Asociación Argentina
de Mecánica Computacional

Mecánica Computacional Vol XXXV, págs. 455-466 (artículo completo)
Martín I. Idiart, Ana E. Scarabino y Mario A. Storti (Eds.)
La Plata, 7-10 Noviembre 2017

IMPROVEMENTS IN SPEED AND SCALABILITY OF A DEM CODE

Rainald Löhner[a] **and Franco Perazzo**[b]

[a]*Center for Computational Fluid Dynamics, George Mason University, M.S. 6A2, Fairfax, VA 22030-4444, USA, rlohner@gmu.edu, https://cfd.gmu.edu/~rlohner*

[b]*AULA DIMEC-CIMNE, Dept. of Mechanical Engineering, Universidad Técnica Federico Santa Maria Valparaiso, Chile, franco.perazzo@usm.cl, http://www.usm.cl*

Keywords: Discrete Element Method, Spatial Search Algorithms, Bins, Contact Algorithm.

Abstract. A number of near-optimal techniques were implemented to reduce computing times for the Discrete Element Method (DEM) code named DESOL. Among these, the following showed the largest improvements: multilevel bins, periodic rebuild, trimming and Symmetric Multiprocessor (SMP) parallelization. These improvements have led to Central Processing Unit (CPU) reduction of the order of 1:3-1:5 on scalar machines, while also showing excellent scalability up to the point of memory saturation, which on current Intel Xeon processors occurs at approximately 8 cores for double precision and 16 cores for single precision.

1 INTRODUCTION

Many engineering processes involve the transport of materials that, at the engineering scale of interest, have to be considered discontinuous. Examples include loading, crushing, smelting and abrasion of materials. One very common way the describe these materials is via DEM. The material is treated as a system of individual particles, which interact with each other via bonding, contact, friction and damping, producing the resulting motion of the system. Since the seminal publication of Cundall and Strack (1979), the method has developed rapidly and its field of applications has expanded considerably (Cleary and Sawley, 2002; Latham and Munjiza, 2004; Radeke et al., 2010; Govender et al., 2014, 2015).

In DEM techniques, the overwhelming computational cost is due to contact search (the actual contact force calculation is insignificant). This can be particularly burdensome if large variations of sphere/element sizes and large variations of wall face sizes are present in the problem. Faced with stagnating clock cycles on current hardware, a decision was made to revisit these parts of the DEM code DESOL in order to see whether better algorithms could not lead to significant improvements in performance. At the same time, current and emerging hardware has forced developers to vectorize as much as possible all CPU-intensive operations. This led to a review of all CPU-intensive modules in DESOL.

In the sequel, we describe the DEM used, the techniques implemented to reduce contact search and the scalability obtained on modern CPU architectures.

2 DISCRETE ELEMENT MODEL

2.1 Basic Equations

The equations describing the motion of the discrete elements are the well known Newton-Euler equations (Sommerfeld, 1976) for the balance of forces and moments:

$$m_i \ddot{\mathbf{x}}_i = \mathbf{F}_i \ , \tag{2.1.1}$$

$$\boldsymbol{\Theta}_i \dot{\boldsymbol{\omega}}_i - \boldsymbol{\omega}_i \times (\mathbf{I}_i \cdot \boldsymbol{\omega}_i) = \mathbf{M}_i \ , \tag{2.1.2}$$

where $m_i, \mathbf{x}_i, \mathbf{F}_i, \boldsymbol{\Theta}_i, \boldsymbol{\omega}_i, \mathbf{M}_i$ denote, respectively, the mass, position of the center of mass, force vector, tensor of inertial moments, rotation vector, and moment vector of the discrete element/ object i. The mass and moment tensors are dependent on the spatial distribution of the density ρ, and are obtained in the usual way from:

$$m = \int_\Omega dm = \int_\Omega \rho d\Omega \ , \ \ I_{ij} = \int_\Omega r_0^i r_0^j \rho d\Omega \ , \tag{2.1.3}$$

$$\boldsymbol{\Theta} = tr(\mathbf{I}) \cdot \mathbf{1} - \mathbf{I} = \left\{ \begin{array}{ccc} I_{yy} + I_{zz} & -I_{xy} & -I_{xz} \\ -I_{xy} & I_{xx} + I_{zz} & -I_{yz} \\ -I_{xz} & -I_{yz} & I_{xx} + I_{yy} \end{array} \right\} \ , \tag{2.1.4}$$

where $\mathbf{r}_0$ denotes the vector from the center of mass to any given position of the body. The forces and moments are given by the usual body forces (gravity, magnetic) and - most importantly from computational considerations - contact forces with other discrete elements/ objects or walls. With the additional identity:

$$\mathbf{v} = \dot{\mathbf{x}} \ , \tag{2.1.5}$$

these equations may be recast as a coupled system of the form:

$$\dot{\mathbf{u}}_i = \mathbf{r}_i \ , \tag{2.1.6}$$

where the vector of unknowns is:

$$\mathbf{u} = (\mathbf{x}, \mathbf{v}, \boldsymbol{\omega}) \ , \tag{2.1.7}$$

and $\mathbf{r}$ from Eqns.(2.1.1-2.1.4).

