A Grid-based Clustering Algorithm using Adaptive Mesh Refinement [*]

Wei-keng Liao [†] Ying Liu [†] Alok Choudhary [†]

Abstract

Clustering analysis, an automatic process to find similar groups of objects from a database, has been studied for many years. With the increasing data size generated recently, clustering large databases poses a challenging task that must satisfy both the requirements of the computation efficiency and result quality. Among the existing clustering algorithms, grid-based algorithms generally have a fast processing time, which first employ a uniform grid to collect the regional statistic data and, then, perform the clustering on the grid, instead of the database directly. The performance of grid-based approach normally depends on the size of the grid which is usually much less than the database. However, for highly irregular data distributions, using a single uniform grid may not be sufficient to obtain a required clustering quality or fulfill the time requirement. In this paper, we propose a grid-based clustering algorithm using adaptive mesh refinement technique that can apply higher resolution grids to the denser regions. With the hierarchical AMR tree constructed from the multi-grain meshes, this algorithm can perform clustering at different levels of resolutions and dynamically discover nested clusters. Our experimental results also show the efficiency and effectiveness of the proposed algorithm compared to the methods using single uniform grids.

1 Introduction.

Clustering is a process to discover the groups of similar objects from a database that can help characterize the underlying data distribution. Cluster analysis has been studied extensively for many years, especially for spatial-distance based data objects. In general, the existing clustering algorithms can be classified into four categories: partitioning-based, hierarchical-based, density-based, and grid-based methods [12]. Among them, the grid-based methods have the fastest processing time that typically depends on the size of the grid instead of the data objects. These methods use a single uniform grid mesh to partition the entire problem domain into cells and the data objects located within a cell are represented by the cell using a set of statistical attributes from the objects. Clustering is, then, performed on the grid cells, instead of the database itself. Since the size of the grid is usually much less than the number of the data objects, the processing speed can be significantly improved. However, for highly irregular or concentrated data distributions, a grid mesh with very fine granularity will be required in order to sustain a certain clustering quality, for example, to discover nested clusters. As illustrated in Figure 1, it can be hard for a single resolution grid to identify or represent nested clusters. In this case, an additional mesh with higher resolution is needed. To meet a required clustering quality, using a finer grid cannot be avoided, but the size of a fine grid can easily overwhelm the number of data objects and, hence, breakdown the computational efficiency.

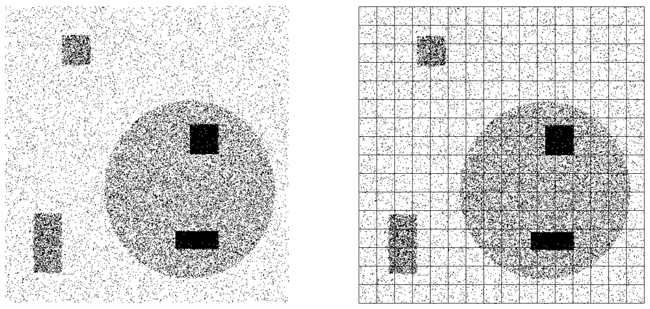


Figure 1: (a) An example of data distribution with nested clusters. (b) A single uniform mesh is applied to the spatial domain, which may not be sufficient for good quality clustering.

Adaptive Mesh Refinement (AMR) is a type of multiscale algorithm that achieves high resolution in localized regions of dynamic, multidimensional numerical simulations [4, 3]. It has been successfully applied to model large-scale scientific applications in a range of disciplines, such as computational fluid dynamics, astrophysics, meteorological simulations, structural dynamics, magnetics, and thermal dynamics. Basically, the algorithm can place very high-resolution grids precisely where the high computational cost requires. Its adaptability allows simulating multiscale resolutions that are out of reach with methods using a global uniform fine grid.

In this paper, we propose a grid-based clustering algorithm that employs the AMR technique for data mining prob-

[*]This work was supported in part by DOE Lawrence Livermore National Laboratory under subcontract No.B524122/W-7405-ENG-48, the Scientific Data Management Center of the DOE SciDAC program under Contract No. W-31-109-ENG-38, NSF No. EIA-0103023, and a gift from Intel Corporation.

[†]Department of Electrical and Computer Engineering, Northwestern University. {wkliao, yingliu, choudhar}@ece.northwestern.edu

lem with highly irregular data distributions. Instead of using a single resolution mesh grid, the AMR clustering algorithm first adaptively creates different resolution grids based on the regional density and these grids comprise a hierarchy tree that represents the problem domain as nested structured grids of increasing resolution. Secondly, the algorithm considers each leaf as the center of an individual cluster and recursively assigns the membership for the data objects located in the parent nodes until the root node is reached. The determination of the membership of a data object is based on the minimum distance to the clusters discovered underneath the tree branch. Using a hierarchical tree, the AMR clustering algorithm can detect nested clusters at different levels of resolutions. Since it is a grid-based method, it also shares the common characteristics of all grid-based methods, such as fast processing time, insensitive to the order of input data, and the ability to separate from the noise. The AMR clustering algorithm is also applicable to any collection of attributes with numerical values. Most importantly, since many large-scale applications today have been programmed using AMR techniques in their data structures, embedding AMR clustering into these applications to perform on-line data analysis can be more straightforward than other types of clustering algorithms.

