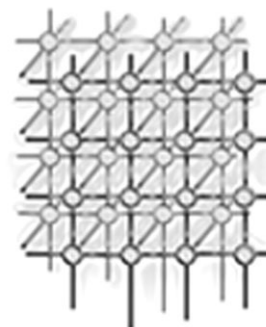


## Clustering revealed in high-resolution simulations and visualization of multi-resolution features in fluid–particle models



Krzysztof Boryczko<sup>1</sup>, Witold Dzwinel<sup>1</sup> and David A. Yuen<sup>2,\*</sup>,<sup>†</sup>

<sup>1</sup>*AGH Institute of Computer Science, al. Mickiewicza 30, 30-059, Kraków, Poland*

<sup>2</sup>*Minnesota Supercomputing Institute, University of Minnesota, Minneapolis, MN 55455-1227, U.S.A.*

---

### SUMMARY

Simulating natural phenomena at greater accuracy results in an explosive growth of data. Large-scale simulations with particles currently involve ensembles consisting of between  $10^6$  and  $10^9$  particles, which cover  $10^5$ – $10^6$  time steps. Thus, the data files produced in a single run can reach from tens of gigabytes to hundreds of terabytes. This data bank allows one to reconstruct the spatio-temporal evolution of both the particle system as a whole and each particle separately. Realistically, for one to look at a large data set at full resolution at all times is not possible and, in fact, not necessary. We have developed an agglomerative clustering technique, based on the concept of a mutual nearest neighbor (MNN). This procedure can be easily adapted for efficient visualization of extremely large data sets from simulations with particles at various resolution levels. We present the parallel algorithm for MNN clustering and its timings on the IBM SP and SGI/Origin 3800 multiprocessor systems for up to 16 million fluid particles. The high efficiency obtained is mainly due to the similarity in the algorithmic structure of MNN clustering and particle methods. We show various examples drawn from MNN applications in visualization and analysis of the order of a few hundred gigabytes of data from discrete particle simulations, using dissipative particle dynamics and fluid particle models. Because data clustering is the first step in this concept extraction procedure, we may employ this clustering procedure to many other fields such as data mining, earthquake events and stellar populations in nebula clusters. Copyright © 2003 John Wiley & Sons, Ltd.

KEY WORDS: large-scale data sets; visualization; feature extraction; parallel clustering; dissipative particle dynamics; fluid particle model

---

\*Correspondence to: David A. Yuen, Minnesota Supercomputing Institute, University of Minnesota, Minneapolis, MN 55455-1227, U.S.A.

<sup>†</sup>E-mail: davey@krissy.msi.umn.edu

Contract/grant sponsor: Complex Fluids Program of the U.S. Department of Energy

Contract/grant sponsor: Energy Research Laboratory Technology Research Program of the Office of Energy Research, U.S. Department of Energy under subcontract from the Pacific Northwest National Laboratory

Contract/grant sponsor: KBN (Polish Committee of Scientific Research); contract/grant number: 4 T11F 02022

---



## INTRODUCTION

Visualization of multi-resolution structures occurring in large-scale simulations or experimental set-ups of natural phenomena involves invariably the processing of large-scale data sets with complex topologies in many spatio-temporal scales. For example, in geophysical fluid dynamics, dynamical processes are characterized by a strong variability over a wide range of spatial and temporal scales from nanoseconds and nanometers to millions of years and kilometers [1]. We do not need to look at the data at full resolution all the time. In fact, looking at the full data set is sometimes not a viable option. Instead we really need techniques to view and interrogate data at various resolution levels, trading off resolution for interactivity or for focusing detailed attention on the scale of interest. In feature extraction of structured data, which can be represented by continuous fields with smooth derivatives, image processing techniques and filters, such as wavelets [2], can be employed. In medical imaging and computer graphics where data may originate from fluid and other materials which may be partially transparent and should be modeled as such, volume rendering that uses ray casting and ray tracing is currently in popular usage. These techniques, however, are not appropriate for discrete systems which are composed of various types of particles producing complex microstructures.

Simulations involving discrete particles are used for studying phenomena in microscopic and mesoscopic scales, where atomistic interactions, thermal fluctuations and excluded volume effects go with large-scale features. Particle methods involve a broad range of models [3–10] such as: molecular dynamics (MD), dissipative particle dynamics (DPD), a fluid particle model (FPM), granular dynamics (GD), direct simulation Monte Carlo (DSMC) methods, particle-in-cell (PIC) methods, smoothed particle hydrodynamics (SPH) and many other techniques in numerical simulations. Particles, which can represent atoms, molecules, ‘lumps’ of fluid, grains of solids, micelles, moving grid nodes, rocks, faults, galaxies or abstract objects [11,12], interact with each other, and evolve according to Newtonian mechanics or other prescribed local rules such as cellular automata [13], as is developed in the lattice Boltzmann gas (LBG) method [9,10]. The spatio-temporal scale of a particle system depends on the particle definition and the number of particles in the system. The large-scale MD simulations in three dimensions performed on current computers employ  $10^6$ – $10^9$  particles [3,4,14], depending on the range and the complexity of interactions. For simulations involving  $10^6$  particles and covering  $10^4$ – $10^5$  time steps, the data-set volume produced in a single run is about 450 Gbytes to 4.5 Tbytes, if the data is saved at each time step. Nonlinear interactions between particles resulting from conservative, dissipative, depletion forces and hydrodynamic interactions [15] result in clustering, which produces a variety of multi-resolutional aggregates such as molecules, micelles, crystals, colloidal aggregates [16], polymeric chains and dispersion microstructures [17]. Extracting these clusters from the whole ensemble is very important for studying microstructural properties of complex and porous materials.

The spatio-temporal hierarchy of ‘particles’ and microstructures, which represents the multi-resolutional character of matter, cannot be visualized by employing classical techniques used in computer graphics, e.g. volume rendering and ray casting algorithms. The large amount of unstructured data merging in different ways at various levels of this hierarchy makes these time-consuming techniques useless. Moreover, the particle features, which can be represented as an  $N$ -dimensional vector, yield complex clustering patterns in higher dimensions, in which  $N \gg 10$ , which are inaccessible for volume rendering techniques. Unlike continuum systems, in which the ultimate resolution can be achieved by increasing the number of grid points, the particle representation can display the finest resolution level [11]. Discrete particles are singular entities and global features



of the particle ensemble are unveiled after averaging. However, to obtain a single grid point of, e.g. a velocity field, the velocities of at least 100 to 1000 particles should be averaged. Even for a  $10^9$  particle ensemble, the resulting resolution after averaging may appear too crude for extracting finer features, such as those found along grain boundaries. Moreover, we will otherwise lose valuable information concerning the clustering of particles.