2.2 Time Integration

Equations 2.1.6 are integrated in time either using an explicit, low-storage Runge-Kutta scheme of the form:

$$\Delta\mathbf{u}^{n+i} = \frac{1}{s+1-i}\,\Delta t\,\mathbf{r}(\mathbf{u}^n + \Delta\mathbf{u}^{n+i-1}) \ , \ \ i = 1, s \ , \ \ \Delta\mathbf{u}^0 = 0 \ , \tag{2.2.1}$$

or the usual 4-stage $(1/6, 1/3, 1/3, 1/6)$ Runge-Kutta scheme (Butcher, 2003). Here $n, i, \Delta t, \mathbf{r}, \Delta\mathbf{u}$ denote the timestep number, the iteration number within a timestep, the timestep size, vector of right-hand sides and increments of unknowns.

2.3 Contact Forces

Both linear and nonlinear (Hertz) dash-pot models are used to estimate the contact forces between particles. With the notation of Figure 2.3.1, the forces are separated into normal and tangential.

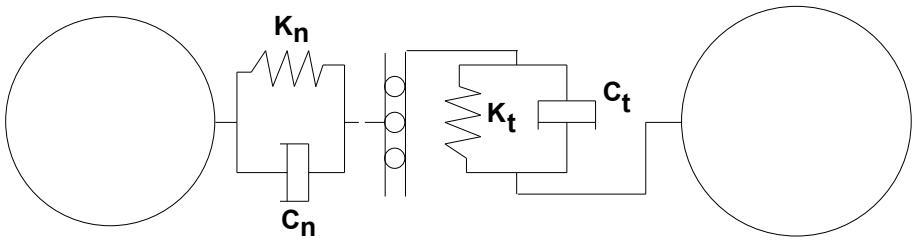

Figure 2.3.1: Dashpot Model Used for Contact Forces

For the linear case, the normal force is given by:

$$\mathbf{f}_n = k_n\delta_{ij}\mathbf{n}_{ij} - c_n(\mathbf{v}_{ij}\cdot\mathbf{n}_{ij})\mathbf{n}_{ij} \ , \tag{2.3.1}$$

where $\delta_{ij}, \mathbf{n}_{ij}, \mathbf{v}_{ij}, k_n, c_n$ denote the penetration depth, contact normal, relative translational velocity, spring stiffness and viscous damping coefficient. For the case of spherical particles of radius r these reduce to:

$$\delta_{ij} = max(0, r_i + r_j - |\mathbf{x}_i - \mathbf{x}_j|) \ , \tag{2.3.2}$$

$$\mathbf{n}_{ij} = \frac{\mathbf{x}_i - \mathbf{x}_j}{|\mathbf{x}_i - \mathbf{x}_j|} \ , \tag{2.3.3}$$

$$\mathbf{v}_{ij} = \mathbf{v}_i - \mathbf{v}_j \ , \tag{2.3.4}$$

The tangential force is given by the friction forces resulting from particles sliding past each other while in contact. The relative velocity is given by:

$$\mathbf{v}_{ij}^t = \mathbf{v}_{ij}^c - (\mathbf{n}_{ij} \cdot \mathbf{v}_{ij}^c)\mathbf{n}_{ij} \ , \tag{2.3.5}$$

where the relative velocity at the contact point takes the rotation of the particles into consideration:

$$\mathbf{v}_{ij}^c = \mathbf{v}_{ij} + (\mathbf{x}_{ij}^c - \mathbf{x}_i) \times \boldsymbol{\omega}_i - (\mathbf{x}_{ij}^c - \mathbf{x}_j) \times \boldsymbol{\omega}_j \ , \tag{2.3.6}$$

and $\mathbf{x}_{ij}^c$ denotes the contact point.
For the nonlinear (Hertz), the equivalent elastic modulus, radius and mass are given by:

$$E_{eq} = \left(\frac{1}{E_i} + \frac{1}{E_j} \right)^{-1} \ ; \ r_{eq} = \left(\frac{1}{r_i} + \frac{1}{r_j} \right)^{-1} \ , \ m_{eq} = \left(\frac{1}{m_i} + \frac{1}{m_j} \right)^{-1} \ , \tag{2.3.7}$$