Recently, data mining techniques are frequently used to help discover the patterns from the simulation outputs as in the post-simulation applications. These post-simulation applications usually are implemented by different programmers as independent applications and it is not easy to integrate to the simulation applications. In many cases, the mining process is required perform on line such that the mining results can be directly fed back to the simulation to tune the parameters and guide the subsequent simulation process. The direct feedback requires the seamless embedding of data mining process into the applications. By reusing the AMR structure already constructed in the simulation applications, our proposed AMR clustering algorithm presents a straightforward solution for on-line data analysis and a great potential for performance enhancement.

The rest of the paper is organized as follows. We discuss related work in Section 2. The concept of the AMR method is described in Section 3. The AMR clustering algorithm proposed in this work is presented in Section 4. In Section 5, we present experimental results and Section 6 discusses the sensitivity of the algorithm to the input parameters. The paper is concluded in Section 7.

2 Related Work.

Generally, clustering algorithms can be classified into four categories: partitioning-based, hierarchical-based, density-based, and grid-based methods. The representatives for partitioning-based clustering algorithms are k-means [17], k-medoid [16], and expectation maximization (EM) algorithms [8]. Both k-means and k-medoid algorithms represent a cluster using a single point, while EM uses a probability distribution to represent a cluster. The partitioning-based algorithms require user provide the parameter, k, the number of clusters, and perform iterative membership relocation until the membership is no longer changed or the change is within a tolerable range. The clustering quality highly depends on the value of k and, in general, the partitioning-based algorithms cannot find arbitrarily shaped clusters well.

Hierarchical-based clustering algorithms use a hierarchical tree to represent the closeness of the data objects. The tree is constructed in either bottom-up or top-down. The bottom-up approach starts with each object forming a cluster and recursively merges the clusters based on their closeness measure. On the contrary, the top-down approach starts with all the objects in a single cluster and recursively splits the objects into smaller groups. The representative hierarchical-based clustering algorithms are BIRCH [22], CURE [11], and CHAMELEON [15].

Density-based clustering algorithms consider clusters as dense regions of objects in the data space and clusters are separated by regions of low density. These algorithms associate each object with a density value defined by the number of its neighbor objects within a given radius. An object whose density is greater than a specified threshold is defined as a dense object and initially is formed a cluster itself. Two clusters are merged if they share a common neighbor that is also dense. The representative density-based clustering algorithms are DBSCAN [10], OPTICS [2], HOP [9], and DENCLUE [13]. These methods can separate the noise (outliers), find arbitrary shape clusters, and do not make any assumptions about the underlying data distribution. However, they are computationally very expensive, especially at the stages of generating the density and searching for the dense neighbors. More efficient approaches rely on spatial indexing, such as R* tree, X tree, and KD tree, to identify the neighbors within the given radius.

Grid-based clustering algorithms first cover the problem space domain with a uniform grid mesh. Statistical attributes are collected for all the data objects located in each individual mesh cell and clustering is, then, performed on the grid, instead of data objects themselves. These algorithms typically have a fast processing time, since they go through the data set once to compute the statistical values for the grids and the performance of clustering depends only on the size of the grids which is usually much less than the data objects. The representative grid-based clustering algorithms are STING [21], WaveCluster [19], and CLIQUE [1]. All these methods employ a uniform grid mesh to cover the whole problem. For the problems with highly irregular data distributions, the resolution of the grid mesh must be fine enough to obtain a good clustering quality. A finer mesh can result in the mesh size close to or even exceed the size of the data objects, which can significant increase the computation load for clustering.

In this work, we propose an Adaptive Mesh Refinement clustering algorithm, which is also a grid-based approach but using multiple meshes with different sizes and resolutions. The proposed approach employs low-resolution meshes for sparse regions and high-resolution meshes for denser regions. In other words, the computation is performed at the regions that need the most.

3 Adaptive Mesh Refinement.

In many applications that process large amount of spatial data, structured meshes are often used to avoid directly operating on the data objects. The meshes partition the spatial domain into cells in which data objects located within a cell's subdomain are represented by the statistical data of the objects, such as means, maximum, minimum, variance, and distribution type. An example is the particle-in-cell (PIC) algorithm [20] that is widely used to simulate plasma and hydrodynamics [14, 5]. In PIC, the movement of a particle is determined by the interaction between the particle itself and all others in a self-consistent system. Instead of directly calculating the interaction of particles, which can result in a much larger computational cost, the PIC algorithm employs a uniform mesh and assigns the particle attributes to nearby grid points of the mesh. The field equations are, then, solved on the mesh to calculate the force that moves the particles. However, a tradeoff exists on most of the applications that employ a single uniform mesh. Using a finer-grain (higher-resolution) mesh can result in a more accurate moving force but incurs higher computation cost. On the contrary, a coarse mesh reduces the computation cost but may generate unsatisfied low quality results.

