of physical variables for three zones in a mesh data set. Traditionally, when $CosSim(\vec{\alpha},\vec{\beta})$ is greater than or equal to the user-defined similarity threshold, $\vec{\alpha}$ and $\vec{\beta}$ are placed into one cluster. Then, a new vector, $\vec{\gamma}$, is placed in the same cluster as $\vec{\alpha}$ and $\vec{\beta}$ only if both of the following inequalities are true:

$$CosSim(\vec{\alpha},\vec{\gamma}) \geq user\_threshold \text{ and } CosSim(\vec{\gamma},\vec{\beta}) \geq user\_threshold$$

Since we have a huge number of zones to process, calculations of all pair-wise *CosSim* measures can be quite burdensome. Furthermore, sampling from the mesh data is not an option since scientists are interested in outliers and do not tolerate results from sampled data. However by increasing the similarity threshold, we are able to eliminate the need to compute all the pair-wise comparisons. Specifically, if we increase the user-defined similarity threshold by using the following function:

$$f(u) = \cos\left(\frac{\cos^{-1}\left((2\times u)-1\right)}{2}\right)$$

where u is the user-defined similarity threshold.[7] The function $f(u)$ returns the *cosine* of an angle that is equal to half the angle of the user-defined similarity threshold. Based on the geometrical properties of the *cosine* function, we obtain the following inequality: $((2 \times u) - 1) < f(u)$.[8] Thus, if we encounter a zone with vector $\vec{\beta}$ and $f(u) \leq CosSim(\vec{\alpha},\vec{\beta})$ for <u>any</u> $\vec{\alpha}$ in a cluster C, then it must be the case that $CosSim(\vec{\alpha},\vec{\beta}) \geq f(u)$ for <u>all</u> $\vec{\alpha}$ in cluster C. In other words, we are guaranteed that any new zone added to a cluster, C, satisfies the user-defined similarity measure. Figure 10 pictorially illustrates this fact. Based on the user's threshold, any two vectors $\vec{\alpha}$ and $\vec{\beta}$ are considered similar if and only if the

[7] The original user-defined threshold is mapped from [0,1] to [-1, 1] by the equation $((2 \times u) - 1)$.

[8] The *cosine* of any angle g is always less than the *cosine* of the angle $\frac{g}{2}$.

cosine of the angle between $\vec{\alpha}$ and $\vec{\beta}$ is greater than or equal to $f(u)$. In our clustering algorithm, if there exists some cluster C with a vector $\vec{\gamma}$ and the angle between $\vec{\gamma}$ and any vector $\vec{\beta}$ is half the size of the angle between $\vec{\alpha}$ and $\vec{\beta}$, then $\vec{\beta}$ is safely added to the cluster C (without any additional pair-wise comparisons). In other words, for each cluster C, we only need to compute the *CosSim* metric for an unassigned (**GREEN)** zone and the first zone added to C.

Figure 10. User-defined threshold and AQSim's threshold

Increasing the user-defined similarity threshold helps us in two ways. First, we do not spend time on pair-wise comparisons. Second, we do not need to shuffle zones between clusters since our bound, $f(u)$, eliminates zones that are on the cluster boundaries. In the best case, our clustering algorithm runs in $O(n)$ time, where n is the number of zones in the mesh data sets. In the worst case, our algorithm runs in $O(n^2)$ time. However, on average, our algorithm runs in $O(n \times g(f(u)))$ times, where $g(f(u))$ is the number of zones that satisfy the $f(u)$ bound. Eliassi-Rad and Critchlow (2003) empirically showed that on average g($f(u)$) is much less than n.

Even though spatial variables do not play a role in building clusters in AQSim's multivariate clusterer, it is desirable to associate each cluster with its correct spatial region. In particular, it is important for the multivariate clusterer to return answers to scientists' queries in the spatial region of the original mesh. Since our mesh data sets have millions of zones, it would be very inefficient (and at times impossible) to link each cluster with all of its zones. Therefore, we present a linking algorithm for connecting each cluster to a small set (*e.g.*, 512) of nodes in the data set's *topology tree* (constructed using the algorithm in Section 3.1.2). Recall that a topology tree stores the spatial information of a data set by utilizing the intrinsic topology of the data given in the original scientific problem (see Figure 6).

Figure 11 shows sample links between the list of clusters and the topology tree. Our linking technique is embedded into the multivariate clustering algorithm. That is, connections between a cluster

and the correct nodes of the topology tree are made as the cluster is being constructed. In this way, we are able to avoid traversing the clusters after they are made, which in turn reduces our execution times. The main challenge for our linking algorithm is to find the "best" *k* nodes in the topology tree for a particular cluster, *C*, where *k* (*e.g.*, 512) is much less than the number of zones in *C* (which typically range in the thousands).

Figure 11. Links between list of clusters and topology tree

Table 4 describes the multivariate clusterer's linking algorithm. When the list of links for a cluster is full, we traverse the topology and the list of links to find the lowest level topology node with the most number of descendents in the list of links. Each link between a cluster and a tree node has a metric, called *percentage_intersection*, which measures the percentage intersection between the zones in the clusters and the descendents of a node. For example, if a zone, *z*, is in cluster *C* and the list of links for *C* has a link connected to *z*, then that link's *percentage_intersection* is 100. This *percentage_intersection* metric measures the "quality" of a link. A threshold can be placed on the *percentage_intersection* metric so that the trade-off between execution time and links to the best-fitting nodes can be exploited. As we will show in the next section, our linking algorithm performs quite well. This is mainly due to the very small number of levels in a topology tree (usually less than 20).

Table 4. AQSim's Linking Algorithm

3.1.3.1 Experiments with AQSim's Multivariate Modeler

Table 5 describes the two mesh data sets used in the experiments on AQSim's multivariate modeler. Both data sets are a simulation of a star at a certain stage of its life and represent readings in point locations of a continuous medium. The data sets use cubic zones (with eight vertices each). Values of variables are associated either with each vertex or with the center of each zone. The White

Dwarf data set (see Figure 12a) is a simulation of a star exploding. The Djehuty-5 data set (see Figure 12b) is a simulation of a star at its mid-life.

Table 5. Characteristics of our two astrophysics data sets

Our experiments describe the performance of AQSim's multivariate clusterer with and without the linking algorithm on the aforementioned two simulation data sets. We also compare these performances to the bottom-up agglomeration algorithm, where the statistical information of the physical variables is propagated upwards in the tree (see Section 3.1.2).