The normal stiffness is obtained from:

$$k_n = \frac{4}{3} E_{eq} \sqrt{r_{eq}} \ , \tag{2.3.8}$$

yielding the normal elastic force:

$$\mathbf{f}_{ne} = k_n \delta^{1.5} \mathbf{n} \ , \tag{2.3.9}$$

The normal damping factor is given by:

$$c_n = \alpha_n \sqrt{m_{eq} k_n \sqrt{\delta}} \ , \tag{2.3.10}$$

resulting in the normal damping force:

$$\mathbf{f}_{nd} = c_n (\mathbf{v}_{ij} \cdot \mathbf{n}) \mathbf{n} \ . \tag{2.3.11}$$

3 DEM-DEM CONTACT DETECTION ALGORITHMS

As in most DEM techniques, the overwhelming computational cost is due to contact search (the actual contact force calculation is insignificant). Therefore, care has to be taken in the design of fast contact detection algorithms. In the present case, lists of possible contacts are built every so often. These pairs, which will be denoted as edges in the sequel, are tested at the beginning of each timestep in order to see if an actual contact exists. Given that this test is cheap, a compromise has to be reached between the extent of possible contacts (memory, tests if contacting) and the cost of rebuilding the possible contact pairs. The larger the list of possible contacts, the less often the list needs to be rebuilt.

3.1 Bin Data Structure

The potential contact edges are built using a bin data structure (see Figure 3.1.1).

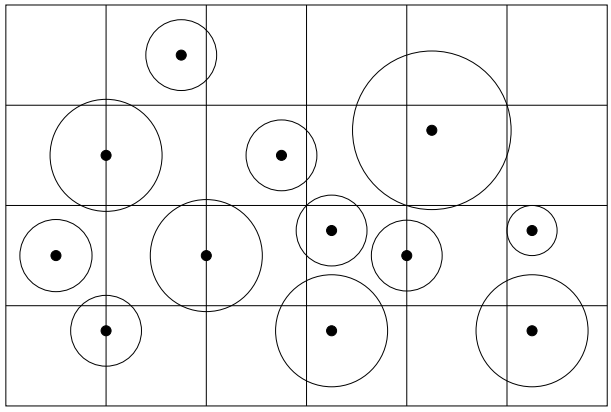

Figure 3.1.1: Bin Data Structure for Points (2-D)

In a first step, the bounding box (maximum extent) of the centroids of the discrete elements is obtained, yielding $x_{min}, x_{max}, y_{min}, y_{max}, z_{min}, z_{max}$. This bounding box is then subdivided into a cartesian mesh of $N_b = N_x \times N_y \times N_z$ bins (voxels) of size $\Delta x, \Delta y, \Delta z$. Given a discrete element with centroid at x, y, z, the bin into which it falls is obtained from:

$$i_x = \frac{x - x_{min}}{\Delta x} \ , \ i_y = \frac{y - y_{min}}{\Delta y} \ , \ i_z = \frac{z - z_{min}}{\Delta z} \ , \tag{3.1.1}$$

$$ibin = N_x \cdot N_y \cdot (i_z - 1) + N_x \cdot (i_y - 1) + i_x \ . \tag{3.1.2}$$

The bin is stored as a linked list with two arrays. The first array stores the discrete elements while the second (of size N_b) stores the locations of the discrete elements in each bin in the first array. These arrays are built in two (scalar) passes over the discrete elements and one pass over the bins. In the first pass over the discrete elements, the storage locations for each bin are added up. In a pass over the bins, the final storage locations are obtained. This is then followed by a second pass over the discrete elements where the storage into bins is completed. Thus, the computational complexity of this step is $O(2N_{DEM} + N_b)$, where N_{DEM} denotes the number of DEMs.

