Clustering Technology of a Data Engine for Analytical Computing

Ratko Orlandic
Department of Computer Science
Illinois Institute of Technology
Chicago, IL 60616
Email: ratko@cs.iit.edu

Ying Lai
Department of Computer Science
Illinois Institute of Technology
Chicago, IL 60616
Email: laiying@iit.edu

Abstract— **Contemporary scientific studies frequently rely on data-intensive analytical computing. While the main goal of this emerging form of computing is to facilitate hypothesis formulation or to test the validity of a postulated model, its primary method is usually that of data clustering. Since typical analytical tasks operate on very large volumes of potentially high-dimensional data, scientific studies also face enormous problems of scale. This paper describes the clustering technology of a new engine for data-intense analytical computing. The technology is designed to operate in high-dimensional feature spaces without requiring dimensionality reduction. This enables the data engine to achieve high degrees of scalability and high interoperability between the analytical tasks. Most processes supported by the engine operate on a shared aggregate representation of data in the original feature space.**

Keywords: scientific databases, data mining, data clustering, data dimensionality.

I. Introduction

Since typical scienti c studies are conducted on large volumes of potentially high-dimensional data, the data-mining facilities used in these studies must cope with enormous problems of scale. For example, due to the high dimensionality of the high-energy physics (HEP) data [13], the indexing techniques appropriate for these environments must be radically different from traditional multi-dimensional access methods. Ensuring that large volumes of HEP data can be clustered both accurately and ef ciently is yet another problem that has eluded researchers thus far.

In a typical representation of scienti c data, each data item (*raw data*) is depicted by a potentially large number d of descriptive attributes represented as numeric or enumerated types [13]. This *summary information* is interpreted as a feature vector (point) in a d-dimensional space. A typical scienti c experiment begins by retrieving from a large data repository all items that satisfy a carefully constructed search predicate, which usually involves range speci cations over a subset of the intrinsic properties of summary data. Later, the selected set grows through incremental updates by requesting new items that satisfy the predicate. Several ongoing research efforts deal with the problem of retrieving data from massive data repositories [6], [8], [13]. However, the focus of this research is on data analysis, once the appropriate subset is retrieved and stored on the designated execution site.

We are currently developing a generic data engine for ef cient storage, retrieval, and analysis of complex scienti c data, called *EGALITE* (*EnGine for AnaLytIcal sTudiEs*), which tightly couples the concerns of data clustering and data retrieval. Using this engine, the scientist/analyst would rst cluster the given set of multi-dimensional data and start analyzing it by performing aggregate functions, projections of data or clusters onto different sub-dimensional spaces, assessing similarities between data items, or just browsing through subsets of particular interest. This process is usually repeated many times with different clustering parameters before the attention is turned onto a different set of data.

The engine is designed to satisfy the following main requirements: (1) *functionality*: it must support an integrated set of different retrieval and clustering techniques; (2) *scalability*: the techniques must deal effectively with large volumes of high-dimensional data; and (3) *interoperability*: the techniques must interact well in order to facilitate bundled operations of typical analytical tasks. Most analytical tasks supported by this engine operate on a shared aggregate representation of data, which facilitates the interoperability of analytical tasks.

The interoperability concerns of the engine s design are diverse. As an example, consider the subtle interdependencies between the problems of data clustering and data or cluster projection. If the clustering method employs dimensionality reduction, it is likely that even a simple operation such as the projection of clusters onto a subset of data dimensions would trigger expensive re-clustering of data. This would be the case if the principal components [3] on which the data set is clustered do not include at least one dimension on which the clusters must be projected. In contrast, if the data is clustered on all dimensions, the operation of cluster projection is greatly facilitated. It is also important to note that dimensionality reduction employs a set of static decisions, which can impede the uid and highly dynamic process of analytical thinking.