Because of the large amount of data involved there is a need to run the clustering procedure on high-performance parallel architectures. For example, the Landsat data (about one terabyte per day, represented by 4- or 256-dimensional vectors) are analyzed by using an efficient, parallel, continuous K-means algorithm developed in Los Alamos, which is run on the top high-performance systems [18]. For the clustering of large spatial data bases and image segmentation a number of workstations connected via the Ethernet is also used [19]. Large-scale parallel clustering is performed in a NOW (Network of Workstations) environment by using a client-server model, in which the clustering task is divided among a set of clients that report their intermediate results to a single server process [20]. This parallel software is an ideal candidate for many Grid computing projects dealing with processing and concept extraction from a large amount of data.

For the analysis of data from large-scale simulations with particles, the non-hierarchical algorithms cited above are inadequate due to the lack of an *a priori* knowledge about the number of clusters, the high computational complexity of the algorithms and serious problems concerned with avoiding local minima in the minimization of the cost function. Due to the local nature of the particle microstructures the hierarchical agglomerative techniques fit better for their extraction. As shown in [21], optimal butterfly and tree parallel algorithms exist for hierarchical clustering for various proximity measures between clusters. The efficiency of agglomerative clustering can be greatly improved by using fast procedures for nearest-neighbor (NN) searches. For example, the Friedman's projection method [22] can be used in extracting high-dimensional clusters.

In this paper we present an efficient parallel algorithm devoted to agglomerative clustering, which is based on the mutual nearest-neighbor (MNN) concept [23] and the linked-list method [24] for NN search. First, we discuss the concept of agglomerative clustering and we show its homogeneity with the particle model. This clustering scheme will be employed to conduct clustering of large-scale data sets, which consist of up to 16 million particles. Then, the MNN parallel algorithm is introduced and timings from IBM SP and SGI/Origin 3800 parallel systems are compared. We show examples of the application of MNN clustering in extracting features from simulations of complex fluids. Finally, to sum up, we discuss the conclusions and implications of this novel procedure.

## SERIAL CLUSTERING ALGORITHM

Clustering is a fundamental concept in pattern recognition [25–27], but also has many applications in fields such as earthquake physics, astrophysics and polymer fluid dynamics [30–35]. Clustering is basically a threshold phenomenon, which occurs in states far enough from equilibrium [16]. Clustering is used for classifying similar (or dissimilar) patterns represented by  $N$ -dimensional vectors. Depending on the data structures, different clustering schemes must be used. There are two principal approaches for classifying data. The first one consists of a non-hierarchical extraction of clusters. This approach is used mainly for extracting compact clusters by using global knowledge of the data structure. The most well known and simplest techniques are based on the K-means algorithm

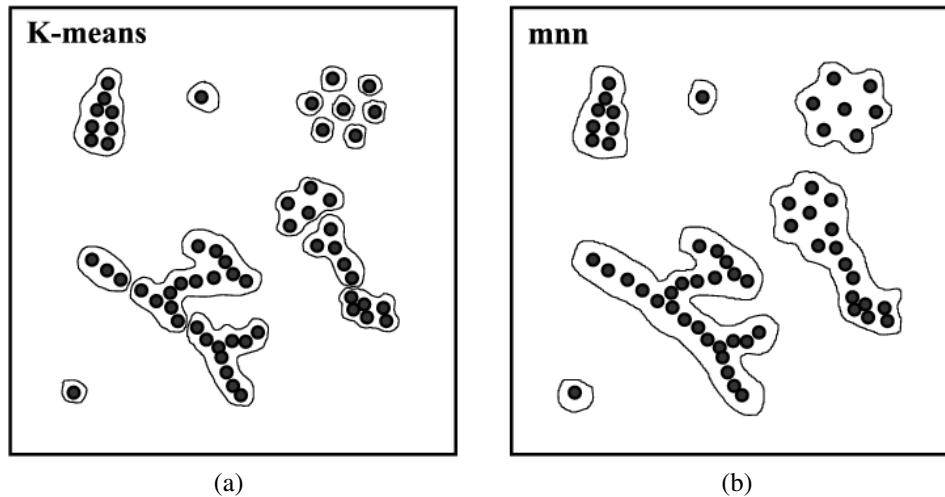


Figure 1. A schematic drawing showing the results of (a) K-means and (b) MNN clustering on the same two-dimensional synthetic data set.

[18,25–27,36]. The main problem with K-means is that the final classification represents the local minimum of the criterion function. Thus for multi-modal functions, starting from different initial iterations, one can obtain different cluster structures. Because the global information about the data is required for the non-hierarchical schemes, the non-hierarchical algorithms also suffer from a high computational complexity and most of them require an *a priori* knowledge about the number of clusters. In Figure 1 we show how the K-means algorithm searches for compact clusters with regular density. As shown in Figure 1(a), it does not work for elongated, bridged clusters and clusters with a different density appearing in simulations with particles [23].

Agglomerative clustering schemes [25–27] consist of the subsequent merging of smaller clusters into larger clusters, based on a proximity criterion. Depending on the definition of the proximity measure, there exist many agglomerative schemes such as: average link, complete link, centroid, median and minimum variance algorithm. The hierarchical schemes are very fast for extracting localized clusters with non-spherical shapes. All of them suffer from the problem of not having properly defined control parameters, which can be matched to the data of interest and which can hence be regarded as invariants for other similar data structures.

In the MNN algorithm proposed by Gowda and Krishna in [23] invented almost 25 years ago, the proximity measure between merging clusters is constructed on the basis of two types of distances: **mnn** distance and the classical Euclidean metrics. The **mnn** distance comes from the MNN concept which can be outlined as follows.

1. Let us consider an ensemble consisting of  $Np$  particles and  $i = 1, 2, \dots, Np$ .
2. Find the lists  $L_i$  of  $Ncut$  nearest neighbors  $j$  in  $RCN$  cut-off radius for each particle  $i$ .



3. Sort out the neighbors  $j$  in the lists in ascending order according to the Euclidean distance between the particles  $i$  and  $j$ . Thus  $L_i(k) = j$ , where  $k$  is the position of the particle  $j$  in the  $i$ th list and  $L_j(m) = i$ , where  $m$  is the position of the particle  $i$  in the  $j$ th list.
4. Because  $L_i(k) = j$  and  $L_j(m) = i$ , compute **mnn** distances defined as  $\mathbf{mnn}(i, j) = m + k$ . The maximum **mnn** distance is less than  $2 \cdot Ncut$ .

As a difference from the original scheme, we have introduced an additional threshold value  $RCN$ . The values of  $Ncut$  and  $RCN$  control the size of clusters (e.g., the length of polymeric chains). The  $\mathbf{mnn}(i, j)$  distances are sorted out in ascending order. The particle pairs  $(i, j)$  having the same  $\mathbf{mnn}(i, j)$  distance are also sorted out the same way but according to the Euclidean distance between the particles.