3.2 Near Neighbour Detection

For each centroid x, y, z the search region for nearest neighbours is given by:

$$x - d_s \leq x \leq x + d_s \ , \ y - d_s \leq y \leq y + d_s \ , \ z - d_s \leq z \leq z + d_s \ . \tag{3.2.1}$$

As stated before, a compromise has to be found between the storage of many potential contact pairs and the need to rebuild the contact pairs. If d_s is chosen too large, memory and potential contact detection costs increase while the need to redo the lists decreases. The opposite is true if d_s is kept too small. As contact between two spheres occurs if:

$$\delta_{ij} = r_i + r_j - |\mathbf{x}_i - \mathbf{x}_j| \leq 0 \ , \tag{3.2.2}$$

a safe choice is to take:

$$d_s = 2r + d_{safe} + d_{rb} \ , \tag{3.2.3}$$

where r denotes the radius of the particle, d_{safe} is a safety distance based on how far the particle can move in a given number of timesteps, and d_{rb} the distance particles can move before the bin is rebuilt. Typical choices are:

$$d_{safe} = n\Delta t|\mathbf{v}|_{max} \ , \ \ d_{rb} = 2r_{min} \ , \tag{3.2.4}$$

where $\Delta t, |\mathbf{v}|_{max}, r_{min}$ denote the timestep, maximum velocity of the particles and minimim radius, and $n = O(5)$. Given the bounds of the search region (Eqn.(3.2.1)), the minimum and maximum extent of bins in the x, y, z directions are obtained from Eqn.(3.1.1). The centroids stored in these bins then yield the list of potential contact pairs.
As the pairs of potential contact pairs are built, a pair i, j may appear twice, having been obtained for both the near neighbour region of sphere i and sphere j. Therefore, one has to decide which pair to keep. In the present case, the sphere with the larger radius is kept as the first one. This implies that when searching for potential near neighbours of sphere i, all spheres of radius larger than r_i can be discarded.

3.3 Trimming The Near Neighbour List

The search of neighbours in the region given by Eqn.(3.2.1) can yield a much larger number of neighbours than those that can actually contact. As an example consider the situation shown in Figure 3.3.1.

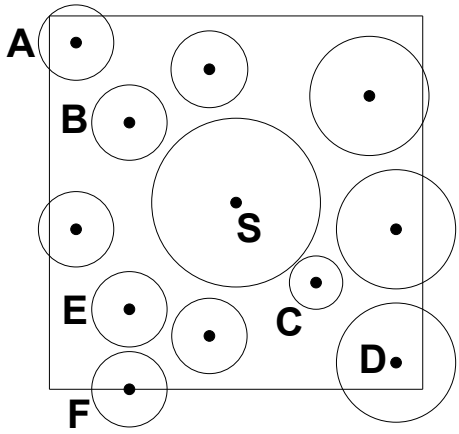

Figure 3.3.1: Search Region for Close Neighbours (2-D)

The potential contact neighbours in the search region of discrete element S include A, F and D. But these are shielded from S by B, E and C. In order to avoid considering these redundant discrete elements as neighbours of S, the list of neighbours found from Eqn.(3.2.1) is ordered according to the distance from S. For each of the neighbours, a check is made whether other neighbours are blocked by this neighbour.

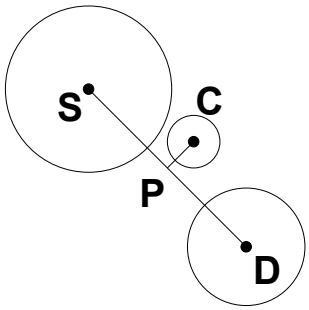

Figure 3.3.2: Neighbour Blocking Criteria

Using the notation in Figure 3.3.2, blocking is assumed to occur if the following conditions are satisfied:

$$(\mathbf{x}_D - \mathbf{x}_S) \cdot (\mathbf{x}_C - \mathbf{x}_S) > 0 \ , \tag{3.3.1.a}$$

$$|\mathbf{x}_D - \mathbf{x}_S| > |\mathbf{x}_P - \mathbf{x}_S| \ , \tag{3.3.1.b}$$

$$|\mathbf{x}_P - \mathbf{x}_C| < \alpha \cdot r_C \ , \tag{3.3.1.c}$$

with $\alpha = O(1)$. The trimming of neighbours is not very important for cases with a uniform distribution of sphere radii. However, if sphere radii vary greatly, trimming greatly reduces the number of potential neighbours, leading to considerable gains in CPU performance.

3.4 SMP Parallelization

While the placing of discrete elements into bins and the rebuilding of the linked lists is a mostly scalar operation, the search for potential DEM-DEM contact pairs can easily be parallelized for shared memory parallel machines such as common multicore Intel Xeon, KNL or AMD processors. In the present case, the lists of nearest neighbours are obtained independently in groups of 2056 discrete elements at a time. Scalability is excellent up to the point of memory saturation, which on current Intel Xeon processors occurs at approximately 8 cores for double precision and 16 cores for single precision.