Obviously, both scalability and interoperability concerns imply a need for a clustering technology that can operate effectively and ef ciently in high-dimensional feature spaces even without dimensionality reduction. The clustering technology of EGALITE, called *GARDEN* (*GAmma Region DENsity*) clustering, is designed with this goal in mind. The focus of this paper is on this novel clustering technology. In [8], we outlined the general principles of GARDEN clustering. However, two of its constituent algorithms, called GARDEN$_{HD}$ and GARDEN$_{SP}$, are described in this paper for the rst time.

To provide high degrees of scalability to the growing dimensionalities of scienti c data, the algorithms of GARDEN clustering build on the properties of a new space-partitioning scheme, called Γ [10]. In fact, it is the application of this partitioning strategy that enables GARDEN clustering to avoid the need for dimensionality reduction.

In the rest of this paper, Section 2 gives a schematic design of EGALITE and describes the aggregate representation of data. Sections 3 and 4 describe the operation of GARDEN$_{HD}$ and GARDEN$_{SP}$, respectively. Section 5 summarizes the paper and discusses the ongoing development of the data engine.

II. Design of the Data Engine

Figure 1 shows the main components of EGALITE, which include: an ef cient and scalable indexing technique for data in high-dimensional spaces, called Γ_{SLK} [9]; a new access method for similarity searching in multi-dimensional spaces, called Γ_{NN} [9]; as well as GARDEN$_{HD}$ and GARDEN$_{SP}$ described later in this paper.

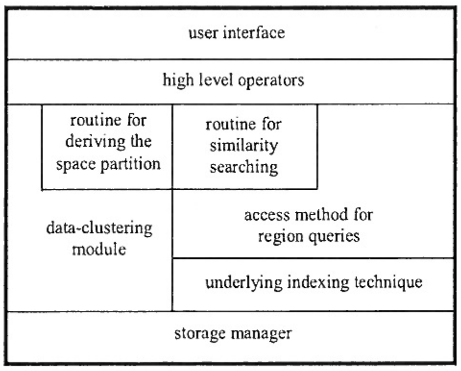
user interface
high level operators
routine for deriving the space partition
routine for similarity searching
data-clustering module
access method for region queries
underlying indexing technique
storage manager

Fig. 1. Schematic design of the data engine.

The engine is designed to maximize code reuse. In particular, the facility for similarity searching (Γ_{NN}) runs on top of the access method for region queries (Γ_{SLK}), which is implemented by reusing a more traditional indexing technique [9]. Other than for clustering data, the GARDEN$_{HD}$ clustering technique is also used by the process of deriving an appropriate representation of data.

For each data set, the engine maintains an overloaded *aggregate representation*, which is based on a new partitioning strategy for multi-dimensional spaces, called Γ [8], [10]. The advantages of Γ over traditional space-partitioning schemes are especially pronounced in high-dimensional situations. For example, in order to split each axis of the given d-dimensional space at least once, a typical grid-like space partition would create at least 2^d different regions. In contrast, the Γ strategy can split each axis multiple times while creating only O(d) regions. While each point in the space has a neighborhood that is exponential in the number of dimensions, each region in a Γ space partition has only O(d) neighbors. Therefore, the Γ partitioning strategy effectively attacks certain kinds of exponential explosion associated with more traditional space-partitioning schemes. It is this effect of Γ that allows GARDEN clustering technology to avoid the need for dimensionality reduction.

In a Γ space partition, the d-dimensional universe is statically partitioned by several nested hyper-rectangles, which we also call *generators*. The space inside one and outside its immediately enclosed generator de nes one Γ subspace. Except for the innermost subspace, every Γ subspace is further divided into d rectangular Γ *regions*, by means of d-1 hyperplanes, each lying on an outer side of its inner generator. With m generators, there are up to $1 + (m-1) \cdot d$ different Γ regions in the space [10].

Each Γ region can be further partitioned along different dimensions into several hyper-rectangular *slices*. With an additional measure, Γ space partitions can also eliminate potentially signi cant amounts of dead (empty) space associated with skewed data distributions. Toward this end, for each Γ region or slice, it is appropriate to maintain dynamically its *live region*, i.e. the minimum bounding hyper-rectangle enclosing all points of that region or slice.