Figure 12. (a) The White-Dwarf data set and (b) Djehuty-5 data set

Figure 13 shows the number of clusters constructed *per time step* for White Dwarf. The user-defined clustering threshold for the blue and pink lines are 0.95 (in [0,1]) and 0.99 (in [0,1]), respectively. An intuitive way of thinking about the user-defined clustering threshold is to state that the user requires all zones in a cluster to be (*user_threshold* × 100) percent similar. So, when *user_threshold* is 0.95, the required similarity between zones in a cluster is 95%. AQSim's threshold tightens the 95% and 99% similarities among the zones in a cluster to 98.5% and 99.75% similarities, respectively.

Notice the small number of clusters that we are able to build from hundreds of thousands of zones. There is a dramatic increase in the number of cluster made for the White Dwarf data set from time step = 0 to time step = 1 (due to the start of explosion in the star). Tables 6 and 7 summarize the minimum, average, and maximum number of clusters and execution times for White Dwarf with the given clustering thresholds, respectively. As expected, the execution times are also longer with the bound of 99.75% similarity as opposed to a bound of 98.5% similarity.

Figure 13. Number of clusters made for White Dwarf with AQSim's threshold of 98.5% & 99.75%

Table 6. Clustering on White Dwarf (*user_threshold* = 95% and *f*(*u*) = 98.5%)

Table 7. Clustering on White Dwarf (*user_threshold* = 99% and *f*(*u*) = 99.75%)

Figure 14 shows the number of clusters constructed per time step for Djehuty-5. Similar to experiments on White Dwarf, the user-defined clustering threshold for the blue and pink lines are 95% and 99%, respectively. Again, AQSim's threshold tightens the 95% and 99% similarities among the zones in a cluster to 98.5% and 99.75% similarities, respectively.

Figure 14. Number of clusters made for Djehuty-5 with AQSim's threshold of 98.5% & 99.75%

Notice again the small number of clusters that we are able to build from more than 1.6 million zones. Tables 8 and 9 summarize the minimum, average, and maximum number of clusters and execution times for Djehuty-5 with the given clustering thresholds, respectively. Again, as expected, the execution times are also longer with the bound of 99.75% similarity as opposed to a bound of 98.5% similarity.

Table 8. Clustering on Djehuty-5 (*user_threshold* = 95% and *f*(*u*) = 98.5%)

Table 9. Clustering on Djehuty-5 (*user_threshold* = 99% and *f*(*u*) = 99.75%)

Tables 10 and 11 show the execution times of multivariate clusterer with and without linking on the first six time steps of White Dwarf and Djehuty-5, respectively. In addition, they list the best and worst quality of links made between clusters and the topology tree. As expected, it takes longer to build clusters and connect them to the topology tree. However, the most increase is by a factor of 5.6 (see time step = 0 in Table 10). Both the White Dwarf and the Djehuty-5 topology trees have a maximum of 11 levels. The highest level in the tree reached for a connection between a cluster and a node is 4 and 5 for White Dwarf and Djehuty-5, respectively. This is quite good since a link accessing a high level in the tree usually has a worse fit (*i.e.*, *percentage_intersection*) as compared to a link connecting to a lower level in the tree. Even so, in both data sets, the maximum *percentage_intersection* at the highest level is 100% for both data sets. That is, for the Djehuty-5 data set, there are nodes at level 5 which

contain all the zones in a cluster. The minimum *percentage_intersection* at the highest level is 25% and 12.5% for White Dwarf and Djehuty-5, respectively. This minimum *percentage_intersection* value shows the quality of the worst links. For example, in the Djehuty-5 data set, there are nodes at level 5 which contain only 12.5% of the zones in a cluster.

Table 10. Clustering on White Dwarf (*user_threshold* = 99% and *f*(*u*) = 99.75%)

Table 11. Clustering on Djehuty-5 (*user_threshold* = 99% and *f*(*u*) = 99.75%)

Table 12 illustrates the execution times of AQSim's multivariate clusterer (without linking). As was expected, the average value of $g(f(u))$, which is the number of zones satisfying our threshold $f(u)$, is much less than n (the total number of zones). In fact, the average $g(f(u))$ is in single digits while the average n is in hundreds of thousands.

Table 12. Execution Times for Our Multivariate Clustering Algorithm (without Linking) and Average Value for *g*(*f*(*u*))

Table 13 depicts average distortion within the clusters that were made for White Dwarf and Djehuty-5. This average distortion measures the diversity within the clusters. For each cluster c with mean $\vec{\mu}_c$, standard deviation $\vec{\sigma}_c$, maximum $\vec{max}_c$, minimum $\vec{min}_c$, the distortion within clusters is computed by finding the value for d in these two equations: $(\vec{\mu}_c - d \times \vec{\sigma}_c) = \vec{min}_c$ and $\vec{max}_c = (\vec{\mu}_c + d \times \vec{\sigma}_c)$. The average distortion is then computed by taking the average of the d values for all physical variables in the mesh data. When an average d is equal to 3.0, it implies that on average the minimum and maximum values within a cluster are three standard of deviations away from its mean. As we showed in Section 3.1.1.3, AQSim is able to produce excellent results within this range. As expected the distortion within clusters decreases as the similarity threshold increases.

Table 13. Average Distortion within Clusters

Tables 14 and 15 compare the multivariate clusterer with linking to the bottom-up agglomeration algorithm with the mean modeler. The number of nodes made by the bottom-up agglomeration algorithm (in one level) is much larger than the number of clusters made by our clustering algorithm. This is mostly due to the design of the bottom-up agglomeration algorithm, which combines no more than eight zones at a time. This strategy of agglomerating only small number of zones based solely on their topology also makes the bottom-up agglomeration algorithm run faster than our clustering algorithm.

Table 14. White Dwarf Data Set: Comparison of multivariate clusterer (with linking) versus the bottom-up agglomeration (with mean modeler)

Table 15. Djehuty-5 Data Set: Comparison of multivariate clusterer (with linking) versus the bottom-up agglomeration (with mean modeler)

3.2. Other Model Generators

AQUA (Achraya, et al. 1999) is an approximate query answering system. It uses cached statistical summary of the data in *on-line analytical processing* (*a.k.a.* OLAP) applications. Unfortunately, AQUA uses sampling and histogram techniques to compute summaries of the data. Such techniques are not desirable for scientific data sets because by sampling you might miss outliers (which are important in scientific simulation data sets).[9] Moreover, histograms are computationally expensive on high-dimensional data.

STING (Wang, et al. 1997) is also similar to AQSim in that it builds a grid cell hierarchy and then propagates the data upwards in the hierarchy. However, STING assumes that the underlying distribution of the data is known. In most scientific phenomena, this assumption does not hold. In

[9] AQSim's statistical modelers capture outliers by not sampling and allowing users to specify error bounds on the models' descriptive statistics.

addition, STING has only been tested on small data sets containing only tens of thousands of data points.