4 DEM-WALL CONTACT DETECTION ALGORITHMS

DEM-wall contact detection is the second most CPU-intensive operation of DEM codes after DEM-DEM contact detection. The wall description is always assumed to be given in the form of a triangulation that may move, deform, or change topology in time. As before, a list of potential contact pairs (DEM-Triangle) is kept and rebuilt on a regular basis.

4.1 Multi-Level Bin Data Structure

The potential wall contact edges are built using a bin data structure for the faces (see Figure 4.1.1). In a first step, the bounding boxes of the boundary faces is obtained. These bounding boxes are then placed into a bin. However, unlike the centroids (points) of discrete elements, boundary faces can cover several bins, so they must be stored multiple times. Therefore, a balance has to be struck between bins that are too fine (storing a face in many bins) or too coarse (storing many faces in a bin) (see Figure 4.1.2).

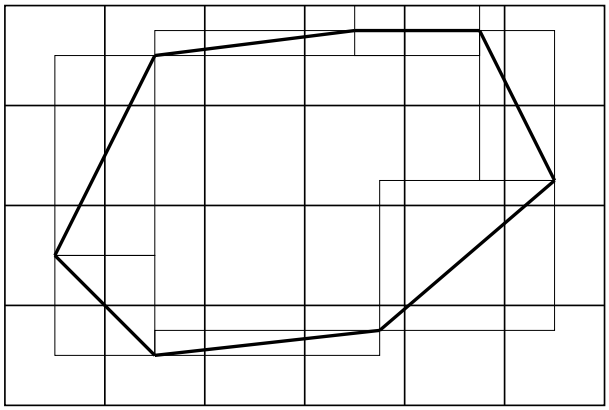

Figure 4.1.1: Bin Data Structure for Faces (2-D)

The optimal bin size is of the order of the size (e.g. maximum side length or bounding box) of the face being stored. For many realistic geometries, face size can vary by orders of magnitude. A single bin size would therefore negatively affect performance.

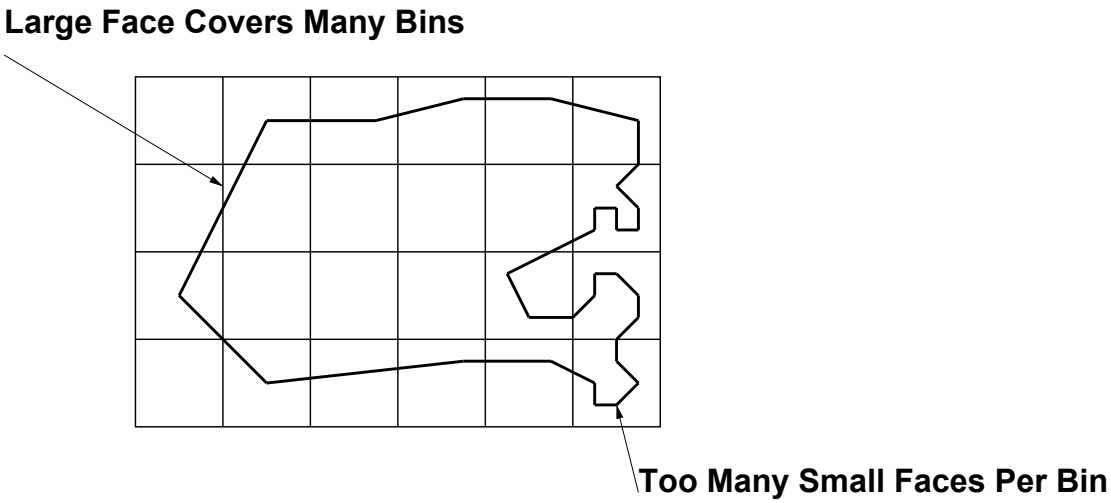

Figure 4.1.2: Small/Large Faces Placed in Bin (2-D)

The solution is to use a hierarchy of bins (Teschener et al., 2003; Sommerfeld, 1976). Assume a series of N_b bins with bin sizes increasing from Δx_{min} via increase factors of $c_{incr} = O(2-5)$ to $\Delta x_{max} = c_{incr}^{N_b} \Delta x_{min}$. Each face is then placed in the bin most suitable for its size. Compared to the single bin, the memory allocation is optimal. Every face is stored in only a few bins, all of them optimally suited to its size. The extra bin placement information amounts to a very modest increase in memory, as the number of bins in the coarser bins decreases rapidly (at least an order of magnitude for each level of coarsening). For cases with very small faces, the finest bin may be limited by the available memory. In this case, all the faces with sizes smaller or equal to the size of this finest bin will be stored in it. This implies no extra storage, but more faces per bin and hence more data to be processed when detecting contact. The computational complexity to build the bin data structure is $O(2N_{TRI} + N_b)$, where N_{TRI} denotes the number of triangles defining walls.