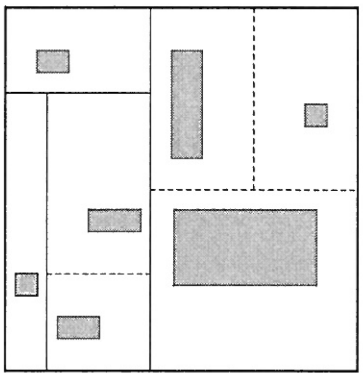

Fig. 2. Aggregate representation of a 2-dimensional data set.

In EGALITE, the aggregate representation of data is a main-memory structure used to support region queries, similarity searches, and other analytical tasks. Since the representation also provides an approximate description of the data clusters, it can be augmented with summary information about the clusters in order to facilitate aggregate operators (hence, its name). As illustrated in Figure 2, this structure also represents a static Γ space partition with possible slicing of the Γ regions. In the gure, solid lines separate Γ regions, whereas dashed lines separate slices of the Γ regions.

Due to the process of deriving the aggregate representation, which employs GARDEN$_{HD}$ and GARDEN$_{SP}$ techniques, every Γ region or slice corresponds to a cluster of data. However, certain large or irregularly shaped clusters could be broken into two or more slices. The live portions of the Γ regions or slices (dark rectangles in Figure 2) approximate the locations of the corresponding data clusters.

For each Γ region or slice, the aggregate representation maintains: its live portion; its cardinality (the number of points in the region or slice); its cluster representative (e.g., a data point lying close to the middle of the live region), which is used to facilitate similarity searching; and possibly some other parameters, e.g. the descriptive scalar values required for ef cient processing of typical aggregate operators.

The basic processes of EGALITE include initial and incremental loading of data, retrieving data based on either similarities or multi-dimensional selections, as well as data and cluster projections. The process of *initial data loading* starts by constructing the aggregate representation for the given data set. Then the summary parts of data are inserted into a primary index structure, which is based on Γ_{SLK} [9]. While Γ_{SLK} supports ef cient *region queries* in multi-dimensional spaces, many analytical tasks require a specialized access method for *similarity searching*. Our Γ_{NN} for similarity (*k*-nearest neighbor) searching [9] operates on the aggregate representation of the clustered data set maintained by the underlying Γ_{SLK} index.

The process of *cluster projection* involves two steps. In the rst step, the live regions in the aggregate representation of data are simply projected onto the desired subset of dimensions. In the second step, the process performs merging of live regions whose projections in the targeted space overlap.

The process of *data projection* rst performs cluster projection onto the targeted sub-dimensional space. Based on the knowledge of data clusters in the projected space, it applies GARDEN$_{SP}$ to select the aggregate representation for a new Γ_{SLK} index. Following that, the projected summary items are inserted into the new index. Each such insertion must dynamically update the live portion, the cardinality, and the cluster representative of the corresponding region or slice in the aggregate representation of the projected data.

III. GARDEN$_{HD}$ Clustering Technique

It is well known that contemporary clustering algorithms [2], [5] do not scale well with the increasing volumes and dimensionalities of data. Typical limitations of these algorithms are some implicit restrictions on the data-set size [15] or that they require *a priori* knowledge about the clusters [5]. The popular cell-based clustering techniques [11], [14] generally apply grid-like space partitions into rectangular cells whose number grows exponentially with data dimensionality. As a result, in high-dimensional spaces, these algorithms require some form of dimensionality reduction. While dimensionality reduction is useful in many situations, it often results in a loss of clusters as well as the distortion of both the spatial properties and densities of clusters. Other problems associated with it are discussed in the summary of this paper.