Adaptive Mesh Refinement (AMR) has become popular in the field of computational physics and been used in a variety of applications in computational fluid dynamics [6, 7]. AMR is a technique that starts with a coarse uniform grid covering the entire computational volume and automatically refines certain regions by adding finer subgrids. New child grids are created from the connected parent grid cells whose attributes, density for instance, exceed given thresholds. Refinement is performed on each grid separately and recursively until all regions are captured with the desired accuracy. AMR grids comprise a hierarchy tree that represents a spatial domain as nested structured grids of increasing resolution, and provides the ability to increase resolution locally only where it is needed. Figure 2 shows an example of an AMR tree in which each tree node uses a different resolution mesh. The root grid with the coarsest granularity covers the entire domain, which contains two subgrids, grids 1 and 2. Grid 2 at level 1 also contains two subgrids that are discovered using a finer mesh. The deeper the node is located in the tree, the finer the mesh is used.

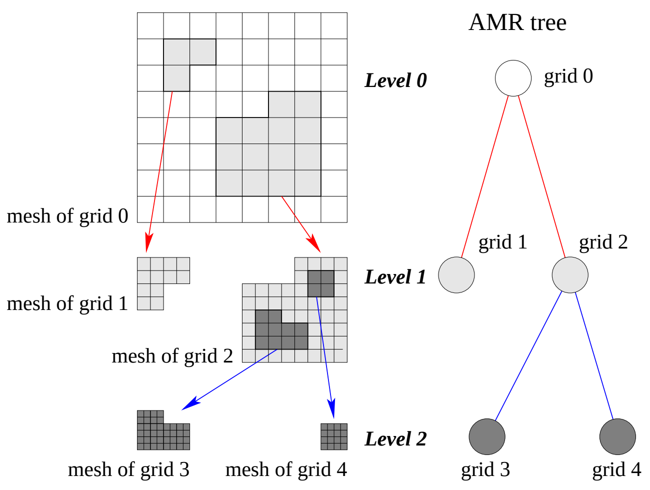


Figure 2: A 2-dimensional AMR example with 2 levels of refinement. A finer resolution mesh is applied each time a subgrid is created.

4 AMR Clustering Algorithm.

The motivation of combining the AMR concept into the clustering comes from the observation that a very fine mesh can be required for clustering on a highly irregular or concentrated data distribution if a grid-based clustering algorithm that employs a single uniform mesh is used. A fine mesh results in high computation cost and, in some cases, the mesh size can even overwhelm the number of the data objects. On the other hand, AMR's adaptability on irregular data distribution can significantly reduce the amount of the grid size and, hence, the computation cost without losing the clustering quality. In addition, the meshes in different granularities generated in the AMR tree are naturally suitable for discovering the nested clusters. Since the AMR method is also grid-based, the clustering algorithm using AMR approach shares the characteristics of all grid-based algorithms, such as fast processing time since clustering is performed on grids instead of data objects and the ability to separate from the noise.

Let us consider the density-based clustering algorithms that associate each object to its densest neighbor and objects associated to the same neighbors are merged to form a cluster. (Taking the HOP algorithm [9] for example, the association of a data object "hops" from its current densest neighbor to that neighbor's densest neighbor and this process continues until the densest neighbor of an object is itself. The objects hop to the same densest object are considered to belong to the same cluster and the densest object becomes the pivot of the cluster.) In general, the density-based algorithms try to find the dense regions and the number of the discovered dense regions determines the number of clusters. If the refinement is based on the density, the AMR method can re-

```
AMR (grid, level)
1.  create and setup the mesh for this grid
2.  for each particle contained in this grid
3.      find the mesh cell id in which the particle is located
4.      accumulate the particle to the density of cell id
5.  for each mesh cell
6.      if the density of this cell is greater than the threshold
7.          mark this cell to be refined
8.          connect this cell to its neighbor that is also marked
9.          if no such neighbor exists
10.             create a new subgrid containing this cell only
11. for each subgrid
12.     call AMR (subgrid, level+1)
```

Figure 3: The algorithm constructs the AMR tree.

cursively identify the dense regions and represent them in a hierarchical tree structure in which the tree nodes near the leaves indicate the denser regions and the nodes closed to the root have lower densities. Similar to the density-based algorithms that an object's density is determined by the number of neighbors in a given radius and objects spatially adjacent to each other have approximately the same density, the AMR method can generate grids in which the data objects located in a grid also have approximately the same density. Therefore, the leaves of the AMR tree can naturally be regarded as cluster pivots and clustering for the objects in the upper-level (parent) nodes can be based on the distance to the clusters formed in the lower levels.

Our proposed AMR clustering algorithm connects the grid-based and density-based approaches through AMR technique and, hence, preserves the advantages of both type algorithms. The algorithm consists of two stages: constructing the AMR tree and data clustering. The AMR tree construction is a top-down process starting from the root node that covers the entire problem volume. The data clustering stage is a bottom-up process which starts at a given tree level (depth) and works toward to the root. Note that, although the AMR clustering algorithm employs a hierarchical tree and clustering is performed on the tree nodes, this algorithm should not be classified as a hierarchical-based clustering algorithm. This is because clusters generated at the lower levels will not be merged when the clustering is processing toward the tree root.

