DEVELOPMENT OF A PARALLEL DISCRETE ELEMENT SIMULATION SYSTEM FOR STUDYING SELF-COMPACTING CONCRETE BEHAVIOR

Li-Shin Lin[1], Shang-Hsien Hsieh[2], and Wei-Tze Chang[3]

ABSTRACT

This paper reports an effort on development of a parallel discrete particle simulation system, named Iris, for studying Self-Compacting Concrete (SCC) behavior. Iris is developed based on an in-house C++ discrete objects simulation framework, name VEDO (VErsatile Discrete Objects framework). VEDO is designed to be capable of handling simultaneously discrete objects of various shapes and various mechanisms of interactions between discrete objects and it has great flexibility in facilitating additions of new discrete object shapes and solution algorithms for discrete object interactions. In this work, objects of the spherical shape are used to model both the aggregates and mortars of the SCC flow, while objects of other shapes are used only for modeling immobile boundaries or obstacles in the simulation. The development of VEDO employs the object-oriented technology with design patterns. The development of Iris extends the object-oriented framework of VEDO for parallel computing. A graph partitioning approach is developed to achieve parallelism and dynamic load balancing among processors for Iris. In addition, parallel simulation of an example SCC V-funnel experiment is carried out on a shared-memory IBM P690 computer to investigate and demonstrate the parallel speed-up and efficiency of Iris.

KEY WORDS

discrete element simulation, parallel computing, graph partitioning, domain decomposition.

INTRODUCTION

Discrete Element Method (DEM) was first proposed by Cundall in 1971. It treats material as a finite set of discrete elements, and simulates the macroscopic behavior of the material through microscopic interactions among the discrete elements. Although DEM was initially developed for simulation of progressive movement of blocky rocks, it has now been widely applied in many engineering fields dealing with non-homogeneous and discontinuous materials.

[1] Formerly M.S. Student, Department of Civil Engineering, National Taiwan University, Taipei 10617, Taiwan, wokerla@caece.net

[2] Professor, Department of Civil Engineering, National Taiwan University, Taipei 10617, Taiwan (also, Head of Information Technology & Management Division, National Center for Research on Earthquake Engineering), Phone +886/2-3366-4313, FAX +886/2-2368-8213, shhsieh@ntu.edu.tw

[3] Ph.D. Candidate, Department of Civil Engineering, National Taiwan University, Taipei 10617, Taiwan, Phone +886/2-3366-4342 ext. 12, FAX 886/2-2392-5290, d92521003@ntu.edu.tw

The discrete element simulation often requires use of very large number of discrete elements and very small time step (for the reason of numerical stability), leading to considerable computational time for each simulation. To alleviate this problem, some researchers have employed parallel computing techniques to speed up the simulation (e.g., Dowding, *et al.*, 1999; Tang and Zurawski, 1993; Schafer, *et al.*, 2004).

In this work, a parallel discrete element simulation system for studying Self-Compacting Concrete (SCC) behavior, called Iris, has been developed based on parallelized version of an in-house C++ discrete objects simulation framework, named VEDO (VErsatile Discrete Objects framework) (Yang and Hsieh, 2005). A graph partitioning strategy has been developed and implemented in Iris for achieving parallelism and dynamic load balancing among processors. In the SCC flow simulation, the aggregates and mortars of SCC are modeled using discrete objects of the spherical shape.

The remaining of this paper is organized as follows. First, the VEDO framework is reviewed. Second, the parallel discrete element simulation strategy developed in this work is described. Then, the extension of VEDO for parallel computing and the development of Iris are discussed. Also, the parallel performance of the proposed strategy is investigated through parallel simulation of an example SCC experiment on IBM P-690, a shared-memory high-performance computer. Finally, some conclusions are drawn for this research.