Freitag and Loy (1999) describe an approach similar to AQSim for visualization of large scientific data sets. Their system builds distributed octrees from large scientific data sets. Unlike AQSim, they reduce the data points by constraining the physical data values to their spatial locations. In addition, they do not allow the user to query the octree; instead, the user can view the tree at different resolutions.

DuMouchel, et al. (1999) present a method for compressing flat files. However, they use binning techniques to "squash" files, which impose an *a priori* distribution on the data. For large-scale scientific data sets, it is not desirable to use binning techniques since *a priori* distribution on the data is unknown. Also, most simulation data are not stored in flat files and their conversion can be computationally burdensome (see Section 3.1.1).

Clustering algorithm such as BIRCH (Zhang, et al. 1996), CHAMELEON (Karypis, et al. 1999), CLARANS [16], CLIQUE (Agrawal, et al. 1998), CURE (Guha, et al. 1998), and DBSCAN (Ester, et al. 1996) cannot be used or scaled to large-scale simulation data sets for one or more of the following reasons:

i. They employ sampling techniques. Scientists already sample the data produced by their simulation programs. They do not accept models that sample the sampled data, particularly since they are mostly interested in outliers.

ii. They initially divide the data in subspaces and then build clusters locally on those subspaces. Such strategies are not desirable for clusters, which are supposed to capture the global properties of the simulation data.

iii. They utilize some kind of binning or histogram techniques, which can be computationally expensive on high-dimensional data sets. In addition, it is unrealistic to assume an *a priori* distribution on large-scale simulation data sets.

As for partitioning and agglomeration techniques, spatial data structures have been utilized in the database and visualization communities. The database community uses these structures to build efficient indexing algorithms for query processing. The visualization community uses them for adaptive mesh refinement and fast graphics display.

Samet (1989a, 1995) describes a method for using quad-trees and octrees to index objects in two- and three-dimensional spaces. His approach involves (*i*) laying an artificial grid on top of the data, (*ii*) splitting the background grid with a top-down methodology, and (*iii*) establishing spatial relations between the objects. Unlike AQSim, Samet's algorithms are not able to capture topological problems (such as singular edges) or construct trees that are independent of the data's distribution over the initial grid.

The visualization community provides sub-division methods for surfaces and objects. Arden (2001) approximates surfaces by choosing subsets of the original points that make up a surface and "fitting" the newly constructed surface through them. This work does not address topological problems. The multi-grid community has also examined issues related to grid coarsening (McCormick 1989). However, they do not deal with topological problems as well.

4 Future Directions

The most desirable modelers for large-scale scientific simulation data (*i*) require only one sweep of data, (*ii*) are good at finding outliers, (*iii*) can be easily parallelized, and (*iv*) can efficiently answer complex queries.

Separating the modeling process for the spatio-temporal variables and the physical variables is an interesting approach. In particular, if the clustering algorithm is able to capture similar behavior across time steps, then a cluster hierarchy can be constructed from each time steps. Subsequently, such hierarchies can speed-up response times for queries on both the physical variables and the spatio-temporal variables since the cluster hierarchy will be shallower than the topology hierarchy. The success of this approach will rely on scalable algorithms that find a manageable and representative number of links from the cluster hierarchy to the topology hierarchy.

Allowing the user to specify his/her desired similarity behavior for clusters is an important step in building non-generic clusters. In addition, tools that track particular zones as they change clusters across time steps will provide valuable insight into the scientific simulation and the similarity metric.

Scaling more intelligent partitioning algorithms such as K-d and AVL trees to massive data sets is another area for future research. In addition, efficient techniques are needed to capture the topology of mesh data sets with zones of arbitrary polygonal shapes.[10] Finally, algorithms that construct topology hierarchies with "non-octree like" structures need further investigations.

5 Conclusions

Massively parallel computer programs (which simulate complex scientific phenomena) generate large-scale data sets over the spatio-temporal space. Modeling such massive data sets is an essential step in helping scientists discover new information from these computer simulations. We present several systems that build queriable statistical models. Such models help scientists gather knowledge from their large-scale simulation data. In particular, we discuss AQSim, which consists of two components: (*i*) the model generator and (*ii*) the query processor. The model generator reduces the data storage requirements by creating and storing queriable statistical models of the data at multiple

[10] Such adaptive meshes ensure that the grid better conforms to the complex shapes such as tetrahedrons.

resolutions. The query processor decreases the query access times by evaluating queries on the models of the data instead of on the original data set. We describe three simple but effective statistical modeling techniques for simulation data. AQSim's univariate mean modeler computes the unbiased mean of systematic spatial-temporal partitions of the data. It makes no assumptions about the distribution of the data and uses a variant of the root mean square error to evaluate a model. AQSim's univariate goodness-of-fit modeler utilizes the Andersen-Darling goodness-of-fit method on systematic spatial-temporal partitions of the data. This modeler evaluates a model by how well it passes the normality test on the data. AQSim's multivariate clusterer utilizes the cosine similarity measure to cluster the physical variables in a data set. The exclusion of the spatial location is important since "similar" characteristics could be located (spatially) far from each other. To scale AQSim's multivariate clustering algorithm for large-scale data sets, we take advantage of the geometrical properties of the cosine similarity measure. This allows AQSim to reduce the modeling time from $O(n^2)$ to $O(n \times g(f(u)))$, where n is the number of data points, $f(u)$ is a function of the user-defined clustering threshold, and $g(f(u))$ is the number of data points satisfying the threshold $f(u)$. We show that on average $g(f(u))$ is much less than n. Finally, even though spatial variables do not play a role in building a cluster, it is desirable to associate each cluster with its correct spatial location. To achieve this, we present a linking algorithm for connecting each cluster to the appropriate nodes in the data set's topology tree. Our experimental analyses on two large-scale astrophysics simulation data sets illustrate the value of AQSim's model generator.

Acknowledgements

This work was performed under the auspices of the U.S. Department of Energy by the University of California Lawrence Livermore National Laboratory under contract No. W-7405-ENG-48.1. UCRL-BOOK-200992. We thank Bill Arrighi, Kevin Durrenberger, Susan Hazlett, Roy Kamimura, Nu Ai Tang, and Megan Thomas for their assistance.