4.1 Constructing AMR Tree. Figure 3 shows the algorithm of constructing an AMR tree based on the spatial density refinement. Given a database, the AMR tree construction starts at the root grid (node) that uses a mesh grid with an initial granularity (resolution) to cover the entire problem domain. The algorithm calculates the mesh cell id for each data object using its spatial coordinates and updates the density value of the cells where the objects are located. Mesh cells are, then, examined to check if the density exceeds the given threshold. The cells whose density is larger than the threshold are marked to be refined. A new subgrid is created from all marked cells that are connected (adjacent) with each other. The algorithm recursively goes to the child grids while a hierarchical tree is built. The refine factor that defines the mesh resolution ratio of a child grid to its parent is used to create finer meshes at the child grids. The algorithm stops when the maximum level of tree depth is reached or there are no mesh cells with density that is larger than the threshold. The process of constructing the AMR tree is a top-down operation. This is also the main difference of our AMR approach from the other grid-based algorithms whose hierarchical trees are built in a bottom-up fashion. In addition, the AMR method will revisit the data located in the dense regions while other grid-based algorithms read the database only once.

Assuming n is the number of the data objects, d is the dimensionality, t is the number of attributes in each dimension, h is the AMR tree height, and p represents the average percentage of data objects to be refined at each level, the complexity for scanning the database is $O(dtn + dtnp + dtnp^2 + \cdots + dtnp^{h-1}) = O(dtn\frac{1-p^h}{1-p}) \leq O(\frac{dtn}{1-p})$. When $p = 0.5$, the complexity is $O(2dtn)$. The complexity of finding the subgrids highly depends on the size of the mesh in each grid. We assume the mesh size at the root is m and q is the average ratio of mesh sizes between two levels of grids. The complexity for marking the mesh cells that exceed the threshold and connecting the marked cells to form the subgrids is $O(2^d 3^d mq + 2^d 3^d mq^2 + \cdots + 2^d 3^d mq^{h-1}) = O(6^d m\frac{1-q^h}{1-q})$, assuming the refinement factor is 2 and each cell must check its $(3^d - 1)$ neighbors for connected subgrid. If $q = 0.5$, the complexity becomes $O(2 \cdot 6^d m)$. Therefore the complexity for constructing the AMR tree is $O(dtn\frac{1-p^h}{1-p} + 6^d m\frac{1-q^h}{1-q})$.

4.2 Data Clustering. The stage of data clustering is performed on the AMR tree. The clustering starts from the leaves of the AMR tree and considers all leaves as individual clusters. Next, the clustering goes up to the parent grids of the leaves. For the parent grids that contain only one cluster (one child grid), the memberships of the data objects in these grids are automatically assigned to the only cluster. For the parent grids containing more than one cluster, the data objects are assigned based on the shortest distance from the clusters. There are many options for the membership assignment. The simplest one is to assign membership for each data object independently. For the performance considera-

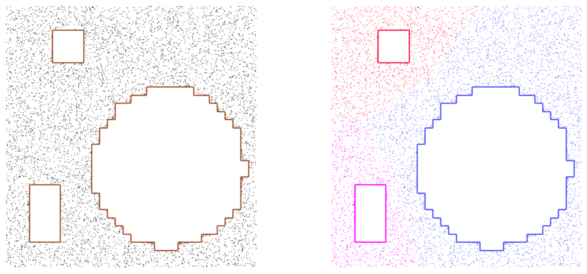


Figure 4: In root grid, the data objects are assigned to the nearest clusters based on their distance to the cluster's outlines.

tion, we can let all data objects located within the same cells have the same membership and, then, assign the membership in the unit of cells. In the latter approach, a cell can be represented by its center. There are also many options for representing a cluster. One option is to use the center of the cluster as in k-means and k-medoid clustering methods. The distance of a data object to a cluster is, then, calculated from the center of the cluster. However, using a single point to represent a cluster is known for its splitting problem that can split large clusters. This problem is commonly seen in the partition-based methods and using more than one representative points usually can improve the problem [11].

In our approach, we choose the cluster outlines to represent the cluster. We define a cluster's outlines as a list of its mesh cells that have at least one of its adjacent neighbor cells belonging to a different cluster. If the number of the outlines is too large, a fix number can be used to reduce the cost for distance computation. We, then, refer the distance of a data object to a cluster as the shortest spatial distance between the object and the cluster's outlines. Figure 4(a) shows the data objects left in the root grid after the first level of AMR tree is constructed using the example shown in Figure 1. The root grid contains 3 child grids and each child grid forms a cluster in this example. Figure 4(b) presents the clustering results of data objects in the root grid in which objects are assigned to one of the three clusters. Figure 5 describes the algorithm for the stage of data clustering. The clustering algorithm is performed bottom-up recursively and stops when it reaches the root.

Data clustering can also start at any level between the root and the maximum depth of the tree. Given a starting clustering level, λ, every internal tree node at the level λ is also considered as a cluster and all its descendant grids are regarded as part of the cluster. In other words, the tree branches with depth greater than λ are pruned and all leaves of the pruned AMR tree are considered as individual clusters. Those leaf nodes whose depths are between 0 and λ are also considered as individual clusters. Hence, the number of clusters equals to the number of leaves of the pruned AMR tree. This starting clustering level determines the computation cost of the clustering stage as well as the clustering quality.