VERSATILE DISCRETE OBJECTS FRAMEWORK

To be capable of performing simulations with elements of various shapes and various inter-element mechanisms, an in-house discrete element simulation framework named VEDO (VErsatile Discrete Objects framework) has been developed by Yang and Hsieh (2005). Unlike the traditional framework focusing mainly on the modeling of discrete objects, the design of VEDO focuses more on the modeling of "interaction" between discrete objects. As shown in Figure 1, the *DiscreteObject*, *ContactDetector*, and *ImpactSolver* classes are designed to model discrete objects of various shapes, various methods for detecting contact between two discrete objects of the same or different types, and various solvers for computing impact forces between two discrete objects, respectively. The *Interaction* class is at the center of the design of the VEDO framework and is responsible for binding the appropriate *ContactDetector* and *ImpactSolver* objects to any two interacting discrete objects. The *DOContainer* and *InacContainer* classes are designed to construct the run-time data structure for managing all discrete elements and keeping track of inter-element interactions, respectively, during the simulation. Besides, three auxiliary classes are designed to help carrying out the simulation. The *SimMediator* class manages the simulation procedure and all required run-time data structures. The *Assembler* class helps on the decision of appropriate *ContactDector* and *ImpactSolver* bindings for the *Interaction* class. The *DOWorld* class records the system information for each time step.

In addition, based on VEDO, an application program for sequential three-dimensional discrete element simulation, named Knight&Anne, has been developed. Knight&Anne can handle a wide range of interaction mechanisms between discrete objects of several different shapes, including rectangular plate, cylinder, spherical particle, and polyhedron (Chen, *et al.*, 2004; Yang, 2004).

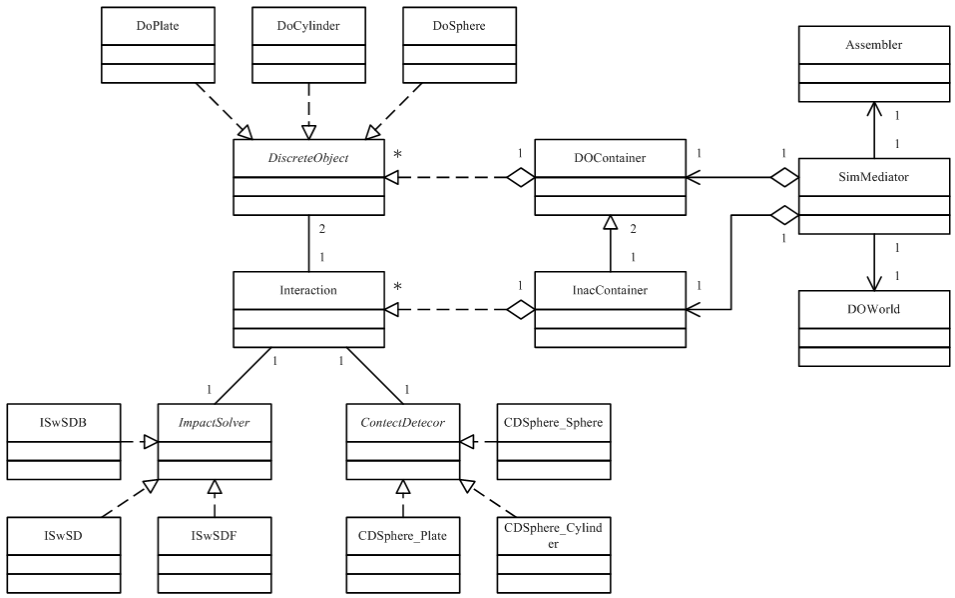


Figure 1: The VErsatile Discrete Objects framework

PARALLEL COMPUTING STRATEGY

Contact detection and impact solution are the two core computation steps in the typical discrete element simulation procedure as shown in Figure 2. The computing time required by these two steps often sums up to more than 90% of the total computing time for the simulation of large-scaled problems. This means that the parallelization of these computations should be the major focus of any parallel computing strategy in order to effectively achieve high parallel speed-up.