References

1. Abdulla, G., Critchlow, T., Arrighi, W. (2003) Simulation Data as Data Streams, Lawrence Livermore National Laboratory (LLNL) Technical Report, UCRL-JC-152697.
2. Abdulla, G., Baldwin, C., Critchlow, T., Kamimura, R., Lozares, I., Musick, R., Tang, N.A., Lee, B., and Snapp, R. (2001) Approximate ad-hoc query engine for simulation data. In *Proceedings of the 1st ACM+IEEE Joint Conference on Digital Libraries (JCDL)*, ACM Press, pp. 255-256.
3. Adelson-Velskii, G.M., and Landis, E.M. (1962) An Algorithm for the Organization of Information, *Soviet Math. Dokl*., vol. 3, pp. 1259-1262.
4. Acharya, S., Gibbsons, P.B., Poosala, V., and Ramaswamy, S. (1999) The Aqua approximate query answering system. In *Proceedings of the 1999 ACM SIGMOD Conference*, ACM Press, pp. 574-576.
5. Agrawal, R., Gehrke, J., Gunopulos, D., Raghavan, P. (1998) Automatic subspace clustering of high dimensional data for data mining applications, In *Proceedings of the 1998 ACM SIGMOD Conference*, ACM Press, pp. 94-105.
6. Arden, G. (2001) *Approximation Properties of Subdivision Surfaces*, PhD Thesis, University of Washington, Seattle.
7. Baldwin, C., Abdulla, G., and Critchlow, T. (2003a) Multi-Resolution Modeling of Large Scale Scientific Simulation Data, In *Proceedings of the 12th International Conference on Information and Knowledge Management*, ACM Press, pp. 40-48.
8. Baldwin, C., Eliassi-Rad, T., Adbulla, G., and Critchlow, T. (2003b) The evolution of a hierarchical partitioning algorithm for large-scale scientific data: three steps of increasing complexity, In *Proceedings of the 15th International Conference on Scientific and Statistical Database Management*, IEEE Computer Society, pp. 225-228.

9. Bowers, R.L., and J.R. Wilson (1991) *Numerical Modeling in Applied Physics and Astrophysics*, Jones & Bartlett Publishers, Boston.
10. Chakrabarti, K., Garofalakis, M., Rastogi, R., and Shim, K. (2000) Approximate query processing using wavelets, In *Proceedings of the 26th International Conference on Very Large Data Bases*, Morgan Kaufmann, pp. 111-122.
11. Cormen, T.H., Leiserson, C.E., and Rivest, R.L. (1989) *Introduction to Algorithms*, McGraw-Hill.
12. D'Agostino, R.B., and Stephens, M.A. (1986) *Goodness-of-fit Techniques*, Marcel Dekker, Inc.
13. Devore, J.L. (1991) *Probability and Statistics for Engineering and the Sciences*, 3rd edition. Brooks/Cole Publishing Company, Pacific Grove, CA.
14. DuMouchel, W., Volinsky, C., Johnson, T., Cortes, C., and Pregibon, D. (1999) Squashing flat files flatter, In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM Press, pp. 6-15.
15. Eliassi-Rad, T., Critchlow, T. (2003) Multivariate Clustering of Large-Scale Scientific Simulation Data, Lawrence Livermore National Laboratory (LLNL) Technical Report, UCRL-JC-151860-REV-1.
16. Eliassi-Rad, T., Critchlow, T., and Abdulla, G. (2002) Statistical modeling of large-scale simulation data, In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM Press, pp. 488-494.
17. Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. (1996) A density-based algorithm for discovering clusters in large spatial databases with noise, In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, AAAI Press, pp. 226-231.

18. Freitag, L.A., and Loy, R.M. (1999) Adaptive, multiresolution visualization of large data sets using a distributed memory octree. In *Proceedings of the 1999 Supercomputing Conference*, ACM Press, Article 60.

19. Friedman, J.H., Bentley, J.L., and Finkel, R.A. (1977) An Algorithm for Finding Best Matches in Logarithmic Expected Time, *ACM Transactions on Mathematical Software*, vol 3, ACM Press, pp. 209-226.

20. Guha, S., Rastogi, R., Shim, K. (1998) Cure: An efficient clustering algorithm for large databases, In *Proceedings of the 1998 ACM SIGMOD Conference*, ACM Press, pp. 73-84.

21. Hilderman, R.J., and Hamilton, H.J. (2001) *Knowledge Discovery and Measures of Interest*, Kluwer Academic Publishers, Boston, MA.

22. Karypis, G., Han, E.-H., Kumar, V. (1999) Chameleon: Hierarchical clustering using dynamic modeling, *IEEE Computer*, pp. 68-75.

23. Lee, B., Critchlow, T., Abdulla, G., Baldwin, C., Kamimura, R., Musick, R., Snapp, R., and Tang, N.A. (2003) The framework for approximate queries on simulation data, *International Journal of Information Sciences*, vol. 157, no. 1-4, Elsevier Sciences.

24. McCormick, S. (1989) Multilevel Adaptive Methods for Partial Differential Equations, *Frontiers in Applied Mathematics*, vol. 6, Society for Industrial and Applied Mathematics.

25. Musick, R., and Critchlow, T. (1999) Practical lessons in supporting large-scale computational science, In *Proceedings of the 1999 SIGMOD Record*, ACM Press, vol. 28, no. 4, pp. 49-57.

26. Ng, R.T., and Han, J. (1994) Efficient and effective clustering methods for spatial data mining, In *Proceedings of the 20th International Conference on Very Large Data Bases*, Morgan Kaufmann, pp. 144-155.

27. Samet, H. (1995) Spatial Data Structures, In W. Kim (Ed.), *Modern Database Systems: The Object Model, Interoperability, and Beyond*, Addison Wesley/ACM Press, pp. 361-385.
28. Samet, H. (1989a) The Design and Analysis of Spatial Data Structures, Addison-Wesley.
29. Samet, H. (1989b) Applications of Spatial Data Structures: Computer Graphics, *Image Processing and GIS*, Addison-Wesley.
30. Todorovski, L., and Dzeroski, S. (2001) Using domain knowledge on population dynamics modeling for equation discovery. In *Proceedings of the 12th European Conference on Machine Learning*, Springer. pp. 478-490
31. Van Rijsbergen, C.J. (1979) *Information Retrieval*, 2nd edition, Butterworths, London, UK.
32. Wang, J., Wang, X., Lin, K.-I., Shasha, D., Shapiro, B.A., Zhang, K. (1999) Evaluating a class of distance-mapping algorithms for data mining and clustering, In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM Press, pp. 307-311.
33. Wang, W., Yang, J., and Muntz, R. (1997) STING: A statistical information grid approach to spatial data mining, In *Proceedings of the 23rd International Conference on Very Large Data Bases*, Morgan Kaufmann, pp. 186-195.
34. Zhang, T., Ramakrishnan, R., and Livny, M. (1996) BIRCH: An efficient data clustering method for very large databases, In *Proceedings of the 1996 ACM SIGMOD Conference*, ACM Press, pp. 103-114.