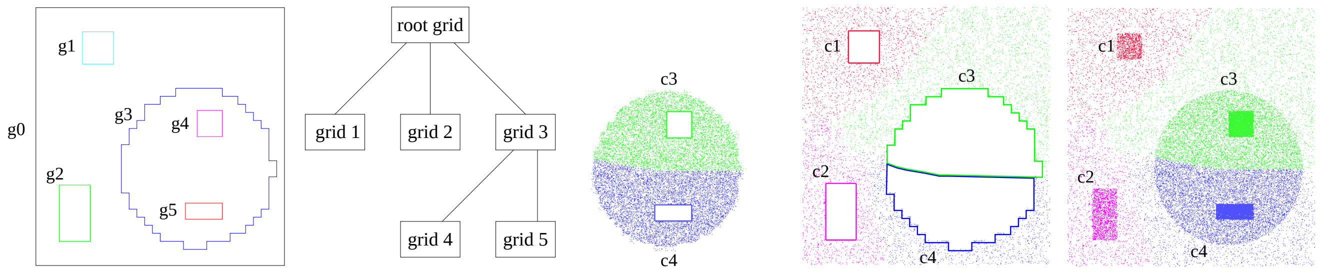
```
Clustering (grid, λ)
1.  if grid's level < λ
2.      for each subgrid
3.          call Clustering (subgrid, λ)
4.  if the number of subgrids = 0  OR  grid's level = λ
5.      create a new cluster for objects of this grid and all subgrids
6.      find the outliners of this cluster
7.      return
8.  if the number of clusters contained in this grid = 1
9.      assign all unmarked mesh cell to this only cluster
10.     generate new outliners for this cluster
11. else
12.     for each unmarked mesh cell
13.         for each cluster in this grid
14.             find the distance to the cluster's outliners
15.         assign this cell to the cluster with minimum distance
```

Figure 5: The data clustering algorithm. λ is the given tree level that clustering begins with.

Considering the example given in Figure 1, the process sequence of the data clustering stage is described in Figure 6. Figure 6(a) and (b) illustrates the grids and the AMR tree generated from our AMR clustering algorithm, respectively. Figure 6(c) shows the data object assignment inside grid 3 and this clustering result is extended upward to grid 0. The data objects in grid 0 are assigned to 4 clusters in Figure 6(d). The final clustering result of 4 clusters is shown in Figure 6(e).

The complexity of the data clustering stage depends on the options used in representing the clusters (e.g. the outlines or centers of the clusters) and the AMR tree structure.

Assuming k is the number of leaf nodes (k clusters) in the AMR tree, the mesh size at the root is m, and q is the average ratio of mesh sizes between two consecutive levels of grids, the complexity for clustering the mesh cells inside the grids based on the distance to the cluster centers is $O(dtkmq + dtkmq^2 + \cdots + dtkmq^{h-1}) = O(dtkm\frac{1-q^h}{1-q})$, where d is the dimensionality, t is the number of attributes in each dimension, and h is the AMR tree height. When $q = 0.5$, the complexity becomes $O(2dtkm)$. Combining with the complexity of the constructing AMR tree stage, the complexity of the AMR clustering algorithm is $O(dtn\frac{1-p^h}{1-p} + (dtk + 6^d)m\frac{1-q^h}{1-q})$.



(a) Root grid contains 3 subgrids grid 3 contains 2 subgrids (b) The AMR hierarchical tree (c) Clustering within grid 3 (d) Clustering extends to level 0 (e) Final clustering result has 4 clusters

Figure 6: AMR clustering applied to the data set of the example shown in Figure 1. (a) Six grids in total are created in which grids 0 and 3 contains 3 and 2 subgrids, respectively. (b) The AMR tree structure that connects all grids created. (c) Clustering inside grid 3 assigns data objects to the clusters c3 or c4 based on the object's distance to the cluster's outlines. (d) Clustering in grid 0. Clustering is extended from level 1 to level 0. (e) The final clustering result contains 4 clusters.

5 Experimental Results.

To evaluate the proposed AMR clustering algorithm, we use the star particle data sets generated from a large-scale production cosmological application, named ENZO, which is currently in use on at least six different sites [6, 18]. ENZO simulates the formation of a cluster of galaxies consisting of gas and stars. The simulation starts near the beginning of the universe when the galaxy is in a relative uniform radiation distribution, and continues until the present day when the star particles are in a highly irregular distribution. The ENZO's outputs are periodical checkpointing data dumps that are used to illustrate the evolution of the galaxy formation. We pick two of ENZO simulation output data sets that are near the end of the simulation, which contain 61440 and 491520 star particles, respectively. We refer the two data sets as data set 1 and 2 in this section. To visualize the data distribution and clustering results, we also run the same sets of data projected in two dimensions. Figure 7(a) shows the spatial distribution of the smaller data set projected in two dimensions.