Our GARDEN$_{HD}$ clustering technique falls in the general class of cell-based and density-based clustering methods [1], [4], [11], [12]. It is appropriate in low-dimensional situations, but its main advantage is that it can operate in high-dimensional spaces even without dimensionality reduction [7]. The subscript of its name emphasizes this fact. The operation of the technique is governed by two basic parameters: δ, the

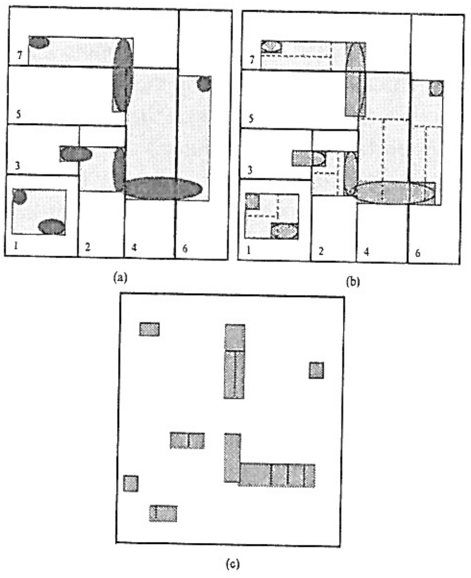

Fig. 3. Illustration of the GARDEN$_{HD}$ operation.

high-density threshold for dense regions, and , the *minimum distance* along a single dimension between two distinct clusters.

GARDEN$_{HD}$ applies a recursive partition of sparse regions in the space using a variant of the Γ partitioning strategy [7]. In fact, it is the application of the Γ strategy that enables GARDEN$_{HD}$ to avoid the need for dimensionality reduction. This is because Γ can split every axis of the feature space within linear space complexity. This, in turn, means that its recursive application will eventually detect the separating gaps between any two dense clusters. Since each Γ region has a relatively small neighborhood, merging of dense regions is also fast. As result, GARDEN$_{HD}$ can operate ef ciently in high-dimensional situations, detecting the clusters of virtually any shapes or spatial orientations. With appropriately chosen parameters δ and , this algorithm runs in virtually guaranteed O($n$log$n$) time [7].

Figure 3 illustrates the basic operation of GARDEN$_{HD}$, which is performed in three steps. The *rst step* selects the initial Γ partition of the given *d*-dimensional space and inserts in it all data items, computing the live portions of each Γ region. The result is illustrated in Figure 3a, where the dark shapes consisting of one or more ovals represent different clusters of data. While the initial Γ partition can be derived using an arbitrary number *m* of nested hyper-rectangles, which can be chosen in proportion to the number of points in the given data set, subsequent partitions of live regions are produced using a single inner generator.

The *second step*, illustrated in Figure 3b, applies a recursive partition of live regions using the Γ partitioning strategy. As shown in the gure, sparse live regions whose density is below δ (in the gure, those are live portions of the Γ regions 1, 2, 4, 6 and 7) must be recursively partitioned until all their dense sub-regions are identi ed. Each time a region is split, its points are re-examined in order to compute the live portions of the sub-regions resulting from the split. Normally, when the density of a live region exceeds the high-density threshold (see the live portions of the regions 3 and 5 in Figure 3b), the live region is immediately included into a temporary set of clusters. However, as we will see, for highest accuracy, even some dense live regions may need to be split further.

In the second step, the algorithm must decide how to split a live region that needs to be partitioned. The problem is that a live region enclosing two or more smaller clusters may become so elongated along certain dimensions that its partition using the original Γ strategy [10] may not be bene cial. In particular, since the original Γ strategy may create narrow regions along the upper boundaries of the base region that is being split, some regions may be so narrow that they could become dense even if they have just a handful of points. Such live regions may erroneously bridge two or more distant clusters, making them arti cially a single cluster.

Because of this issue, GARDEN$_{HD}$ performs the partition of a live region *LR* taking into account its properties. First, it orders the sides of *LR* according to their extension from the longest to the shortest side. Then, it computes the Γ sub-regions following the given ordering of the dimensions, which improves the squareness of the resulting sub-regions. Further improvements in this regard are obtained by not splitting the sides of *LR* that are much shorter than its longest side [7].

The *third step* of GARDEN$_{HD}$ performs merging of adjacent dense regions (dense cells) into larger clusters. While other options are possible, in the present version of GARDEN$_{HD}$, merging of two regions is performed when their distance along every dimension is below . The process of merging takes place within the invocation of each instance of recursive space partition [7]. The result of GARDEN$_{HD}$, illustrated in Figure 3c, is a *cluster representation* of data, consisting of the dense cells detected in the process of recursive space division and organized according to the larger clusters they belong to. Since the detected clusters are just unions of adjacent dense cells, they can be arbitrarily shaped.