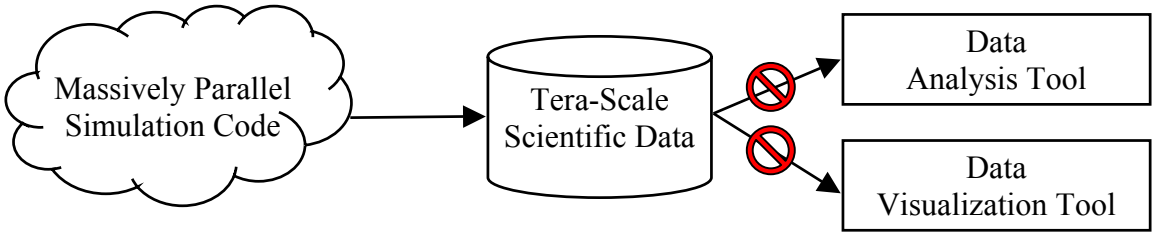


Figure 1. Currently available analysis and visualization tools cannot efficiently process tera-scale scientific data.

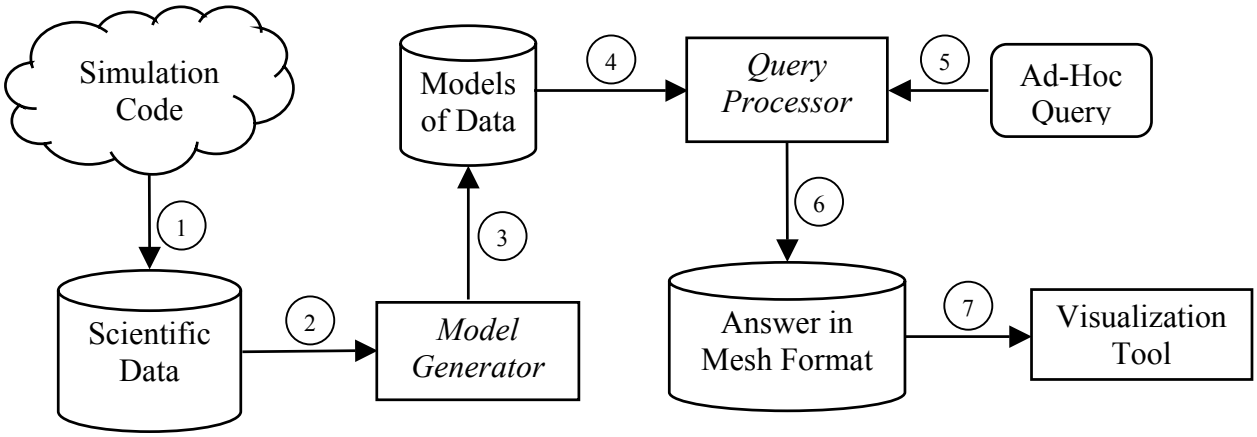


Figure 2. AQSim's architecture

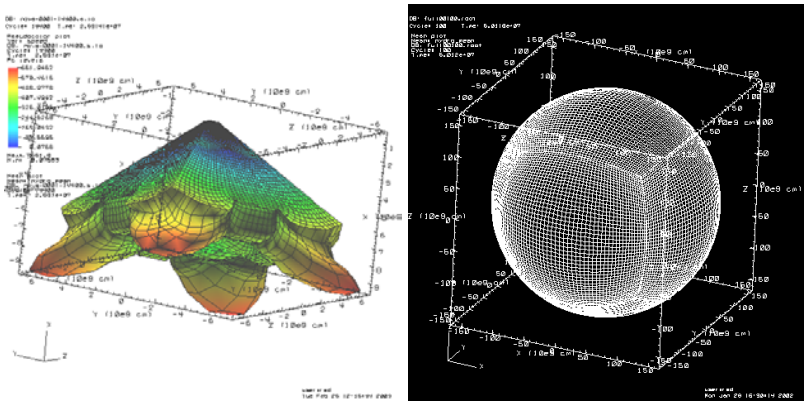


Figure 3. Part of a mesh data set representing the explosion of a star (on the left) and a star in its mid-life (on the right)

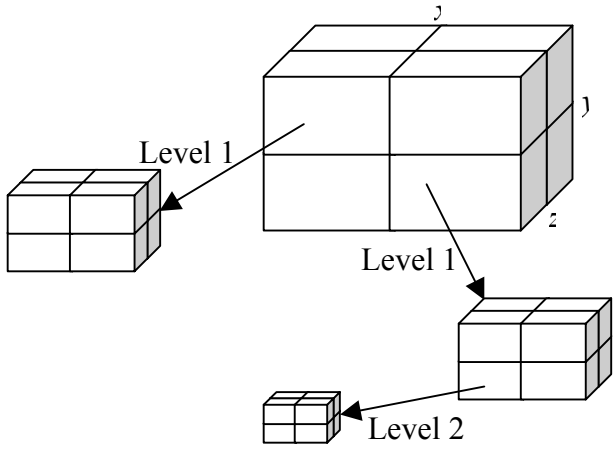
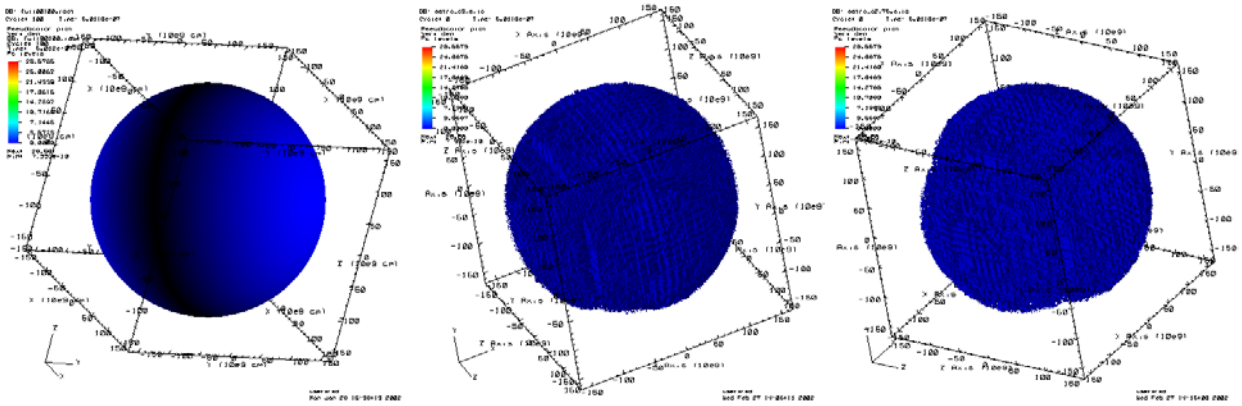


Figure 4. Top-down partitioning of the data at a particular time step



(a) **(b)** **(c)**

Figure 5. (a) The actual simulation of the star its first time step. (b) The mean modeler's representation of the first time step with partition threshold of 3.00. (c) The goodness-of-fit modeler's representation of the first time step with partition threshold of 99.99%