The nine subgrids discovered by the first level of AMR are shown in Figure 7(b). Figure 7(c) presents the clustering results by assigning the root grid's data objects to one of the clusters formed by its subgrids. This clustering output also represents the same results obtained from other grid-based clustering algorithms that use a single uniform mesh. Figure 7(d) shows five nested subgrids discovered by AMR at the second level of refinement, which can only be detected by using a mesh with finer granularity. Data clustering inside grid 5 assigns the objects of grid 5 to one of the five clusters (subgrids), as illustrated in Figure 7(e). Then, the clustering results of grid 5 is extended to the upper level at root grid in which 13 clusters are generated in total, as shown in Figure 7(f).

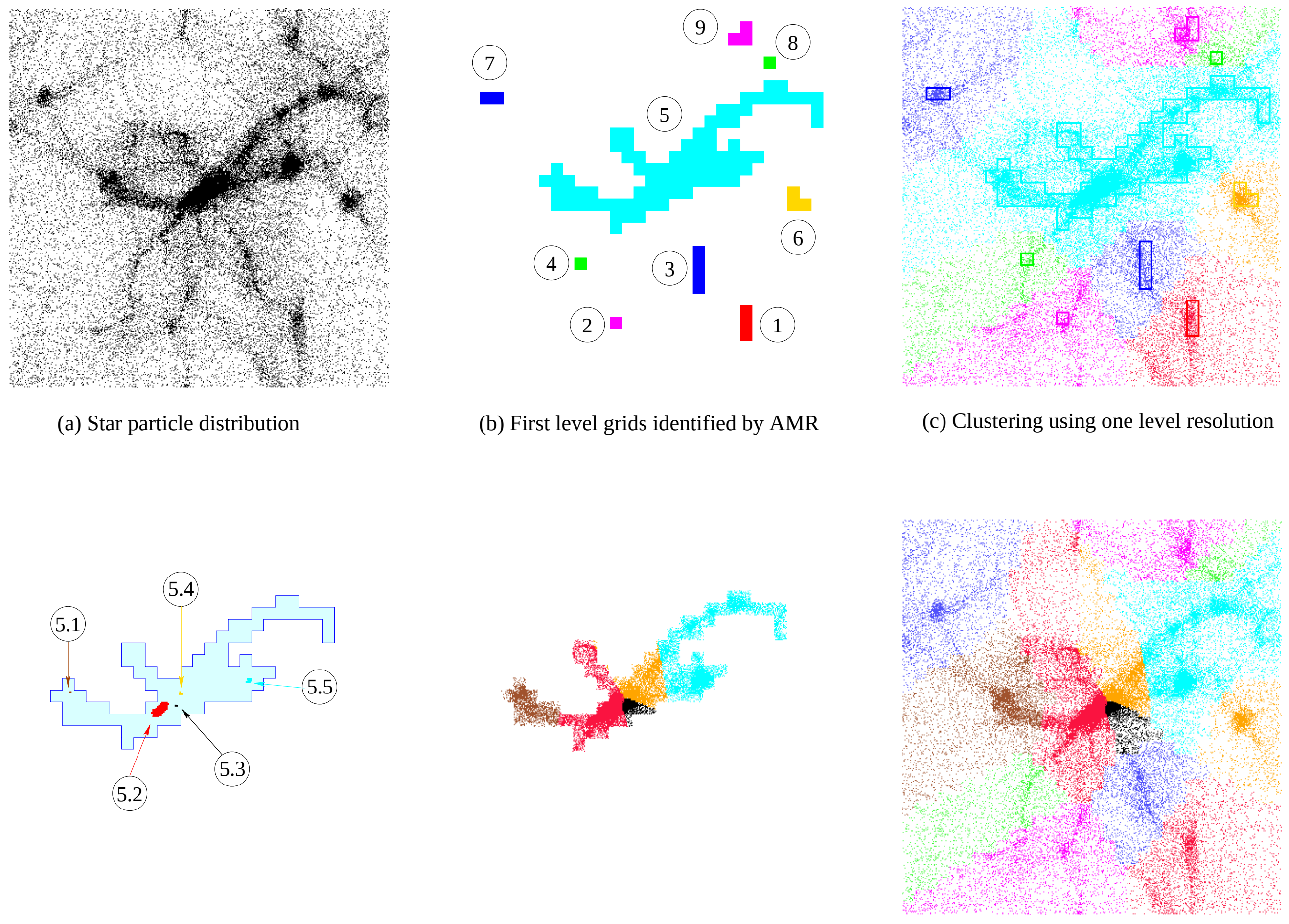
We compare the performance results of using the AMR clustering algorithm with the grid-based clustering algorithm that employs a single uniform mesh for the two sets of ENZO particle data. The projected data sets were made from the three-dimensional data into X-Y dimensions. For data set 1 in 2D projection, we use a uniform mesh of size 128×128 for the uniform grid algorithm and a 32×32 initial mesh for the AMR clustering algorithm. A refine factor of 4 is used in the AMR algorithm so that the mesh granularity at the second level is the same as the uniform mesh case. For data set 1 in 3D, the mesh sizes for the uniform grid algorithm and AMR algorithm are $64 \times 64 \times 64$ and $16 \times 16 \times 16$, respectively. For data set 2 projected in 2D, the mesh of size is set to $64 \times 64 \times 64$ for the uniform grid algorithm and an initial mesh of size $16 \times 16 \times 16$ and a refine factor of 4 are chosen for the AMR clustering algorithm. For data set 2 in 3D, the mesh sizes for the uniform grid algorithm and AMR algorithm are $64 \times 64 \times 64$ and $16 \times 16 \times 16$, respectively. Table 1 gives the performance timings of our experiments that were run on an AMD Athlon XP 1700+ machine with the CPU speed of 1.47 GHz and 512 MB memory. These results clearly show a better performance when using the AMR clustering algorithm over the methods using a single uniform mesh.