4.2 Near Neighbour Detection

For each centroid x, y, z the search region for nearest neighbours is given by Eqns(3.2.1,3.2.4). This search region yields the extent of bins that need to be interrogated for faces. Each of the multilevel bins is interrogated, and all faces stored in these bins are retrieved for subsequent processing.

4.3 Trimming The Near Face List

The search of close faces in the region given by Eqn.(3.2.1) may yield faces that are stored multiple times in neighbouring bins. This list is trimmed using hashing techniques (Schornbaum, 2009).

4.4 SMP Parallelization

While the placing of wall faces into bins and the rebuilding of the linked lists is a mostly scalar operation, the search for potential DEM-wall contact pairs can easily be parallelized, as the same way that in the search for potential DEM-DEM contact.

5 TIMINGS

Typical timings are reported for the mill case shown in Figure 5.1. The rotating surface walls were described by `nface=796` surface triangles. The total number of discrete spherical elements was `npart=74,189`, of which `npart=74,111` had radii in the range $0.0095 - 0.0105\ m$ (representing the material being crushed) and `npart=78` had radii in the range $0.095 - 0.105\ m$ (representing the steel balls used for crushing). The total integration time was $T = 10\ sec$ (one complete rotation of the mill), which required `ntime=86,577` timesteps. The timings obtained, as well as the resulting speedups (both for the overall runs and the detailed parts of the integration in time) are shown Figures 5.2-5.4. Note that the speedup for all parts suffers due to memory bandwidth constraints, but is particularly noticeable for: a) the time integration, which has a very small computational intensity, and b) for the DEM-Wall forces due to the very small number of surface triangles, leading to memory contention.

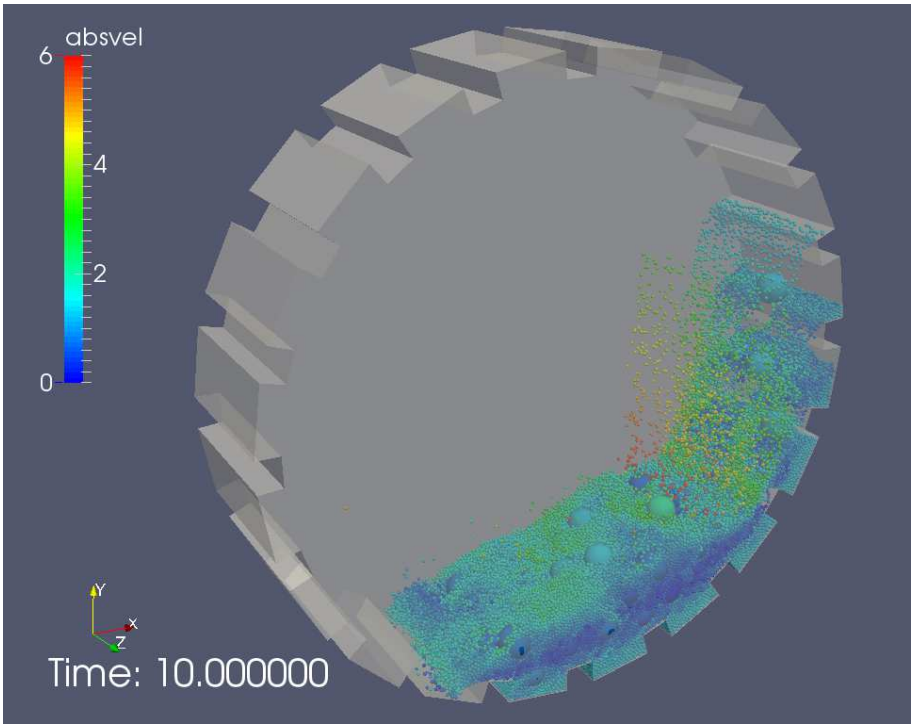

Figure 5.1: SAG Mill (Approximate Diameter: 10 m)