Table 1. Mean Modelers' Compression Results on the Djehuty-5

Partition Threshold	% of Compression	Total # of Partitions	% of Non-Leaf Partitions	% of Leaf Partitions	Avg. # of Data Point in a Partition
1.75	67.4	728,081	17.9	82.1	2.9
2.00	70.1	511,395	17.8	82.2	4.1
2.25	79.7	347,471	17.7	82.3	6.0
2.50	85.8	242,840	18.7	81.3	8.7
2.75	89.6	177,448	19.0	81.0	11.9
3.00	92.1	135,548	17.8	82.2	15.3

Table 2. Goodness-of-Fit Modeler's Compression Results on Djehuty-5

% Partition Threshold	% of Compression	Total # of Partitions	% of Non-Leaf Partitions	% of Leaf Partitions	Avg. # of Data Point in a Partition
80.0	66.7	564,718	16.8	83.2	3.6
85.0	71.2	492,029	16.7	83.3	4.2
90.0	76.4	404,136	16.9	83.1	5.1
95.0	82.8	293,585	16.8	83.2	7.0
99.99	94.3	97,819	13.3	86.7	20.2

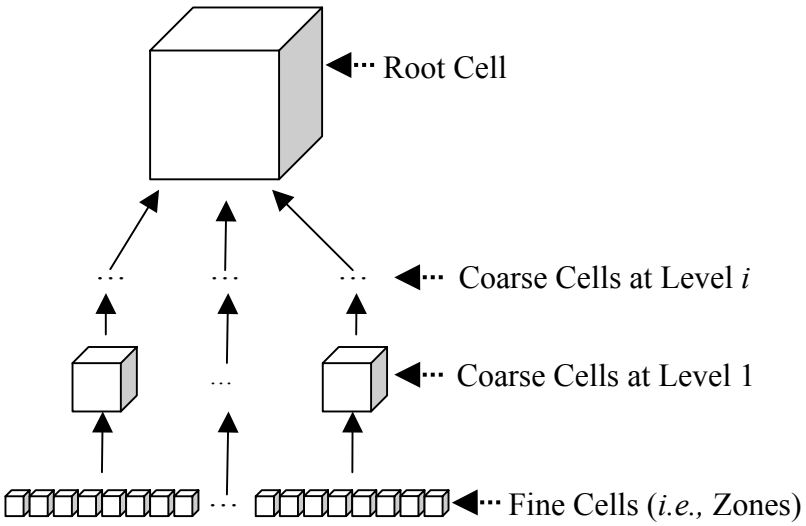


Figure 6. AQSim's bottom-up algorithm generates a *topology tree*.

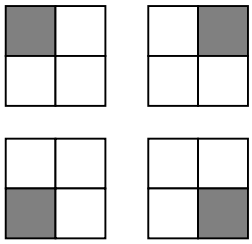


Figure 7. Four possible locations for a cell within a coarse agglomeration in two dimensions

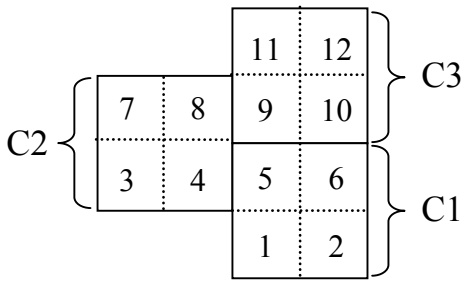


Figure 8. A non-quad tree coarse cell arrangement

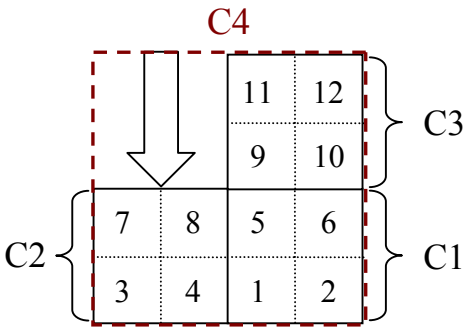


Figure 9. A fix for a non-quad tree coarse cell arrangements

Table 3. AQSim's Multivariate Clustering Algorithm

Inputs

- List of zones
- User-defined clustering threshold $u \in [0, 1]$, where 0 and 1 indicate complete dissimilarity and similarity, respectively.

Output

- A list of clusters

Assumptions

- A zone is either **GREEN** (*i.e.,* available for clustering) or **RED** (*i.e.*, already included in a cluster).
- Initially, the colors of all zones are **GREEN**.

Algorithm

For each zone, *z*, do

If (*z.color* ≡ **GREEN**) then

(a) *C* = new Cluster();

(b) Add *z* to *C*;

(c) *C.stats* = *z.stats*; /* *z.stats* contains *z*'s $\vec{\mu}$, $\vec{\sigma}$, $\vec{max}$, and $\vec{min}$. */

(d) *z.color* = **RED**;

(e) For each zone, *z*′, do

If (*z*′*.color* ≡ **GREEN**) and (*CosSim*(*z, z*′) ≥ *f*(*u*)) then

1. Add *z*′ to *C*;
2. Update *C.stats*;
3. *z*′*.color* = **RED**;

(f) Add *C* to the list of clusters;