hyper parameter

6 Sensitivity to Parameters.

For all clustering algorithms, the clustering quality is sensitive to the input parameters, besides the size of the database, in a certain degree. Examples are the cluster number to the partition-based methods, the merge/split decision to the hierarchical-based methods, and the density radius to the density-based methods. A good quality clustering result often relies on the proper selection of these input parameters. Generally, all grid-based clustering algorithms are sensitive to the parameters of the mesh size, filtering threshold, and the ratio of mesh granularity between two consecutive levels in the hierarchical tree.

The input parameters to the AMR clustering algorithms



(a) Star particle distribution (b) First level grids identified by AMR (c) Clustering using one level resolution

(d) Second level grids identified by AMR (e) Clustering using the grids at the second level. (f) Clustering using 2−level AMR.

Figure 7: ENZO data set and its clustering results. (a) The data distribution of the data set 1 projected in two dimensions that has 61440 particles. (b) Nine subgrids are identified at root grid level. (c) Clustering result when only a single mesh is used. (d) Five additional nested clusters are discovered by the second level refinement in grid 5. (e) Clustering inside grid 5. (d) Final clustering results for the entire database.

besides the database size include the maximum refine level, initial mesh granularity, density threshold, and refine factor. The maximum refine level is the upper bound value that stops constructing deeper AMR tree branches when the refinement reaches this level. Good choice of initial mesh granularity and refine factor can minimize the size of the AMR tree and reduce the number of visits to the database. A proper density threshold is also important that can effectively identify the denser regions to form deeper levels of subgrids and potentially nested clusters. An automatic approach to find a suitable threshold value can use a density histogram on the mesh cells and scan for the first number with a large jump from its previous number in the histogram. This number indicates the density gap at the boundary of a new cluster from the surroundings.

During the construction of the AMR tree, whenever a new grid is created, the data objects located in that grid need to be revisited to fill in the new grid's mesh. For very large databases, an out-or-core implementation for managing data objects located in the AMR tree nodes can be adopted. Each grid can allocate pages of buffer for storing data objects located in this grid's domain. A scan to the database will sequentially add the data objects to the buffer pages of the grids in which the objects are located. Once a page is full, it is written to a file or appended to the file if the grid has more than one page filled. After a data scan is completed at one level of refinement, the file contains the only necessary data to be processed for the next level and no unnecessary data will be revisited.

The depth of the AMR tree determines the frequency of data revisit and, therefore, it is critical to select proper input parameters that determine the tree depth, such as

Table 1: Performance results of clustering on two sets of data. The results of using 2D projected data are presented in the second table. The timing compares the performance of the AMR clustering algorithm and the method using a single uniform mesh.

	dataset 1 in 3D			dataset 2 in 3D		
stage	build mesh	clustering	total time	build tree	clustering	total time
uniform grid	0.3709	0.2780	**0.6489**	0.8403	0.8525	**1.6929**
AMR	0.0730	0.1801	**0.2531**	0.3187	0.5595	**0.8782**

	dataset 1 projected in 2D			dataset 2 projected in 2D		
stage	build mesh	clustering	total time	build tree	clustering	total time
uniform grid	0.1032	0.0791	**0.1823**	0.3293	0.1553	**0.4846**
AMR	0.0390	0.0269	**0.0659**	0.2529	0.0578	**0.3107**

initial mesh resolution, refine factor, and density threshold. Tradeoff between the efficiency and quality exists in the AMR clustering algorithm as in all clustering algorithms. For the clustering algorithms that use a single uniform mesh, database is only visited once, but can suffer from obtaining low quality output, as discussed earlier. AMR clustering can generate a good quality results, but if the AMR tree grows too deep, the overhead of data revisit can become significant.

The advantage of our AMR clustering algorithm is that the different levels of clustering qualities can be obtained. The clustering can start at any level of tree depth and perform upward. It can also be performed within a tree branch for clustering on a specific problem subdomain. The combination of the AMR technique and clustering is especially helpful for many modern scientific applications that are programmed using AMR data structures. Recently, data mining techniques are frequently used to help discover the patterns from the simulation outputs as the stand-alone post-simulation applications. In many cases, the mining process is desired to perform on line so the mining results can be directly fed back to the simulation to tune the parameters and guide the subsequent simulation process. The direct feedback requires the seamless integration of data mining process into the application's data structures. Embedding AMR clustering into the AMR applications is obviously more straightforward than other types of clustering algorithms. By reusing the AMR structure already constructed in the applications, our proposed AMR clustering algorithm shows a great potential on performance enhancement for on-line data analysis that is generally difficult for other clustering algorithms to achieve.