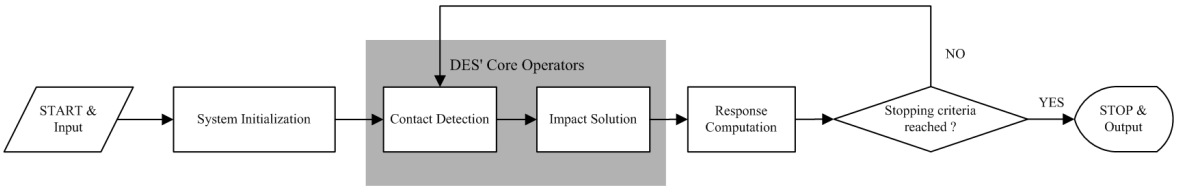


Figure 2: The Typical Procedure for Discrete Element Method

The parallel computing strategy proposed in this work employs graph partitioning to achieve balance of computational loads among processors and minimization of inter-process message passing. The key idea in this graph partitioning approach is to map the problem of partitioning discrete objects (or elements) for parallel processors form the geometric space to the graph space (Lin, 2005). During the mapping, a discrete object becomes a vertex of the graph and any two discrete objects that may be in contact (i.e., if there exists an interaction

object for them) are connected by an edge in the graph. The weight of each vertex is defined as the number of edges it has (see Figure 3a). Then, the approach takes advantage of the graph partitioning technique to decompose the graph into a set of sub-graphs with equal total vertex weights and minimum total edge-cuts. Finally, the corresponding discrete objects of the vertices in a sub-graph are distributed to a processor in parallel computing.

After the vertices (i.e., discrete elements) are partitioned, the edges (i.e., interaction objects) lying across the boundaries of adjacent sub-graphs should be appropriately assigned to a unique sub-graph. This research employs the following approach to reduce the amount of exchanged information between processors:

(a) Let us denote that the symbol *B(i)* represents the processor where the element e_i is distributed; *I(i,j)* represents the interaction object between elements e_i and e_j, and *L(i,B(j))* is the number of interactions between the element e_i and the processor *B(j)*.

(b) In the condition of $B(i) \neq B(j)$, meaning that the elements e_i and e_j are distributed to different processors, if $L(i,B(j)) \leq L(j,B(i))$, the interaction object *I(i,j)* is assigned to processor *B(i)*; Otherwise, it is assigned to processor *B(j)*. For the example shown in Figure 3b, all of the interaction objects will be assigned to processor P_2. In this way, processors P_1 and P_2 only need to exchange the information associated with the element e_1.

Also, for each pair of vertices sharing a cross-boundary edge, the weight of each of the vertices is incremented by one to account for the workload for inter-process communication. Because the above adjustment on the weights of vertices results in imbalance of the total vertex weights among sub-graphs, an iterative re-partitioning scheme is employed for improving the load balance among sub-graphs as follows:

(a) To estimate the imbalance of computational workloads among processors, an index, *U*, is first defined as

$$U = \frac{W_{\max} - \overline{W}}{\overline{W}}, \quad \text{where } W_{\max} = \max_i(W_i) \quad \overline{W} = \frac{\sum W_i}{NP} \tag{1}$$

In Equation (1), W_i denotes the total vertex weight of the sub-graph (or processor) *i* and *NP* denotes the total number of sub-graphs. With the assumption that W_i is proportional to the workload of the processor *i*, the larger the value of *U* is, the higher the workload imbalance among processors is.

(b) If the value of *U* is smaller than the value of a user-defined imbalance tolerance, U_0, the graph partitions obtained are used to distribute elements among processors. Otherwise, the graph is re-partitioned and the steps for assigning of the cross-boundary interaction objects (or edges) and adjusting the weights of the associated vertices as discussed earlier are repeated.

The parallel computing procedure of the proposed graph partitioning approach is summarized as follows:

(a) The status of all discrete objects, such as their positions, velocities, accelerations, and the field forces acting on them, is initialized.

(b) Contact between any discrete object pair is detected and, if contact occurs, an interaction object is constructed for the object pair.

(c) The corresponding graph of the discrete element set is constructed and partitioned using a graph partitioning method. Each of the cross-boundary interaction elements is assigned to a unique processor and the weights of its connected vertices are adjusted.