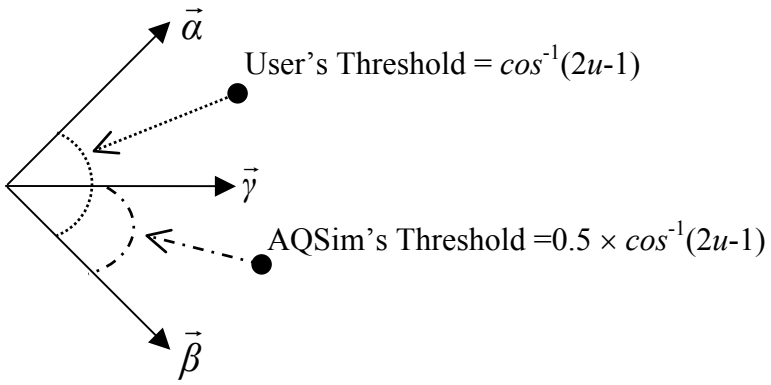


Figure 10. User-defined threshold and AQSim's threshold

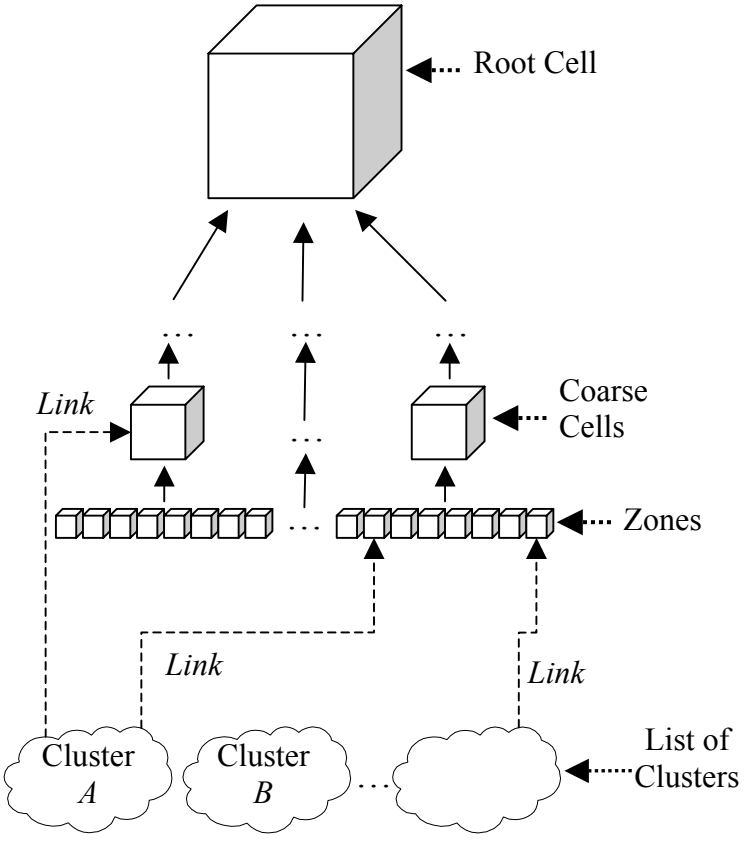


Figure 11. Links between list of clusters and topology tree

Table 4. AQSim's Linking Algorithm

Input

- *links* = list of links for cluster, *C*
- *z* = zone being added to *C*

Output

- Update *links*

Algorithm

If (*links* is not full) then

- *z.percentage_intersection* = 100;
- Add *z* to *links*;
- Return *links*;

Else

- For *l* = 0 (leaf level) to *r* (root level), do

  /* *z.ancestor*[*l*] is *z*'s ancestor at *l*. */

  If (*z.ancestor*[*l*] is in *links*) then

  (a) Update *z.ancestor*[*l*].*percentage_intersection*;

  (b) Return *links*;

- *new_link* = NULL;
- For each link, *j*, in *links*, do

  /* *best_ancestor*[*j*] is *j*'s ancestor with the most number of descendents in *links*. */

  If (*best_ancestor*[*j*].*num_descendents* ≡ MAX_NUM_DESCENDENTS) then

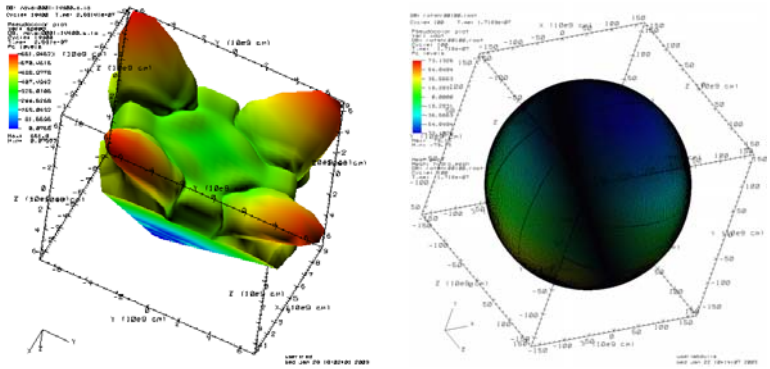
  (a) *new_link* = *best_ancestor*[*j*];

  (b) break;

- If (*new_link* ≡ NULL) then *new_link* = $\text{argmax}_{j \in links}$ *best_ancestor*[*j*];
- Clear all descendents of *new_link* from *links*;
- Set *new_link.percentage_intersection*;
- Add *new_link* to *links*;
- *z.percentage_intersection* = 100;
- Add *z* to *links*;
- Return *links*;

Table 5. Characteristics of our two astrophysics data sets

Data Set	Size (in GB)	# of Zones	# of Variables	# of Time Steps
White Dwarf	3	557,375	20	22
Djehuty-5	5	1,625,000	18	16



(a) **(b)**

Figure 12. (a) The White-Dwarf data set and (b) Djehuty-5 data set

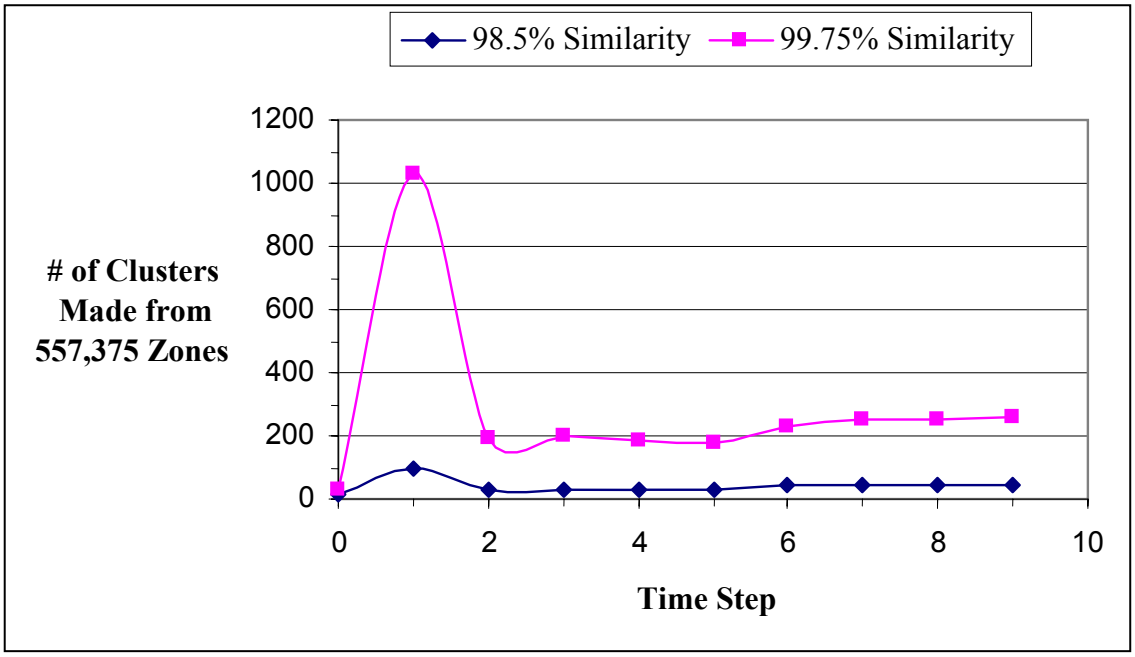


Figure 13. Number of clusters made for White Dwarf with AQSim's threshold of 98.5% & 99.75%