This basic GARDEN$_{HD}$ method works well in most situations. However, for highest accuracy and best performance, additional measures may also be useful. For example, using a parameter κ to control the minimum number of points in a dense cell, the algorithm can prevent the creation of many small dense cells. This would not only speed up the process of merging, but also provide a simple way of handling possible noise in the data. Other enhancements of the basic GARDEN$_{HD}$ operation include an ef cient way of computing live regions, an ef cient data structure for storing and accessing points, and an ef cient way of merging dense cells [7].

GARDEN$_{HD}$ also has a provision for splitting dense live regions under certain conditions [7]. The need for this arises when clusters have widely varying densities. Since δ represents the density of the sparsest region that constitutes a cluster, in the course of the recursive space partition, a cluster with density much higher than δ could make a much wider area than the one it actually occupies look like a dense region. In order to depict such clusters more accurately, it is sometimes advantageous to split even some dense live regions.

Toward this end, we adopted the following heuristics [7]. First, each live region *LR* is associated with its *centroid*, computed as the center of all points falling in *LR*. If *LR* is a dense region, a simple (centroid) test is usually suf cient to determine if *LR* needs to be split further: a) compute *C*, the center of *LR*; b) compute *R*, a rectangle centered in *C* whose each side has extension of the corresponding side of *LR* multiplied by a constant $\phi < 1$ (e.g., 1/2); c) if the centroid of *LR* falls outside *R*, then *LR* must be split further.

IV. Deriving the Aggregate Representation

This section gives a high-level description of the process of selecting the appropriate Γ space partition for the aggregate representation of data. The algorithm underlying this process, called GARDEN$_{SP}$ (the subscript reads space partitioning), takes as its input the cluster representation of data produced by GARDEN$_{HD}$. In turn, GARDEN$_{SP}$ tries to produce a Γ space partition, possibly with region slicing, in which each cluster is assigned to a single Γ region or slice.

GARDEN$_{SP}$ also operates in three steps. The *rst step* produces an intermediate representation Ω. It begins by computing the minimum bounding hyper-rectangle of each cluster (*cluster MBR*). Following that, the process selects an *overlay partition* of the given *d*-dimensional space by slicing each dimension into *q* equal regions, producing exactly $q \cdot d$ overlapping regions. In contrast to the regions of a grid-like space partition, each region produced along an axis *i* of the overlay partition has full extension along all other dimensions. Thereby, it overlaps all regions created by dividing other dimensions. As illustrated in Figure 4a, regions along any given dimension are numbered from the high to low coordinates.

The overlay partition is used to derive the representation Ω, called the *overlay representation*. Each element $\Omega[i,j]$, representing a region $R_{i,j}$ of the overlay partition, maintains a triple $< I, L, H >$, where: *I* is the *intersect set* containing the identi ers of all clusters whose MBRs intersect $R_{i,j}$ (whose *i*-th sides fully enclose the corresponding side of $R_{i,j}$); *L* is the *low-endpoint set* containing the identi ers of all clusters whose MBRs have low endpoints in $R_{i,j}$; and *H* is the *high-endpoint set* containing the identi ers of all clusters whose MBRs have high endpoints in $R_{i,j}$.

The *second step*, illustrated in Figures 4a-c, operates on the elements of the representation Ω, alternating the dimensions in their pre-determined order. In Figure 4, we alternate rst the horizontal and then the vertical axis of the given 2-dimensional space. While examining the elements along an axis *i*, the algorithm simulates the process of line sweeping from the high toward the low end of that dimension. The main objective of this process is to extrapolate the regions of the nal Γ space partition, so that each Γ region contains as few full cluster MBRs as possible. However, the data distribution may be such that this objective cannot be fully achieved.