(d) Go to (c) unless $U<U_0$ or the maximum number of iterations is reached.

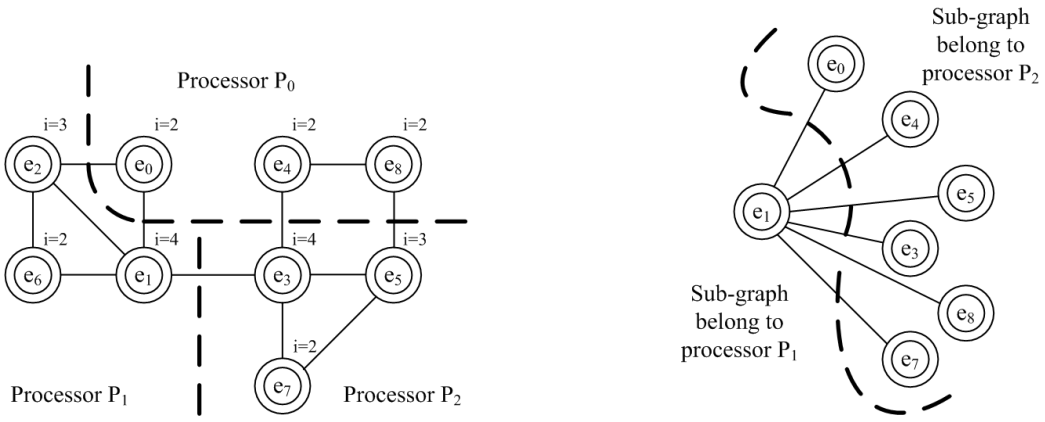
(e) Partitioned discrete objects and interaction objects are distributed among processors.

(f) Each processor performs the contact detection and impact solution computations associated with the interaction objects assigned to the processor and their associated discrete object pairs. If $B(i) \neq B(j)$ and assume that *I(i,j)* is assigned to *B(i)*, message passing is required for the element e_j to send its status information to *B(i)* and for the interaction object *I(i,j)* to send the calculated impact force to *B(j)*.

(g) Update the status of each discrete object.

(h) Go to Step (f) unless re-partitioning and re-distribution are needed (in this research, the re-partitioning and re-distribution are performed every 1,000 time steps).

(i) All information of discrete objects is collected to processor P_0 and merged. Then, go to Step (b).



(a) vertices, e, and their weights, i (b) Sub-graph

Figure 3: The Graph Partitioning Strategy

EXTENSION OF VEDO FOR PARALLEL COMPUTING

To support the parallel strategy already discussed in the previous section, VEDO is extended to include a series of *Consultant* classes, as shown in Figure 4 (Lin, 2005). The *Consultant* classes serves as computing-strategy consultants for the *SimMediator* class. For the parallel

computing, the *ParallelConsultant* class implements the strategy for distributing interaction objects and discrete objects among processors, including the message-passing mechanisms. Even for the sequential computing, the *Consultant* classes can implement some smarter contact detection strategies, for example, for efficiency improvement.