Table 6. Clustering on White Dwarf (*user_threshold* = 95% and *f*(*u*) = 98.5%)

	Min	Avg	Max
Number of Clusters Made from 557,375 Zones	13	39.14	98
Execution time without Linking in Seconds	78.48	138.94	204.79

Table 7. Clustering on White Dwarf (*user_threshold* = 99% and *f*(*u*) = 99.75%)

	Min	Avg	Max
Number of Clusters Made from 557,375 Zones	28	281.3	1029
Execution time without Linking in Seconds	94.10	1568.62	4727.93

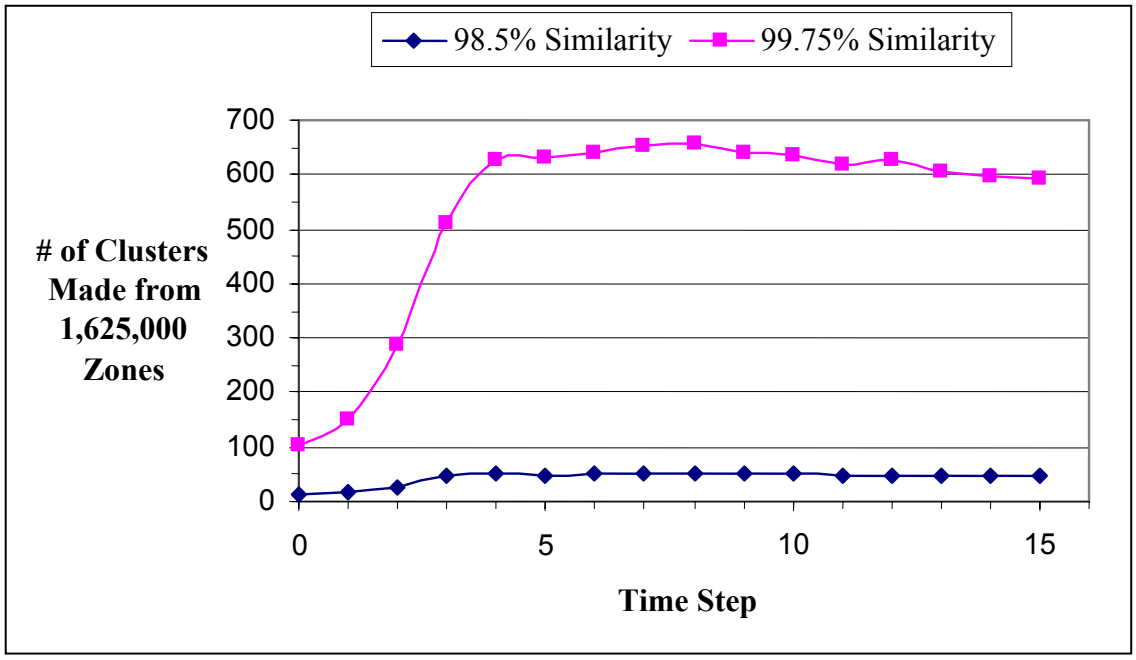


Figure 14. Number of clusters made for Djehuty-5 with AQSim's threshold of 98.5% & 99.75%

Table 8. Clustering on Djehuty-5 (*user_threshold* = 95% and *f*(*u*) = 98.5%)

	Min	Avg	Max
Number of Clusters Made from 1,625,000 Zones	14	43.56	53
Execution time without Linking in Seconds	519.78	579.40	800.39

Table 9. Clustering on Djehuty-5 (*user_threshold* = 99% and *f*(*u*) = 99.75%)

	Min	Avg	Max
Number of Clusters Made from 1,625,000 Zones	101	536.20	657
Execution time without Linking in Seconds	939.44	4787.21	11319.7

Table 10. Clustering on White Dwarf (*user_threshold* = 99% and *f*(*u*) = 99.75%)

Time Step	Execution Time Without Linking in Seconds	Execution Time With Linking in Seconds
0	94.10	867.24
1	1259.80	1489.42
2	622.99	763.51
3	361.91	712.41
4	323.89	672.48
5	384.50	753.06
Max Level Reached in Topology Tree	Max Level Reached with Linking	Min and Max % intersection at Max Level
11	4	Min = 25% and Max = 100%

Table 11. Clustering on Djehuty-5 (*user_threshold* = 99% and $f(u) = 99.75\%$)

Time Step	Execution Time Without Linking in Seconds	Execution Time With Linking in Seconds
0	939.44	2467.04
1	1148.61	2569.44
2	1650.27	2926.92
3	2288.11	3524.46
4	3382.61	4828.16
5	4047.15	5597.04
Max Level in Topology Tree	Max Level Reached with Linking	Min and Max % intersection at Max Level
11	5	Min = 12.5% and Max = 100%

Table 12. Execution Times for Our Multivariate Clustering Algorithm (without Linking) and Average Value for $g(f(u))$

Data Set	User Threshold	Avg $O(n)$ in Seconds	Avg $O(n \times g(f(u)))$ in Seconds	Avg $g(f(u))$
White Dwarf	95%	232.32	732.09	3.15
White Dwarf	99%	281.33	2179.6	7.75
Djehuty-5	95%	351.22	1249.6	3.56
Djehuty-5	99%	763.91	1587.1	2.08

Table 13. Average Distortion within Clusters

Data Set	User Threshold	Average d in $(\vec{\mu}_c - d \times \vec{\sigma}_c) = \vec{min}_c$ and $\vec{max}_c = (\vec{\mu}_c + d \times \vec{\sigma}_c)$
White Dwarf	95%	2.96
White Dwarf	99%	2.39
Djehuty-5	95%	2.79
Djehuty-5	99%	2.48

Table 14. White Dwarf Data Set: Comparison of multivariate clusterer (with linking) versus the bottom-up agglomeration (with mean modeler)

Data Set = White Dwarf Time Step = 0 User Threshold = 99%	Execution Time in Seconds	Number of Agglomerations Made from 557,375 Zones
Bottom-up Agglomeration Algorithm with Mean Modeler	753.44	73924
Multivariate Clusterer with Linking	867.24	28

Table 15. Djehuty-5 Data Set: Comparison of multivariate clusterer (with linking) versus the bottom-up agglomeration (with mean modeler)

Data Set = Djehuty-5 Time Step = 0 User Threshold = 99%	Execution Time in Seconds	Number of Agglomerations Made from 1,625,000 Zones
Bottom-up Agglomeration with Mean Modeler	946.57	203125
Multivariate Clusterer with Linking	2467.04	101