A serial agglomerative clustering algorithm is given as follows:

### 1. INITIALIZATION—NEIGHBORS SEARCH

- a. Choose  $R_0 = \{C_i = \{\mathbf{x}_i\}, i = 1, \dots, Np\}$
- b. Create the list  $\Omega(t)$ , of particle pairs  $(i, j)$  where  $t = 0, 1, 2, \dots, M$  and  $M$  is the number of pairs, sorted out in increasing **mnn** and the Euclidean distances between  $i$  and  $j$
- c.  $t = 0$

### 2. REPEAT—MERGE CLUSTERS

- a. Check if the particles  $(i, j)$  from  $\Omega(t)$  belong to different clusters  $C_i$  and  $C_j$ , respectively.
- b. If not:  $t = t + 1$  and go to 2
- c. If yes: merge clusters, i.e.  $C_i = C_i \cup C_j$
- d.  $R_t = R_{t-1} - \{C_j\}$
- e.  $t = t + 1$

### 3. FINISH if $t = M$

The value of  $M$  is usually chosen as being equal to the greatest value of **mnn**.

In Figure 2 we compare the clustering results by using K-means and MNN schemes on the real data set. The use of an inappropriate clustering scheme leads to serious errors in defining the discrimination surface. As shown in [23] and in Figures 1(b) and 2(b), the MNN scheme can extract clusters of complicated shapes with irregular structures. Such structures are found in discrete particle simulations of molecules, micelles, colloidal crystals and aggregates [16,17,40–42].

Finding the NNs consumes most of the computational time of the **mnn** scheme, as is also the case for particle-based algorithms. For searching for neighbors in 3D space, however, there exists an efficient  $O(Np)$  algorithm, which employs the linked-list concept [24]. The linked-list scheme is widely used in computations of short-ranged forces in particle simulations. Therefore, the neighbor tables obtained in the *forces()* procedure can be reused for clustering or recomputed in the same way.

To measure the efficiency of the serial clustering code written in FORTRAN 95, we performed our tests on an IBM SP with Power3+ and an SGI/Origin 3800 system with R14000/500 CPUs. Both processors have a secondary cache with 8 Mbytes. We used three data files representing particles positions from the same short simulation of blood cell flow in a capillary vessel by using a FPM for different  $Np$  (the number of particles). After 1000 time steps of thermalization, the FPM particles are simulated in the following 1000 time steps. We have performed clustering as a post-processing

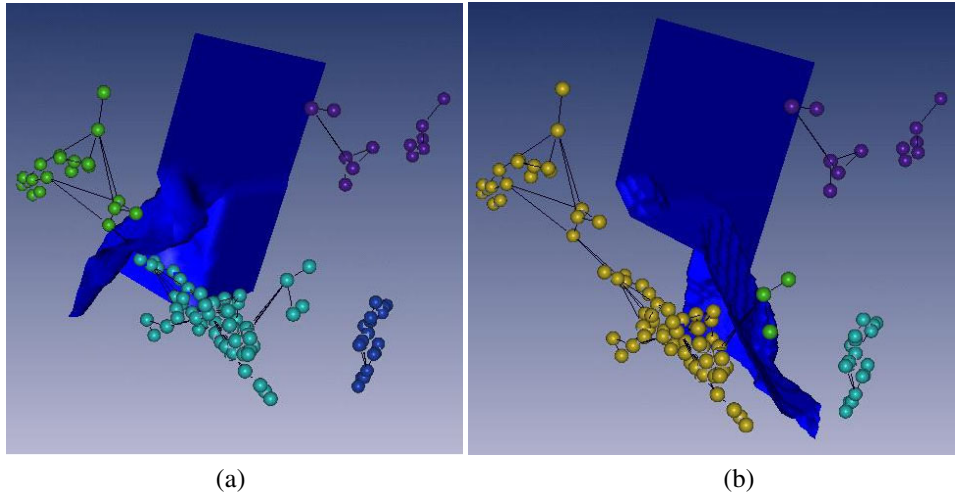


Figure 2. The results of (a) K-means and (b) MNN clustering on the same 256-dimensional data set representing vibration modes of the IBR-2 (Russia-Dubna) nuclear reactor. The nonlinear Sammon’s mapping [37,38] from 256- into three-dimensional space was employed for dimensionality reduction. Discrimination surfaces are shown for both cases. Visualization was made by using the visualization package Amira 2.3 [39].

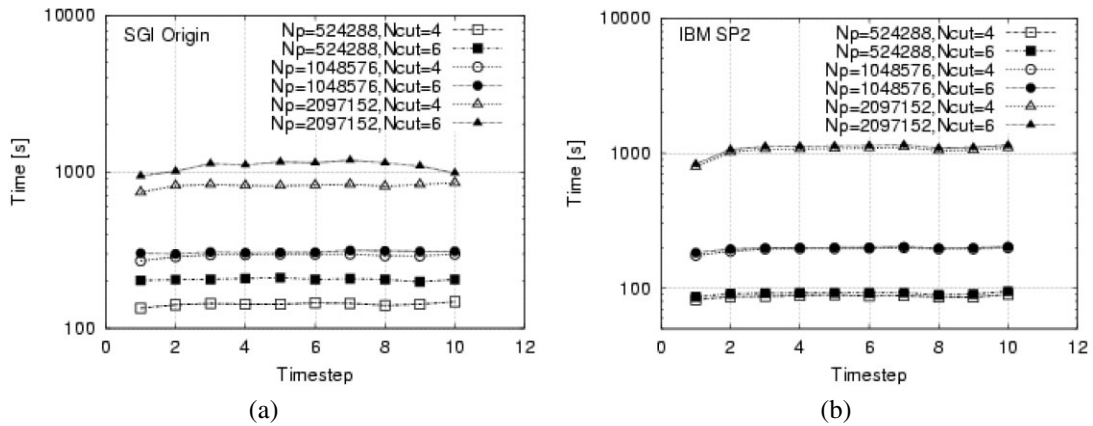


Figure 3. Timings on SGI/Origin 3800 (a) and IBM SP (b) with a Power3+/375 chip. The data sets consisted of FPM particles simulated in 1000 time steps. Clustering is made off-line with the simulation at every 100 time steps.



procedure, at every 100 time steps for different  $N_{cut}$  values. In Figure 3 we show the various numbers of particles as a function of clustering steps.

The most striking observation reveals that the computational time for the clustering algorithm with linear computational complexity  $O(Np)$ —experimentally confirmed for  $Np$  less than one million particles—increases almost five times when  $Np$  increases from one million to two million particles. This sharp loss in efficiency can be observed for both the IBM SP and SGI/Origin systems. The size of data arrays for 2 million particles is about 60 Mbytes. The arrays cannot fit in the cache. To avoid frequent cache misses the particles are renumbered at regular periods. As a consequence, the particles residing in the same cell have consecutive indices and thus they are close to one another in the computer memory. However, the gap between particle indices still exists for particles from different cells. This is due to the linearly addressed computer memory. By increasing the size of computational box  $2^{1/3}$  times in the  $x$ ,  $y$  and  $z$  directions, the gap between particle numbers from neighboring cells increases almost 60%. Because the large clusters not only span along neighboring cells, the cache misses will still be frequent. Therefore, the procedure for merging the clusters represents the most inefficient part of the code.