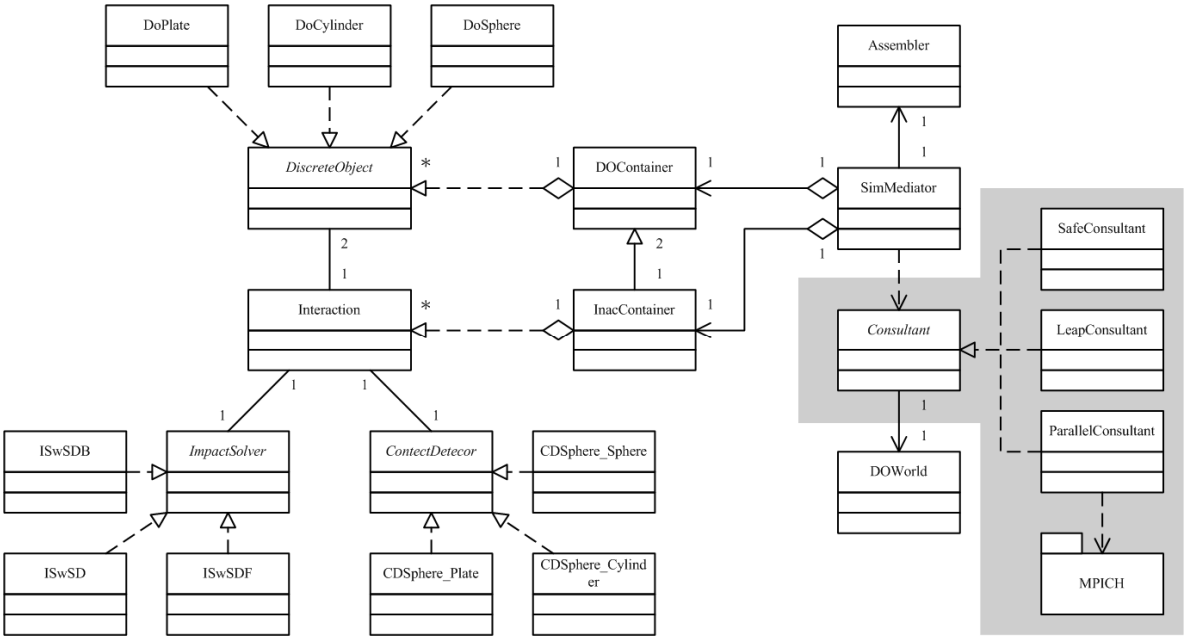


Figure 4: The Extended VEDO Framework for Parallel Computing

For this research, the parallel discrete element simulation program, named Iris, is developed based on the extended version of VEDO (Lin, 2005). Iris inherits the C++ object-oriented technology and design patterns employed by VEDO and supports MS-Windows 32bits and UNIX-like operational system, such as IBM AIX, FreeBSD, and Redhat LINUX. For graph partitioning in Iris, the ParMetis library (Karypis, *et al.*, 1998) based on the parallel multilevel k-way graph-partitioning algorithm is employed. In addition, the MPICH library, based on the MPI technology (Message Passing Interface Forum, 1994), is employed to achieve message passing among processors in both distributed and shared memory computing environment.

NUMERICAL EXAMPLE

SCC has become one of the most important materials in today's construction industry because of its advantages in improving quality, productivity, and working environment of construction. The high workability is the major characteristic that differentiates SCC from conventional concrete. High workability of concrete means good passing ability and segregation resistance. Therefore, the placement of SCC does not need any additional,

external vibration or compaction to avoid problem of segregation or blockage in the formwork.

It is not an easy but important task to estimate the workability of a designed SCC. Several types of laboratory tests, such as slump flow test, L-box test, and V-funnel test (JSCE, 1999), have been proposed to help understand the workability of SCC. However, laboratory tests are often time consuming and costly. Therefore, computer simulations become a potential alternative to cope with this problem.

Due to the non-homogeneous and discontinuous properties of concrete materials plus large translation and rotation of aggregates in fresh concrete flow, DEM becomes a better choice than the continuum mechanics based numerical approaches, e.g., the finite element method, to model and simulate the dynamic behavior of SCC granular flow. Several researchers (e.g., Noor, 2000; Liu, 2002; Yang and Hsieh, 2002) have employed two-dimensional or three-dimensional DEM to simulate the SCC flow behavior. The aggregates and cement of SCC are modeled using circular or spherical particles of different sizes and material properties, and the macroscopic behavior of SCC is simulated through the microscopic interactions between these particles.

The V-funnel test is one of the standard SCC experiments (JSCE, 1999). The V-funnel model consists of a V-shaped container with an opening in the bottom (Figure 5). In the test, the time needed for a container of SCC to flow out the bottom opening is measured to estimate the deformation capacity and the narrow-opening-passing ability of SCC.