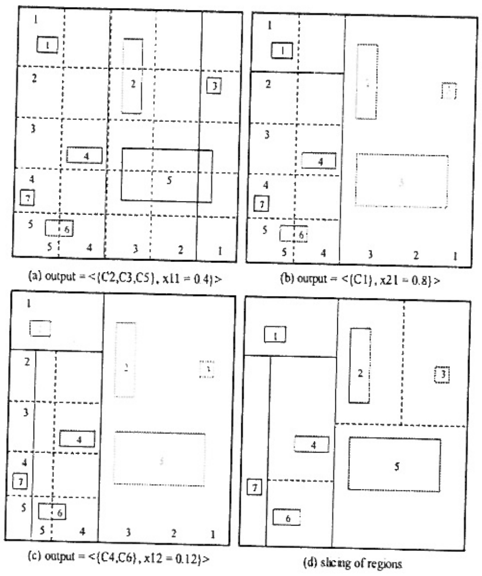
(a) output = <{C2,C3,C5}, x11 = 0.4}>
(b) output = <{C1}, x21 = 0.8}>
(c) output = <{C4,C6}, x12 = 0.12}>
(d) slicing of regions

Fig. 4. Illustration of the GARDEN$_{SP}$ operation.

The process of line sweeping along any given axis i usually proceeds until one nds a coordinate x_i where one can draw a *dividing hyper-plane* perpendicular to the axis i that does not intersect any cluster MBR. Obviously, if either the low-endpoint set L of the given element $\Omega[i, j]$ is empty or the intersect set I is non-empty, one can skip that element and go to the element $\Omega[i, j+1]$. However, if the low-endpoint set L of $\Omega[i, j]$ is not empty, but both its intersect set I and its high-endpoint set H are, then one can set x_i to the low coordinate along the axis i of the corresponding region $R_{i,j}$ and terminate the process of line sweeping along that dimension. Somewhat more complex is the case when I is empty, but both L and H are not. In this case, one must examine the low and high coordinates along the axis i of all cluster MBRs in L and H to see if there is a dividing plane between them. If so, the desired point x_i is found; otherwise, the algorithm must examine the element $\Omega[i, j+1]$ of the overlay representation.

Due to the shape and size of actual clusters, it is possible that no appropriate dividing plane along some axis i can be found. Further complicating the matter is the requirement to have as few clusters assigned to each Γ region of the space as possible. In order to address these problems, the above process of line sweeping must be modi ed to occasionally break a monolithic cluster MBR across two or more Γ regions.

Whenever a dividing plane at a certain coordinate x_i is found, GARDEN$_{SP}$ sets aside the clusters lying above x_i along the axis i and the value of x_i. In order to prepare for the line sweeping along the next dimension $(i+1)$ *mod d*, the clusters in the output of the current iteration are removed from all elements of the representation Ω in which they appear. This iterative process terminates when the last group of clusters is assigned to its own region. At that time, the regions of the induced Γ space partition can be constructed from the values x_i identi ed in the process.

Assuming that the 2-dimensional space of Figure 4 is a normalized universe $[0,1]^2$, the line sweeping along the horizontal axis (dimension 1) encounters a dividing line at the coordinate x_{11} = 0.4. Since the clusters C2, C3 and C5 lie above x_{11} along this dimension, they are included in the output and removed from the overlay representation. In Figure 4b, the three clusters appear dimmed.

The next iteration of line sweeping, shown in Figure 4b, proceeds along the vertical axis (dimension 2). Based on the rules of line sweeping, the process will identify a dividing line at the point x_{11} = 0.8. The cluster C1, which lies above the coordinate x_{11} along the vertical axis, is included in the output and removed from the overlay representation. The nal iteration, shown in Figure 4c, switches back to the rst dimension, identifying a dividing line at x_{11} = 0.12 and the clusters C4 and C6 lying above this point.

After removing the clusters C4 and C6 from the representation Ω, the only remaining cluster C7 is assigned to the remaining portion of the space. The resulting space partition of this example has four Γ regions. Aside from the outer generator whose high endpoint is $< 1, 1 >$, this process has identi ed two other nested generators with high endpoints $< 0.4, 0.8 >$ and $< 0.12, 0.8 >$. In accord to the Γ partitioning strategy, the low endpoint of each of these generators lies in the origin of the space.