From the plots in Figure 3 we can conclude that the Power3+/375 processor is about two times faster than R14000/500 for  $N_{cut} = 6$  but the cache performance is better for SGI/Origin. Surprisingly, for  $N_{cut} = 4$  the SGI processor is only 50% slower. This may be due to a different optimization of the nested loops in the merging cluster procedure, which uses only fixed point arithmetic, by the two compilers.

## PARALLEL ALGORITHM

As shown in Figure 4, the parallel algorithm for MNN clustering has a similar structure to the algorithm for computing short-ranged forces in the particle ensemble [43]. We consider an isothermal two-dimensional system, which consists of  $Np$  particles. The particles are enclosed within an elongated rectangular box. The box is divided into cubic cells with edge size equal to the range of particle interactions  $R_{cut}$ . Because the cell size has been chosen to be slightly larger than  $R_{cut}$ , all the neighbors contributing to the total force acting on a given particle must be located within the cell containing the particle or within the adjacent cells. This key concept of the linked-list algorithm enables us to locate the NNs for all the particles by using the algorithm of  $O(Np)$  complexity. These neighbors can be stored in auxiliary arrays and reused, e.g. in the construction of higher-order predictor–corrector numerical schemes time stepping the equations of motion or deriving radial-distribution functions (RDFs).

Parallel computing requires the decomposition of the computation into subtasks and their mapping onto multiple processors. The total volume of the box is divided into  $P$  subsystems of equal volume, and each subsystem is assigned to a node in an array of  $P$  processors. By using the single program multiple data (SPMD) paradigm, commonly used for MD code parallelization [3,14], each processor follows an identical predetermined sequence to calculate the forces on the particles within the assigned domain. The particles from cells, which are situated on the boundaries of processor domains, are copied to proper neighboring processors (see Figure 4). These boundary cells control the overhead in the communication. Each processor sends the message only in one direction to its closest neighbor.

For clustering on-line with a simulation, we have reused the auxiliary neighboring tables within the framework of the MNN algorithm. In the case of off-line clustering the same linked-lists procedure can be applied for a closest neighbor search. However, instead of the interaction range  $R_{cut}$ , the largest

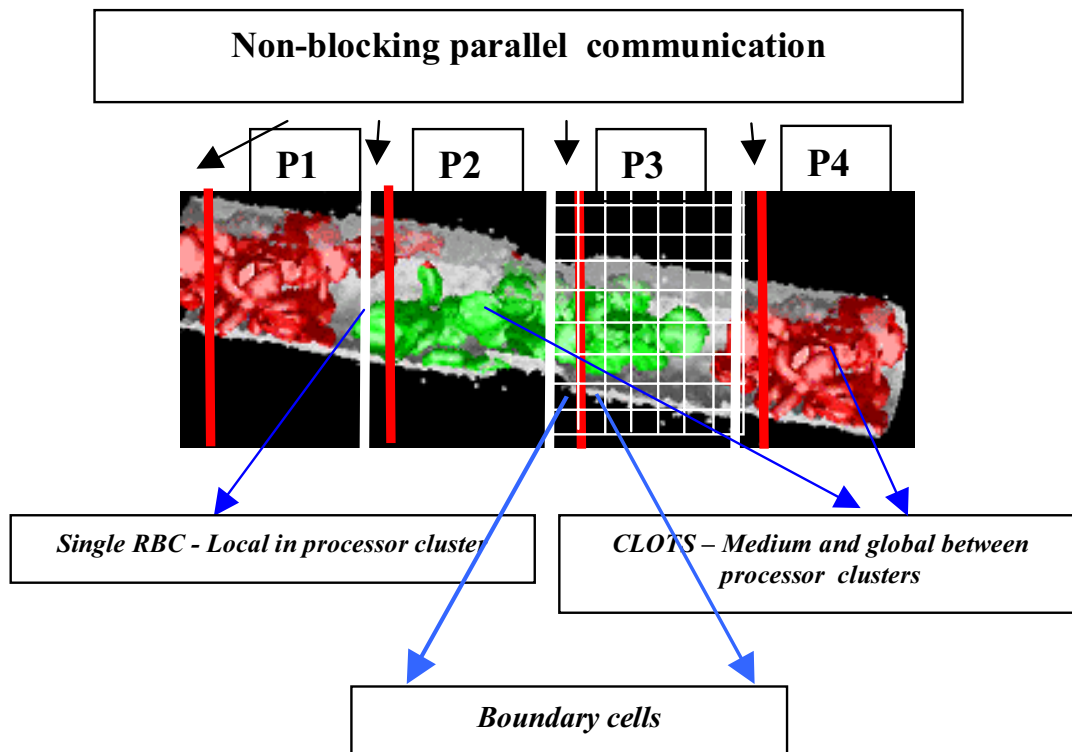


Figure 4. A diagram of processor communication in the FMP and the agglomerative clustering procedure.

distance the NN particles from the single cluster can be separated out  $RCN$  has to be considered. In these simulations we define  $RCN \approx Rcut$ . In the original MNN scheme the points can merge into a cluster if its **min** distance is less than a given value. In the scheme proposed here both criteria are taken into account. They support both a mechanism for extracting clusters with a different density ( $Ncut$ , number of NNs considered) and control the size of clusters ( $RCN$ ).

In Figure 5 we present a more detailed meta-code of an MNN parallel algorithm, which consists of parallel procedures for finding neighbors and merging local and medium cross-domain clusters. The communication between processor domains is more complex than in the fluid particle code [43]. In finding the NN we allow communication to take place only for the adjacent processor. It is the same as in the fluid particle simulation code [43]. Differences appear for merging medium cross-boundary clusters.

The communication remains local, i.e. it occurs for adjacent domains, but may span to a larger number of cells, not only these being close to the domain borders. Merging medium clusters in one large cluster spanning along several processor domains involves global communication. However, this



```

-----PARALLEL PART-----
PAR
  Forall domains P = 1, N
    Divide_Domains_onto_Cells;
    Renumerate_Particles_in_Linked_Cells;
    SENDfrom_P-1_to_P_particles_from_boundary_cells;
    Find_Neighbors for all particles in P domain;
    SENDfrom_P_to_P-1_neighbors_of_particles_from_boundary_cells;
    Find_Clusters-in-P domain;
    SENDfrom_P_to_P-1_particle_clusternumbers_from_boundary_cells
    Merge_cross-boundary_Clusters-in-P
  endforall
ENDPAR
-----SEQUENTIAL PART-----
SEQ
  Merge_Large_cross-processors_clusters()
  Get_statistics ()
ENDSEQ
-----MAIN CLUSTERING PROCEDURES-----
..... Find neighbours .....
procedure Find_Neighbors_in_P()
  for icell = 1, num_of_cells_in P
    iparticle = first_in_icell_link_list()
    while (end_of_link_list_of_icell != 0)
      for jcell = 1, num_of_neigh_cells_to_icell+1
        jparticle = first_in_jcell_link_list()
        while (end_of_link_list_of_jcell != 0)
          dist = distance_between (iparticle, jparticle)
          if (dist < RCN) then
            update_list_of_mnn_neighbours_of_iparticle
            update_list_of_mnn_neighbours_of_jparticle
          endif
          jparticle = next_in_jcell_link_list()
        endwhile
      endfor
      iparticle = next_in_icell_link_list()
    endwhile
  endfor
..... Merge Clusters .....
procedure Find_Clusters_in_P()
  run mnn agglomerative procedure
procedure Merge_cross-boundary_Clusters-in-P_domain
  if(boundary_cell_particle_i ∈ Cluster_A_from P - 1, and,
  boundary_cell_particle_i ∈ Cluster_B_from_P) then
    Merge (A,B) → SEND_to_Merge_Large_cross-processors_clusters;
  endif

```

Figure 5. The parallel algorithm of MNN clustering.