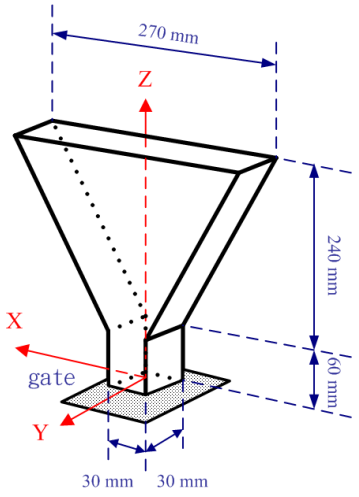


Figure 5: The Reduced-size V-Funnel Model

This research performs parallel simulations of an example SCC V-funnel experiment with the reduced-size V-funnel shown in Figure 5 to investigate the speed-up and efficiency of the proposed parallel simulation strategy. The simulation involves 11,000 mortar particles of 5 mm in diameter. The time step used is 2 x 10^{-6} seconds and the total number of simulation time steps is 500,000. All of the simulations are performed on an IBM P-690 Series Server with up to 14 processors and 122 gigabyte shared memory.

Table 1 and Figure 6 show the good performance obtained by the proposed parallel strategy implemented in Iris. For up to 8 processors, the efficiency can still be maintained at 80%.

Table 1: The Speed-up and Efficiency of the Parallel Simulations

Number of Processors	Computing Time (hours)	Speed-Up	Efficiency
1	39.30	---	---
2	20.30	1.94	97 %
4	11.12	3.53	88 %
6	7.96	4.94	82 %
8	6.13	6.41	80 %

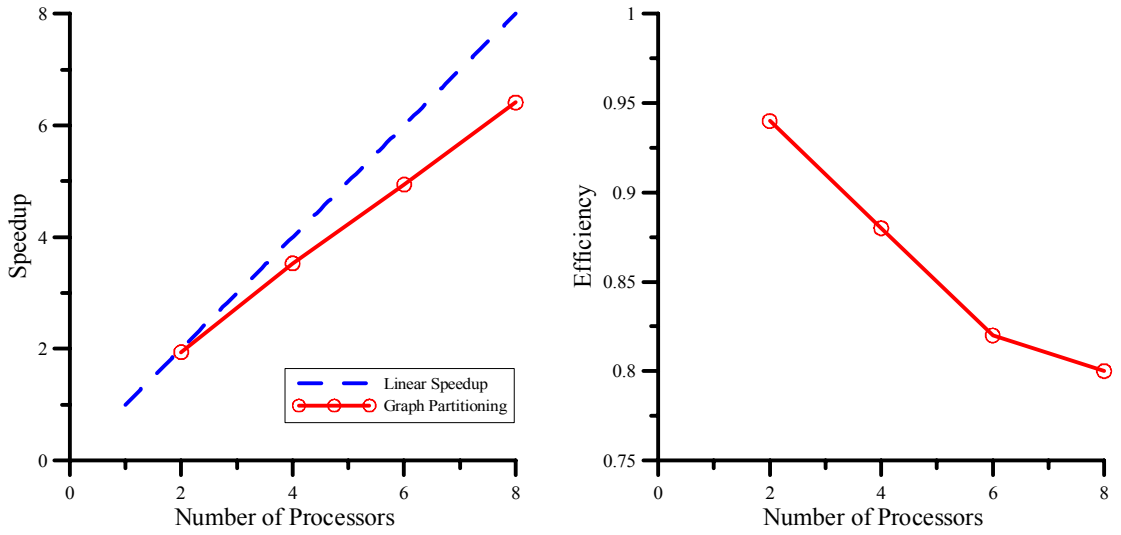


Figure 6: Speed-up and Efficiency of the Parallel Simulations

CONCLUSIONS

To speed up the discrete element simulation of SCC flow behavior that is often computationally intensive, a parallel computing strategy based on graph partitioning has been developed in this research. An in-house discrete object simulation framework has been extended to implement the parallel strategy. Based on the preliminary investigation on parallel discrete element simulation of an SCC experiment using IBM P-690, it is found that the parallel strategy developed can achieve good parallel efficiency on shared-memory multicomputers.