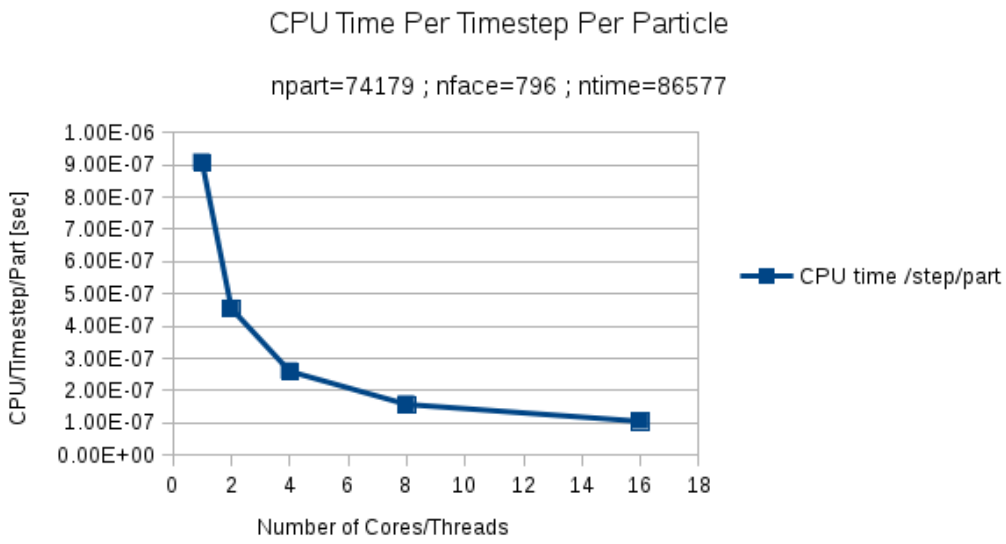

Figure 5.2: Mill: CPU/Particle/Timestep

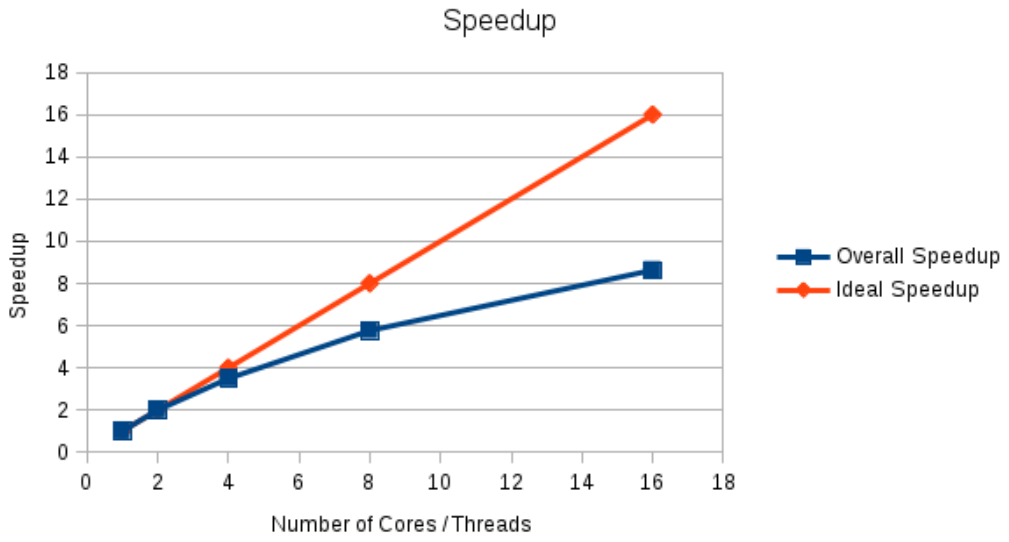

Figure 5.3: Mill: Speedups Achieved

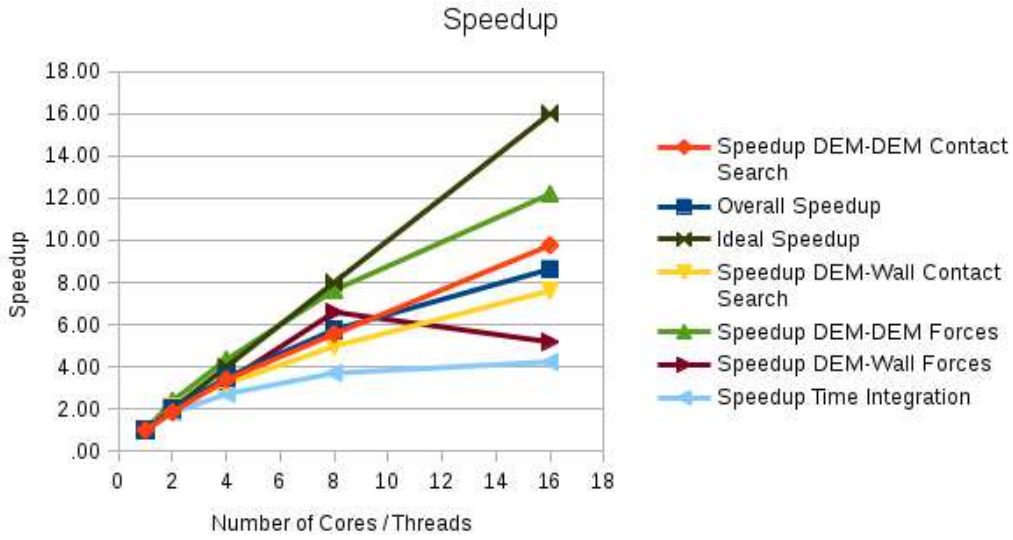

Figure 5.4: Mill: Speedups Achieved (Detailed)

The entire run, including input, initialization and all output diagnostics, took approximately 11 minutes to complete on a 16 core Xeon machine running in single precision (which is entirely sufficient for accuracy purposes), showing that applications of this kind have now become feasible on laptops.

6 CONCLUSIONS AND OUTLOOK

A number of near-optimal techniques were implemented to reduce computing times for the DEM code DESOL. Among these, the following showed the largest improvements:
a) **Multilevel Bins**: The use of multilevel bins for spatial objects allows an optimal placement matching object and bin size. This circumvents the usual disadvatage of bins: too many objects in a bin or an object in too many bins.

b) **Periodic Rebuild**: The list of possible contact pairs is rebuilt every so often. These pairs are tested at the beginning of each timestep in order to see if an actual contact exists. Given that this test is cheap, a compromise has to be reached between the extent of possible contacts (memory, test if contacting) and the cost of rebuilding the possible contact pairs. The larger the list of possible contacts, the less often the list needs to be rebuilt.
c) **Trimming**: The search for possible contact pairs can yield a much larger number of neighbours than those that can actually contact. Particularly when large variations of sphere/ element sizes are present, intermediate particles may 'shield' others, thus making contact impossible. Therefore, the list is examined further and trimmed accordingly.
d) **SMP Parallelization**: The search for potential contact pairs can easily be parallelized for shared memory parallel machines such as common multicore Intel Xeon, KNL or AMD processors. In the present case, the lists of nearest neighbours are obtained independently in groups of 2056 discrete elements at a time.