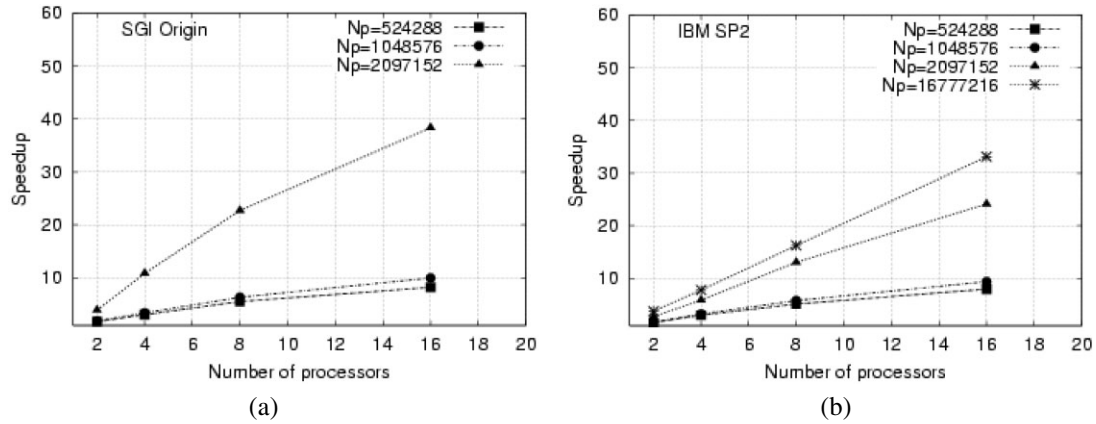


Figure 6. Timings on the SGI/Origin 3800 (a) and the IBM SP (b) with a Power3+/375 chip. The data sets consist of FPM particles simulated in 1000 time steps. Clustering has been carried out off-line with the simulation at every 100 time steps.

usually occurs at the end of the computations. Final merging is fast and can be computed sequentially on the processors, which are defined as the master. Both linked-list and merging cluster procedures are of  $O(Np)$  complexity. As shown in the previous section, due to inefficient use of the cache the performance of the serial code decreases dramatically for  $Np > 10^6$ . Dividing the large task onto several smaller ones should result in a more efficient use of the cache and a considerable increase of the speed performance.

The MNN parallel code was written in FORTRAN 95 and was implemented on the MPI interface for both the IBM SP and SGI/Origin 3800 platforms. We performed our tests on an IBM SP with Power3+ processors (four processors in a single node) with 8 Gbytes of memory per node. For comparison we present the benchmarks for the SGI/Origin 3800 system with R14000/500 CPUs (two processors per node) with 4 Gbytes of memory per node.

As we can anticipate from the previous section (see also Figure 6), we can obtain a super-linear speed-up obtained for  $Np = 2 \times 10^6$  due to the influence of the cache. It is clearly seen that the speed-up is almost two times greater for the SGI/Origin system than for the IBM SP for a large number of particles (SGI, speed-up = 40 for 16 processors and two million particles; SP, speed-up = 33 on 16 processors for 16 million particles). For  $Np < 10^6$  the two speed-ups are very similar. The decrease of speed-up is observed for a larger number of CPUs than eight when CPUs from distant nodes are involved in the computations.

## SIMULATION RESULTS

For testing purposes we have employed the serial MNN clustering code to extract 2D patterns from three types of DPD simulations: phase separation in a binary fluid, droplet condensation and

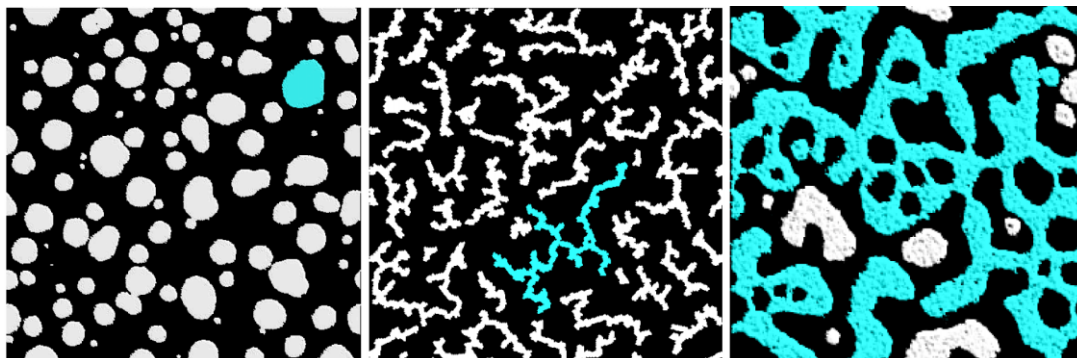


Figure 7. Snapshots from the 2D DPD simulation of condensation, colloid aggregates and phase separation in a binary fluid. The largest clusters extracted by MNN clustering are shown in gray.

agglomeration of colloids. As shown in Figure 7, we did not encounter any problems dealing with MNN clustering in the course of extracting both regular condensation patterns, irregular colloidal agglomerates and shapeless structures during the phase separation stage.

We have implemented the clustering procedure to extract clusters in 3D from the large-scale data sets obtained from the simulation of colloidal structures with the FPM [8]. We display in Figure 8 a few examples from the three different kinds of simulations. In Figure 8(a) we show the clusters obtained for a colloidal slab accelerated in a particle fluid (black in Figure 8(b)).

Clustering allows for the visualization of different types of dispersive microstructures. The largest and the densest cluster of particles is depicted in white. The medium-sized clusters are cast in red and single particles are shown in blue. A different dispersing structure can be seen in the figure on the right. Long red streaks of particles are produced behind the slab head. They are accurately extracted by this clustering code.

The dispersing structures, which are portrayed in Figure 8(b), can be observed for fragmentation produced by the shattering mechanism. Clustering allows for the observation over time of the largest solid part in the slab body. The temporal evolution of very complicated shapes of clusters obtained during colloidal agglomeration can also be clearly delineated by the method of MNN clustering. The extraction of even more complicated clusters is displayed in Figure 9.