The *third step* of GARDEN$_{SP}$, which is illustrated in Figure 4d, performs slicing of all Γ regions in the space partition that contain more than one cluster. The goal here is to assign each cluster of every such Γ region to its own rectangular slice. Toward this end, all cluster MBRs within the given region are pair-wise compared to determine the planes that separate their respective slices. This process must take into account that, unlike actual clusters, their MBRs could overlap. Whenever this happens, the process must examine the actual clusters and break them across multiple slices in such a way that the cluster MBRs assigned to each slice do not overlap. When the shapes of these clusters complicate this slicing, their overlapping MBRs could be assigned to a single slice.

V. Summary and Discussion

In this paper, we have described the clustering technology of a new data engine for analytical computing, which can ef ciently operate in high-dimensional spaces with or without dimensionality reduction. Dimensionality reduction is natural in many situations, in part because clusters may appear only in a sub-dimensional space and additional dimensions may only disperse them. Since different dimensions often convey different degrees of information, in many situations, eliminating certain dimensions leads to only a marginal loss of information [3]. Another factor that often motivates dimensionality reduction is the ef ciency of data clustering.

However, dimensionality reduction has many disadvantages as well. Those that are well documented include loss of certain clusters as well as the distortions of the spatial properties

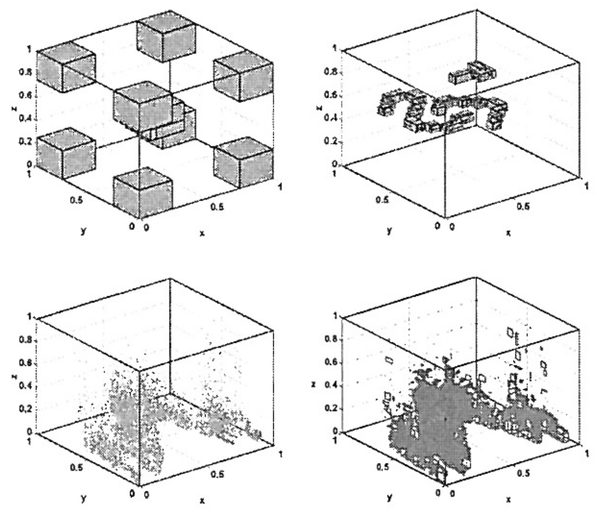

Fig. 5. Visually observable accuracy of GARDEN$_{HD}$ for synthetic and real data in a 3D space.

and densities of clusters. Moreover, if data is clustered using dimensionality reduction, an operation such as project clusters onto a sub-dimensional space may trigger re-clustering of data, which would not be the case when data is fully clustered in the original space.

In addition, when a suf ciently large pool of fully-speci ed data is clustered in the original space, effective classi cation of data with missing information becomes possible. For any data item with missing values, it is possible to have an effective probabilistic ranking of clusters that are likely to contain this item. Based on this ranking, one can also speculate, with fair degree of certainty, what the possible ranges of missing values are. When data is clustered using dimensionality reduction, this capability is generally lost.

The data engine is currently under development. However, some of its techniques have already been implemented as well as fully or partially studied. In particular, numerous experiments with both simulated and real data sets were conducted to assess the performance of Γ_{SLK} against several popular multi-dimensional access methods [9]. For real data and relatively small queries, Γ_{SLK} can generate up to three orders of magnitude fewer page accesses than the latter indexing techniques [9].