These improvements have led to CPU reduction of the order of 1:3-1:5 on scalar machines, while also showing excellent scalability up to the point of memory saturation, which on current Intel Xeon processors occurs at approximately 8 cores for double precision and 16 cores for single precision.

7 ACKNOWLEDGMENTS

The authors are grateful for the support and funding provided by the Directorate of Research, Innovation and Postgraduate Studies, and the Department of Mechanical Engineering of the UTFSM, in order to carry out the present work. An important part of it was developed during the stay of the first author in the DIMEC-CIMNE clasroom of the UTFSM.

REFERENCES

Butcher J. *Numerical Methods for Ordinary Differential Equations*. j. Wiley, 2003.

Cleary P. and Sawley M. Dem modelling of industrial granular flows: 3d case studies and the effect of particle shape on hopper discharge. *Appl. Math. Model*, 26:89–111, 2002.

Cundall P. and Strack O. A discrete numerical model for granular assemblies. *Geotechique*, 29:47–65, 1979.

Govender N., Rajamani R., Kok S., and Wilke D. Discrete element simulation of mill charge in 3d using the blaze-dem gpu framework. *Minerals Engineering*, 79:152–168, 2015.

Govender N., Wilke D., Kok S., and R. E. Development of a convex polyhedral discrete element simulation framework for nvidia kepler based gpu. *J. Comp. Appl. Math.*, 270:386–400, 2014.

Latham J. and Munjiza A. The modelling of particle system with real shapes. *Phil. Trans. R. Soc. Lon. Ser. A Math. Phys. Eng. Sci.*, 362:1953–1972, 2004.

Radeke C., Glasser B., and Khinast J. Large-scale powder mixer simulation using massively parallel gpu architectures. *Chem. Eng. Sci.*, 65:6435–6442, 2010.

Schornbaum F. Hierarchical hash grids for coarse collision detection. Technical Report, Rep. Institut for Informatics, Friedrich-Alexander-Univ., Erlangen, Germany, 2009.

Sommerfeld A. *Vorlesungen Ãijber Theoretische Mechanik*. Verlag Harri Deutsch, Frankfurt, 1976.

Teschener M., Heidelberger B., Mueller M., Pomeranets D., and Gross M. Optimized spatial hashing for collision detection of deformable objects. *Proc. Vision, Modeling, Visualization*, pages 47–54, 2003.