Clustering becomes an indispensable tool for the extraction of multi-level structures created by complex two-phase flows such as red blood cells (RBCs) flowing in microscopic vessels. Two upper images in Figure 9 present two snapshots from simulations of RBC clotting close to the constriction both for the ‘healthy’ (left) and ‘sickle’ (right) cells. The following image shows the clotting in bifurcating capillaries. The RBCs are made of particles coupled by harmonic forces. The two-level MD-FPM particle model was implemented.

Automatic identification of the clot size is difficult due to the existence of a hierarchy of multi-resolution structures: particles, RBCs and clots of cells. The clots may be represented by medium and large between-processor clusters. Extracting such clusters consisting of flexible agglomerates

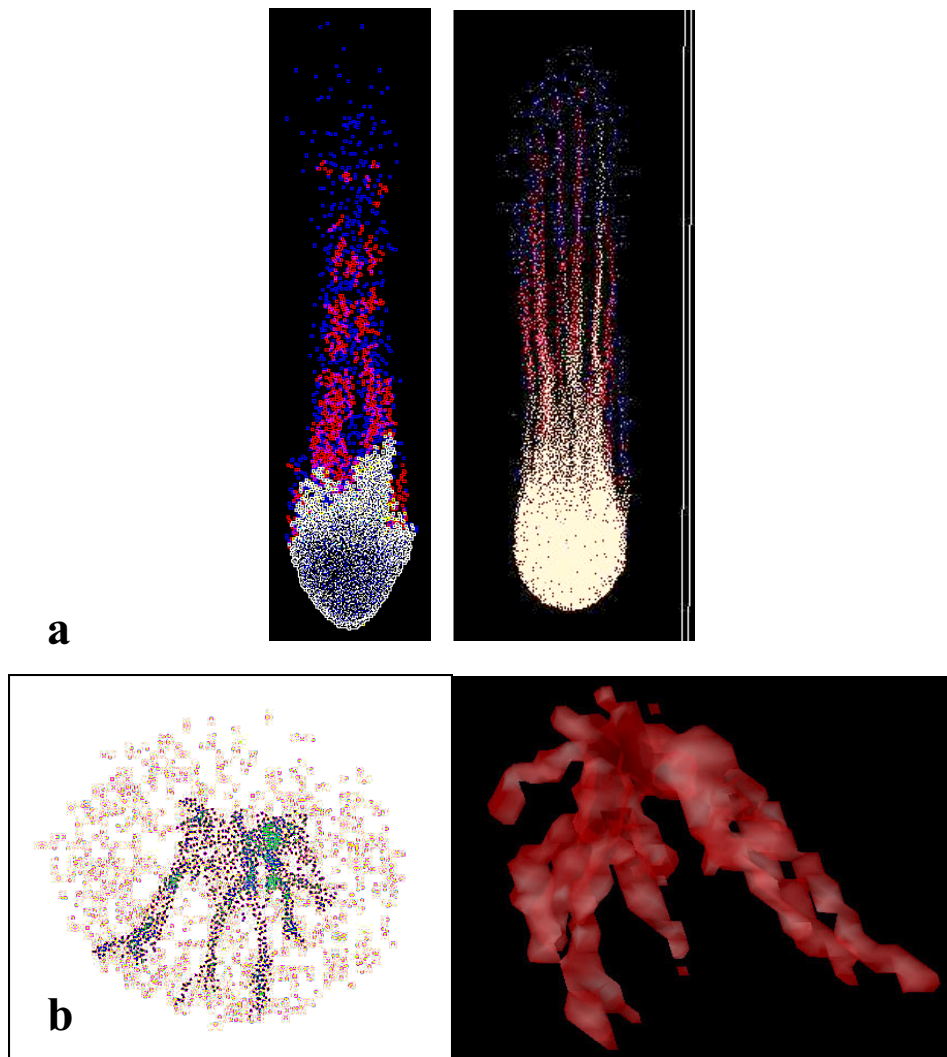


Figure 8. 3D results of clustering for the dispersion of a slab in a FPM particle fluid [8,17].