ACKNOWLEDGMENTS

The financial support from the National Science Council of Taiwan (Grant No. NSC 94-2211-E-002-053) and the use of the IBM P-690 high-performance computing server at National Center for Research on Earthquake Engineering are gratefully acknowledged. The authors would like to thank Prof. Yin-Wen Chan of National Taiwan University, for his helpful discussions and suggestions on the simulation of SCC behavior.

REFERENCES

Chen, C. S., Hsieh, S. H., Yang C. T. and Chou, C. C. (2004). "Contact Detection Analysis of Polyhedral Discrete Elements Using Linear and Quadratic Programming," *Proceedings of the 17th KKCNN Symposium on Civil Engineering*, Thailand, December 13-15, 2004, 231-236.

Cundall, P. A. (1971). "A Computer Model for Simulating Progressive, Large Scale Movement in Blocky Rock System," *Proceedings of the ISRM Symposium on Rock Fracture*, Nancy, France, Vol. 1, Paper II-8, 129-136.

Dowding, C. H., Dmytryshyn, O. and Belytschko, T. B. (1999). "Parallel Processing for a Discrete Element Program," *Computers and Geotechnics*, 25(4), 281-285.

JSCE (1999). "Standard Test Methods for Self-Compacting Concrete," *Recommendations for Self-Compacting Concrete*, T. Uomoto and K. Ozawa (Editors), Concrete Engineering Series, No. 31, Japan Society of Civil Engineers, 50-77.

Schloegel, K., Karypis, G., and Kumar, V. (2002). "Parallel Static and Dynamic Multi-constraint Graph Partitioning," *Concurrency and Computation: Practice and Experience*, 14(3), 219-240. (available at http://glaros.dtc.umn.edu/gkhome/metis/parmetis/overview/)

Karypis, G., Schloegel, K., and Kumar, V. (1998). "ParMETIS2.0: Parallel Graph Partitioning and Sparse Matrix Ordering Library," *Technical Report*, Department of Computer Science, University of Minnesota, Minneapolis, MN. (available at http://glaros.dtc.umn.edu/gkhome/metis/parmetis/overview)

Lin, L. S. (2005). "*Parallel DEM Simulation Strategy*," M.S. Thesis, Department of Civil Engineering, National Taiwan University, Taipei, Taiwan. (in Chinese)

Message Passing Interface Forum (1994). "MPI: A Message-Passing Interface Standard," *International Journal of Supercomputer Applications*, 8(3-4), 159-416.

Noor, M. A. (2000). "*Three-Dimensional Discrete Element Simulation of Flowable Concrete*," Ph.D. Dissertation, Department of Civil Engineering, Graduate School of the University of Tokyo, Tokyo, Japan.

Ouchi, M. (1998). "History of Development and Applications of Self-Compacting Concrete in Japan," *Proceedings of the International Workshop on Self-Compacting Concrete*, Kochi University of Technology, Kochi, Japan, August 23-26, 1998, 1-10.

Schafer, B. C., Quigley, S. F., and Chan, A. H. C. (2004). "Acceleration of the Discrete Element Method (DEM) On A Reconfigurable Co-processor," *Computers and Structures*, 82(20-21) 1707-1718.

Tang, S. K. and Zurawski, R. (1993). "C-Linda Implementations of the Distinct Element Model," *Proceedings of the 5th IEEE Symposium on Parallel and Distributed Processing (SPDP'93)*, Dallas, Texas, USA, December 1-4, 1993, 538-545.

Yang, C. T. (2004). "*A Versatile Discrete Objects Simulation System*," Ph.D. Dissertation, Department of Civil Engineering, National Taiwan University, Taipei, Taiwan.

Yang, C. T. and Hsieh, S. H. (2002), "Software Development for Particle Simulation of Self-Compacting Concrete," *Proceedings of the 15th KKCNN Symposium on Civil Engineering*, Singapore, December 19-20, 2002, S280-S284.

Yang, C. T. and Hsieh, S.H. (2005). "An Object-Oriented Framework for Versatile Discrete Objects Simulation Using Design Patterns," *Computational Mechanics*, 36(2) 85-99.