7 Conclusions.

In this paper, we presented a grid-based clustering algorithm using adaptive mesh refinement technique that can dynamically apply meshes with different granularities to the regions with different densities. For highly irregular or concentrated data distributions, a uniform grid cannot perform efficiently or sufficiently to generate the required clustering quality. The adaptability of AMR technique lets the clustering perform at the regions requiring high resolution using finer grid meshes. This approach partitions the problem domain into regions that are represented by the grids in a hierarchical tree. Each grid represents the data in a spatial subdomain and grids at different levels of the tree employ meshes of the different granularity. Data clustering can be start at different levels of the tree that gives a certain resolution of the clustering result or it can perform within a tree branch. This algorithm can dynamically discover nested clusters, which is very useful for highly irregular data distributions. Experimental results presented in this paper also showed the efficiency and effectiveness of the proposed algorithm compared to the grid-based methods using single uniform meshes.

8 Acknowledgments

We would like to thank Dr. Michael Norman from University of California at San Diego and Dr. Greg Bryan from University of Oxford for providing the ENZO source code.

References

[1] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications. In *International Conference Management of Data (SIGMOD'98)*, pages 94–105, Jun. 1998.

[2] M. Ankerst, M. Breunig, H.P. Kriegel, and J. Sander. Optics: Ordering Points to Identify the Clustering Structure. In *International Conference Management of Data (SIGMOD'99)*, pages 49–60, Jun. 1999.

[3] M. Berger and P. Colella. Local Adaptive Mesh Refinement for Shock Hydrodynamics. *Journal of Computational Physics*, 82(1):64–84, May 1989.

[4] M. Berger and J. Oliger. Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations. *Journal of Computational Physics*, 53:484–512, Mar. 1984.

[5] C. Birdsall and A. Langdon. *Plasma Physics Via Computer Simulation.* McGraw-Hill, New York, 1985.

[6] G. Bryan. Fluids in the Universe: Adaptive Mesh Refinement in Cosmology. *Computing in Science and Engineering*, 1(2):46–53, Mar. 1999.

[7] A. Calder, B. Curtis, L. Dursi, B. Fryxell, G. Henry, P. MacNeice, K. Olson, P. Ricker, R. Rosner, F. Timmes, H. Tufo, J. Turan, and M. Zingale. High Performance Reactive Fluid Flow Simulations Using Adaptive Mesh Refinement on Thousands of Processors. In *Supercomputing*, Nov. 2000.

[8] A. Dempster, N. Laird, and D. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, Nov. 1977.

[9] D. Eisenstein and P. Hut. Hop: A New Group Finding Algorithm for N-body Simulations. *Astrophysics Journal*, 498(1):137–142, 1998.

[10] M. Ester, H. Kriegel, J. Sander, , and X. Xu. A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *the 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96)*, pages 226–231, 1996.

[11] S. Guha, R. Rastogi, and K. Shim. CURE: An Efficient Clustering Algorithm for Large Databases", Databases. In *International Conference Management of Data (SIGMOD'98)*, pages 73–84, Jun. 1998.

[12] J. Han and M. Kamber. *Data Mining: Concepts and Techniques.* Morgan Kaufmann, San Francisco, California, 2001.

[13] A. Hinneburg and D. Keim. An Efficient Approach to Clustering in Large Multimedia Databases with Noise. In *International Conference on Knowledge Discovery and Data Mining (KDD'98)*, pages 58–65, Aug. 1998.

[14] R. Hockney and J. Eastwood. *Computer Simulation Using Particle.* Adam Hilger, Bristol, England, 1988.

[15] G. Karypis, E. Han, and V. Kumar. CHAMELEON: A Hierarchical Clustering Algorithm Using Dynamic Modeling. *Computer*, 32(8):68–75, 1999.

[16] L. Kaufman and P. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis.* John Wiley & Sons Inc., New York, 1990.

[17] J. MacQueen. Some Methods for Classification and Analysis of Multivariate Observations. In *the 5th Berleley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, 1967.

[18] M. Norman, J. Shalf, S. Levy, and G. Daues. Diving Deep: Data-Management and Visualization Strategies for Adaptive Mesh Refinement Simulations. *Computing in Science and Engineering*, 1(4):36–47, Jul. 1999.

[19] G. Sheikholeslami, S. Chatterjee, and A. Zhang. WaveCluster: A Multi-Resolution Clustering Approach for Very Large Spatial Databases. In *the 24th International Conference on Very Large Data Bases (VLDB'98)*, pages 428–439, Aug. 1998.

[20] D. Walker. Characterizing the Parallel Performance of A Large-Scale, Particle-in-Cell Plasma Simulation Code. *Concurrency: Practice and Experience*, 2(4):257–288, December 1990.

[21] W. Wang, J. Yang, and R. R. Muntz. STING: A Statistical Information Grid Approach to Spatial Data Mining. In *the 23rd International Conference on Very Large Data Bases (VLDB'97)*, pages 186–195, 1997.

[22] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: an Efficient Data Clustering Method for Very Large Databases. In *International Conference Management of Data (SIGMOD'96)*, pages 103–114, Jun. 1996.