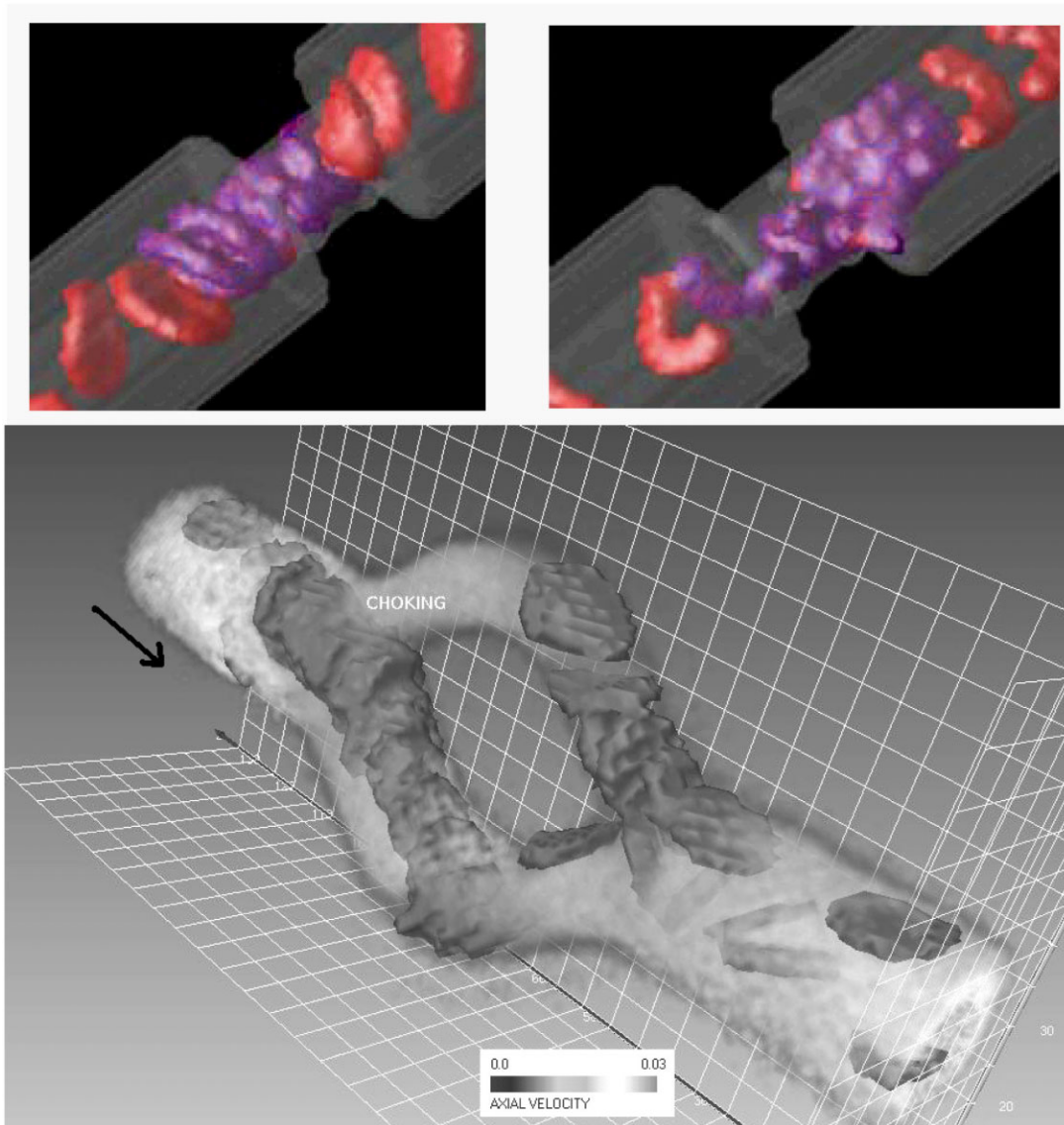


Figure 9. 3D visualization of clustering combined with volume rendering of the RBCs flowing through a choking point in a narrow blood vessel simulated by using a two-level MD-FPM code. Different clot sizes are shown for normal RBCs and sickle cells. The clots extracted by MNN clustering are violet in color. Visualization of the blood flowing in the bifurcating vessels is made by using Amira 2.3 [39].



of finer resolution, which consist of thousands of particles, is currently impossible when employing the standard K-mean techniques or simple agglomerative clustering. As shown in Figure 9, the **mnn** parallel code can easily handle this prodigious problem.

## CONCLUDING REMARKS

We have demonstrated here that the MNN hierarchical clustering scheme can extract distinct features from large-scale data sets produced in large-scale particle simulations efficiently. These schemes, usually applied for classifying  $N$ -dimensional patterns, are also truly promising tools for visualizing voluminous complex scientific data. Unlike the standard algorithms deployed in computer graphics, such as volume rendering, ray casting and ray tracing, the clustering schemes can be applied when extracting effortlessly valuable information from unstructured, irregular data, such as found in finite-element calculations and wavelets, besides the discrete particle simulations treated here.

Unlike other agglomerative schemes, which employ local metrics, the MNN distance does not only depend on the two particle positions. The MNN distance between two particles combines sorted lists of the nearest neighbors of these particles. Thus the **mnn** distance is not as local as, e.g., the Euclidean distance or  $L^2$ -norm distance and not as demanding as the Mahalanobis metrics. It reflects a cluster structure in the neighborhood of a particle defined by  $N_{cut}$ , the number of neighbors considered. Therefore, MNN clustering can self-adapt to the local resolution of a bunch of complex clusters with multi-resolution structures.

High complexity of the algorithms for finding NNs makes the clustering methods slow for large-scale and highly inhomogeneous multi-dimensional data. Fortunately, due to the high degree of locality of both agglomerative clustering and short-ranged  $N$ -body codes such as MDs and fluid particle methods (DPD, FPM, SPH) and the low dimensionality of the feature space, the same linked-list procedure of  $O(Np)$  complexity can be employed when searching for the neighbors. We show, however, that the serial version of the MNN algorithm is of limited use.

The clustering of more than one million particles for the IBM Power3+/375 and R14000/500 processors results in large overheads coming from inefficient use of the processor cache. The procedure of merging small clusters into larger ones is mainly responsible for this side effect. To allow for the efficient clustering of data along with particle simulations for a larger number of particles, both procedures must be run in a parallel environment. This is especially important for the algorithms where longer-ranged interactions between the particles can be approximated by particle-cluster and cluster-cluster interactions and the clustering procedure must be performed frequently, so that the algorithmic property of the clustering scheme can considerably increase the computational efficiency of the parallel code on the whole. We have demonstrated that the parallel MNN algorithm, running on the IBM SP and SGI/Origin 3800 systems, can produce a super-linear speed-up.

We have also shown various examples of the application of MNN clustering in the visualization of microstructures resulting from simulations of complex fluids represented by up to 16 million fluid particles. Visualization of particle clusters, combined with averaged velocity or density fields, allows complex microstructures appearing in both micro and mesoscopic scales to be studied. As shown in Figure 9, the clusters representing clouds of particles can be visualized in various spatio-temporal scales by using volume rendering algorithms or a standard visualization software equipped with this



technique, such as Amira [39]. This aspect makes both clustering and rendering complementary tools devised for the parallel visualization of multi-resolutional patterns.

These methods can also be utilized beyond these physical scales, for classifying not only objects but also dynamical discrete events such as: earthquakes [28,29,32] and the micro-earthquakes induced by mining activities [31,33], or dissipation phenomena occurring in a dispersed stellar population [34,35]. Clustering of organisms has profound social consequences on social behavior [44]. Environmental variability can cause clustering, e.g. in slums in cities or in plankton growth from marine turbulence. The clustering and feature extraction software can also be applied in large collaborative Grid computing projects [45–47] dealing with processing and concept extraction from large-scale data sets.

#### ACKNOWLEDGEMENTS

We are grateful for discussions with Dr Jim Rustad from PNNL, Professor Gordon Erlebacher from Florida State University and Professor Yehuda Benzion of the University of Southern California. This research was supported by the complex fluids program of the Department of Energy and by the Energy Research Laboratory Technology Research Program of the Office of Energy Research of the U.S. Department of Energy under subcontract from the Pacific Northwest National Laboratory and by the KBN (Polish Committee of Scientific Research) project 4 T11F 02022.