The experiments with GARDEN$_{HD}$ [7] show remarkable effectiveness and ef ciency of this clustering algorithm [7]. The rst row of Figure 5 shows the results of GARDEN$_{HD}$ for two synthetic data distributions, each with 100,000 points, in a 3-dimensional space. In each of the two pictures, one can observe both the distributions of points as well as the front edges of dense cells detected in the course of the recursive space partition. The left picture of the second row shows the distribution of 3-dimensional vectors derived from a set of real multi-dimensional data extrapolated from an environmental database. The displayed points are the projections of the data set normalized to t the $[0, 1]^3$ coordinate system. The corresponding picture on the right shows the result of GARDEN$_{HD}$ for this distribution (dense cells obtained in the courses of recursive space partition). Due to many small dense cells that were produced, the results appear dark in the pictures. For several data distributions, the effectiveness of GARDEN$_{HD}$ was also tested in spaces with up to 40 dimensions. In each space and every scenario, the technique detected the exact number of clusters and matched the shape of each cluster correctly [7].

ACKNOWLEDGMENT

This material is based upon work supported by the National Science Foundation under Grant No. IIS-0312266.

REFERENCES

[1] R. Agrawal, J. Gehrke, D. Gunopulos and P. Raghavan, Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications, *Proc. ACM SIGMOD Int. Conf. on Management of Data*, 94–105, 1998.

[2] M. Ester, H.-P. Kriegel, J. Sander and X. Xu, A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise, *Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining KDD 96*, 226—231, 1996.

[3] C. Faloutsos and K. Lin, Fastmap: A Fast Algorithm for Indexing, Data Mining and Visualization of Traditional and Multimedia Databases, *Proc. ACM SIGMOD Int. Conf. on Management of Data*, 163–174, 1995.

[4] A. Hinneburg and D.A. Keim, Optimal Grid-Clustering: Towards Breaking the Curse of Dimensionality in High-Dimensional Clustering, *Proc. 25th Int. Conf. on Very Large Data Bases*, 506–517, 1999.

[5] A.K. Jain, M.N. Murthy and P.J. Flynn, Data Clustering: A Review, *ACM Computing Surveys* 31(3):264–323, 1999.

[6] T. Kurc, U. Catalyurek, C. Chang, A. Sussman and J. Saltz, Visualization of Large Datasets with the Active Data Repository, *IEEE Computer Graphics and Applications* 21(4):24–33, 2001.

[7] Y. Lai and R. Orlandic, GARDEN$_{HD}$: Clustering High-Dimensional Data in Original Feature Spaces, Technical Report, Department of Computer Science, Illinois Institute of Technology, 2004.

[8] R. Orlandic, Effective Management of Hierarchical Storage Using Two Levels of Data Clustering, Proc. 20th IEEE Conf. on Mass Storage Systems and Technologies, 270–279, 2003.

[9] R. Orlandic and J. Lukaszuk, An Effective Framework for Scalable Retrieval of Multi-Dimensional Data, Technical Report, Department of Computer Science, Illinois Institute of Technology, 2004.

[10] R. Orlandic, J. Lukaszuk and C. Swietlik, The Design of a Retrieval Technique for High-Dimensional Data on Tertiary Storage, *SIGMOD Record* 31(2):15–21, 2002.

[11] E.J. Otoo, A. Shoshani and S. Hwasng, Clustering High Dimensional Massive Scienti c Datasets, *Proc. 13th Int. Conf. on Scienti c and Statistical Database Management SSDBM 01*, 147–157, 2001.

[12] G. Sheikholeslami, S. Chatterjee and A. Zhang, WaveCluster: A Multi-resolution Clustering Approach for Very Large Spatial Databases, *Proc. 24th Int. Conf. on Very Large Data Bases*, 428–439, 1998.

[13] A. Shoshani, L.M. Bernardo, H. Nordberg, D. Rotem and A. Sim, Multidimensional Indexing and Query Coordination for Tertiary Storage Management, *Proc. 11t Int. Conf. on Scienti c and Statistical Database Management SSDBM 99*, 214–225, 1999.

[14] W. Wang, J. Yang and R. Muntz, STING: A Statistical Information Grid Approach to Spatial Data Mining, *Proc. 23rd Int. Conf. on Very Large Data Bases*, 186–195, 1997.

[15] T. Zhang, R. Ramakrishnan and M. Livny, BIRCH: An Ef cient Data Clustering Method for Very Large Databases, *Proc. ACM SIGMOD Int. Conf. on Management of Data*, 103–114, 1996.