#### REFERENCES

1. Yuen DA, Vincent AP, Bergeron S, Dubuffet F, Ten A, Steinbach V, Starin L. Crossing of scales and non-linearities in geophysical processes. *Problems in Geophysics for the New Millennium*, Boschi E, Ekstrom G, Morelli A (eds.). Editrice Compositori: Bologna, 2000; 403–462.
2. Yuen DA, Vincent AP, Kido M, Vecsey L. Geophysical applications of multidimensional filtering with wavelets. *Pure and Applied Geophysics* 2002; **159**(10):2285–2309.
3. Beazley DM, Lomdahl PS, Gronbech-Jansen N, Giles R, Tomayo P. Parallel algorithms for short range molecular dynamics. *Annual Reviews of Computational Physics III*. World Scientific: Singapore, 1996; 119–175.
4. Kadau K, Germann TC, Lomdahl PS, Holian BL. Microscopic view of structural phase transitions induced by shock waves. *Science* 2002; **296**:1681–1684.
5. Bird GA. *Molecular Dynamics and the Direct Simulation of Gas Flow*. Oxford Science Publications: Oxford, 1994.
6. Hoogerbrugge PJ, Koelman JMVA. Simulating microscopic hydrodynamic phenomena with dissipative particle dynamics. *Europhysics Letters* 1992; **19**(3):155–160.
7. Libersky LD, Petschek AG, Carney TC, Hipp JR, Allahdadi FA. High strain Lagrangian hydrodynamics. *Journal of Computational Physics* 1993; **109**(1):67–73.
8. Español P. Fluid particle model. *Physical Review E* 1998; **57**(3):2930–2948.
9. Rothman DH, Zaleski S. *Lattice–Gas Cellular Automata: Simple Models of Complex Hydrodynamics*. Cambridge University Press: Cambridge, 1997.
10. Chopard B, Droz M. *Cellular Automata Modelling of Physical Systems*. Cambridge University Press: Cambridge, 1998.
11. Dzwiniel W, Alda W, Kitowski J, Yuen DA. Using discrete particles as a natural solver in simulating multiple-scale phenomena. *Molecular Simulation* 2000; **25**:6361–6384.
12. Dzwiniel W. Virtual particles and search for global minimum. *Future Generation Computer Systems* 1997; **12**:1–19.
13. Wolfram S. *A New Kind of Science*. Wolfram Media Incorporated, 2002; 1263 pp.
14. Vashishta P, Nakano A. Dynamic fracture analysis. *Computing in Science and Engineering* 1999; **1**(5):20–23.
15. Wylie JJ, Koch DL. Particle clustering due to hydrodynamic interactions. *Physics Fluids* 2000; **12**(5):964–970.
16. Meakin P. *Fractals, Scaling and Growth Far from Equilibrium*. Cambridge University Press: Cambridge, 1998.
17. Dzwiniel W, Yuen DA. Mesoscopic dispersion of colloidal agglomerate in a complex fluid modelled by a hybrid fluid-particle model. *Journal of Colloid Interface Science* 2002; **247**:463–480.
18. Faber V. Clustering and the continuous k-means algorithm. *Los Alamos Science* 1994; **22**:138–149.
19. Xiaowei Xu, Jager J, Kriegel H-P. A fast parallel clustering algorithm for large spatial databases. *Data Mining and Knowledge Discovery* 1999; **3**(3):263–290.



20. Judd D, McKinley PK, Jain AK. Large-scale parallel data clustering. *IEEE Transactions on Pattern Analysis* 1998; **20**(8):871–876.
21. Olson CF. Parallel algorithms for hierarchical clustering. *Parallel Computing* 1995; **21**(8):1313–1325.
22. Friedman JH, Baskett F, Shustek LJ. An algorithm for finding nearest neighbors. *IEEE Transactions on Computers* 1975; **C-24**:1000–1006.
23. Gowda CK, Krishna G. Agglomerative clustering using the concept of nearest neighborhood. *Pattern Recognition* 1978; **10**:105.
24. Hockney RW, Eastwood JW. *Computer Simulation Using Particles*. Institute of Physics Publishing: London, 1981.
25. Andenberg MR. *Clusters Analysis for Applications*. Academic Press: New York, 1973.
26. Jain D, Dubes RC. *Algorithms for Clustering Data. (Advanced Reference Series)*. Prentice-Hall: Englewood Cliffs, NJ, 1988.
27. Theodoris S, Koutroumbas K. *Pattern Recognition*. Academic Press: San Diego, CA, 1998.
28. Ben-Zion Y, Lyakhovksy V. Accelerated seismic release and related aspects of seismicity patterns on earthquake faults. *Pure and Applied Geophysics* 2002; **159**(10). At press.
29. Lyakhovksy V, Ben-Zion Y, Agnon A. Earthquake cycle, fault zones and seismicity patterns in a rheologically layered lithosphere. *Journal of Geophysics Research* 2001; **106**:4103–4120.
30. Wylie JJ, Koch DL. Particle clustering due to hydrodynamic interactions. *Physics Fluids* 2000; **12**(5):964–970.
31. Rundle JB, Klein W, Tiampo K, Gross S. Linear pattern dynamics in nonlinear threshold systems. *Physics Reviews E* 2000; **61**(3):2418–2431.
32. Rundle JB, Gross S, Klein W, Ferguson C, Turcotte DL. The statistical mechanics of earthquakes. *Tectonophysics* 1997; **277**:147–164.
33. Freed AM, Lin J. Delayed triggering of the 1999 Hector Mine earthquake by viscoelastic stress transfer. *Nature* 2001; **411**:180–183.
34. Briceno C, Vivas AK, Calvet N, Hartmann L, Pacheco R, Herrera D, Romero L, Berlind P, Sanchez G, Snyder JA, Andrews P. The CIDA-QUEST large-scale survey of Orion OB1: Evidence for rapid disk dissipation in a dispersed stellar population. *Science* 2001; **291**:93–96.
35. Barnes JE, Hernquist L. Transformation of galaxies. II. Gasdynamics in merging disk galaxies. *Astrophysics Journal* 1996; **471**:115–142.
36. Su MS, Chou Ch. A modified version of the K-means algorithm with a distance based on cluster symmetry. *IEEE Transactions on Pattern Analysis* 2001; **23**:674–680.
37. Dzwinel W. How to make Sammon's mapping useful for multi-dimensional data structures analysis? *Pattern Recognition* 1994; **27**(7):949–959.
38. Dzwinel W, Blasiak J. Method of particles in visual clustering of multi-dimensional and large data sets. *Future Generation Computers Systems* 1999; **15**:365–379.
39. TGS—visual concepts for the new millennium. <http://www.tgs.com>.
40. Gast AP, Russel WB. Simple ordering in complex fluids. *Physics Today* 1998; **51**(12):24–30.
41. Dzwinel W, Yuen DA. Matching macroscopic properties of binary fluid to the interactions of dissipative particle dynamics. *International Journal of Modern Physics C* 2000; **11**(1):1–25.
42. Dzwinel W, Yuen DA. A two-level, discrete-particle approach for simulating ordered colloidal structures. *Journal of Colloid Interface Science* 2000; **225**(1):179–190.
43. Boryczko K, Dzwinel W, Yuen DA. Parallel implementation of the fluid particle model for simulating complex fluids in the mesoscale. *Concurrency and Computation: Practice and Experience* 2002; **14**:137–161.
44. Young WR, Roberts AJ, Stuhne G. Reproductive pair correlations and the clustering of organisms. *Nature* 2001; **412**:328–331.
45. Fox G, Sung-Hoon Ko, Pierce M, Balsoy O, Kim J, Lee S, Kim K, Oh S, Rao X, Varank M, Bulut H, Gunduz G, Qiu X, Pallickara S, Uyar A, Youn Ch. Grid services for earthquake science. *Concurrency and Computation: Practice and Experience* 2002; **14**(6/7):371–393.
46. Fox G. Web services and peer-to-peer technologies for the Grid. *ICCS*, Amsterdam, 24 April 2002. <http://grids.ucs.indiana.edu/ptliupages/publications/presentations/iccsapril02.ppt>.
47. Fox G. Collaboration and Web services, 4 April 2002. <http://grids.ucs.indiana.edu/ptliupages/publications/presentations/collabwncsaapril02.ppt>.