# ParaPART: Parallel Mesh Partitioning Tool for Distributed Systems

Jian Chen Valerie E. Taylor
Department of Electrical and Computer Engineering
Northwestern University
Evanston, IL 60208
{jchen, taylor}@ece.nwu.edu

**Abstract.** In this paper, we present ParaPART, a parallel version of a mesh partitioning tool, called PART, for distributed systems. PART takes into consideration the heterogeneities in processor performance, network performance and application computational complexities to achieve a balanced estimate of execution time across the processors in the distributed system. Simulated annealing is used in PART to perform the backtracking search for desired partitions. ParaPART significantly improves performance of PART by using the asynchronous multiple Markov chain approach of parallel simulated annealing. ParaPART is used to partition six irregular meshes into 8, 16, and 100 subdomains using up to 64 client processors on an IBM SP2 machine. The results show superlinear speedup in most cases and nearly perfect speedup for the rest. Using the partitions from ParaPART, we ran an explicit, 2-D finite element code on two geographically distributed IBM SP machines. Results indicate that ParaPART produces results consistent with PART. The execution time was reduced by 12% as compared with partitions that consider only processor performance; this is significant given the theoretical upper bound of 15% reduction.

## 1 Introduction

High performance distributed computing is rapidly becoming a reality. Nationwide high speed networks such as vBNS [16] are becoming widely available to interconnect high speed computers at different sites. Projects such as Globus [9] and Legion [13] are developing software infrastructure for computations that integrate distributed computational and informational resources. Further, a National Technology Grid is being constructed to provide access to advanced computational capabilities regardless of the location of the users and resources [10]. The distributed nature of this kind of computing environment calls for consideration of heterogeneities in computational and communication resources when partitioning meshes. In this paper, we present ParaPART, a parallel version of a mesh partitioning tool, called PART, for distributed systems.

In [5], we presented an automatic mesh partitioning tool, PART, for distributed systems. PART takes into consideration the heterogeneities in processor performance, network performance and application computational complexities to achieve a balanced estimate of execution time across the processors in a distributed system. We addressed four major issues on mesh partitioning for distributed systems in [6] and demonstrated significant increase in efficiency by using partitions from PART. In this paper, we present ParaPART, a parallel version of PART, which significantly improves performance. Simulated annealing is used in PART to perform the backtracking search for desired partitions. However, it is well known that simulated annealing is computationally intensive. In ParaPART, we use the asynchronous multiple Markov chain approach of parallel simulated annealing [14] for which the simulated annealing process is considered a Markov chain. A server processor keeps track of the best solution and each client processor performs an independent simulated annealing with a different seed. Periodically, the client processors exchange solutions with the server and either continues with its own solution or with a better solution obtained from the server.

ParaPART is used to partition six irregular meshes into 8, 16, and 100 subdomains using up to 64 client processors on an IBM SP2 machine. The results show superlinear speedup in most cases and nearly perfect speedup for the rest. Using mesh partitions from ParaPART, we ran an explicit, 2-D finite element code on two geographically distributed IBM SP machines. We used Globus [9] software for communication between the two SPs. The results indicate that ParaPART produces partitions consistent with PART. The execution time was reduced by 12% as compared with partitions that consider only processor performance; this is significant given the theoretical upper bound of 15% reduction.

The remainder of the paper is organized as follows: Section 2 is related work. Section 3 describes ParaPART. Section 4 is experimental results. Section 5 gives conclusion.

## 2 Related Work

Currently, there are five widely used mesh partitioning tools; the tools are METIS [17,18], HARP [26], CHACO [15], TOP/DOMDEC [25], and JOSTLE [28]. These tools use one or more of the following partitioning algorithms: Greedy [8], Kernighan-Lin [19], Recursive Coordinate Bisection [2], Recursive Spectral Bisection [24], Recursive Inertia Partition [23], Recursive Graph Bisection [11], and Simulated Annealing. All of the aforementioned tools have the goal of generating equal partitions with minimum cuts. Some tools, such as METIS, considers processor performance heterogeneity with an easy modification to the input file. To achieve good performance on a distributed system, however, the heterogeneities in both network and processor performance must be taken into consideration. In addition, computational complexity of the application as well as difference element types in the mesh must also be considered. PART differs from existing tools in that it takes into consideration all of these heterogeneities.

PART uses simulated annealing to partition the mesh. Figure 1 shows the serial version of simulated annealing algorithm. This algorithm uses the Metropolis criteria (line 8 to 13 in Figure 1) to accept or reject moves. The moves that reduce the cost function are accepted; the moves that increase the cost function

```
1. Get an initial solution $S$
2. Get an initial temperature $T > 0$
3. While stopping criteria not met {
4.   $M$ = number of moves per temperature
5.   for $m = 1$ to $M$ {
6.     Generate a random move
7.     Evaluate changes in cost function: $\Delta E$
8      if ($\Delta E < 0$) {
9        accept this move, and update solution $S$
10.    } else {
11.      accept with probability $P = e^{-\frac{\Delta E}{T}}$
12.      update solution $S$ if accepted
13.    }
14.  } /*end for loop*/
15.  Set $T = rT$ (reduce temperature)
16.} /*end while loop */
17.Return $S$
```

**Fig. 1.** Simulated annealing.

may be accepted with probability $e^{-\frac{\Delta E}{T}}$, thereby avoiding being trapped in local minima. This probability decreases when the temperature is lowered. Simulated Annealing is computationally intensive, therefore, a parallel version of simulated annealing is used for ParaPART. There are three major classes of parallel simulated annealing [12]: serial-like [20, 27], parallel moves [1], and multiple Markov chains [3, 14, 22]. Serial like algorithms essentially break up each move into subtasks and parallelize the subtasks (parallelizing line 6 and 7 in Figure 1). For the parallel moves algorithms, each processor generates and evaluates moves independently; cost function calculation may be inaccurate since processors are not aware of moves by other processors. Periodic updates are normally used to address the effect of cost function error. Parallel moves algorithms essentially parallelize the for loop (line 5 to 14) in Figure 1. For the multiple Markov chains algorithm, multiple simulated annealing processes are started on various processors with *different* random seeds. Processors periodically exchange solutions and the best is selected and given to all the processors to continue their annealing processes. In [3], multiple Markov chain approach was shown to be most effective for VLSI cell placement. For this reason, ParaPART uses the multiple Markov chain approach. The details of the use of the multiple Markov chain approach is described below.

## 3 ParaPART Description

ParaPART is a parallel version of PART, which entails three steps. The first step generates a coarse partitioning for the distributed systems. Each group gets a subdomain that is proportional to its number of processors, the performance of the processors, and the computational complexity of the application. A group is a collection of processors with the same performance and interconnected via a local network. Hence computational cost is balanced across all the groups. In the second step, the subdomain that is assigned to each group from step 1 is partitioned among its processors. This coarsening step is performed using METIS. Within each group, parallel simulated annealing is used to balance the execution time by considering variance in network performance. Processors that communicate remotely with processors in other groups will have reduced computational load to compensate for the longer communication time. The third step addresses the global optimization, taking into consideration differences in the local interconnect performance of various groups. Again, the goal is to minimize the variance of the execution time across all processors. In this step, elements on the boundaries of group partitions are moved according to the execution time variance between neighboring groups. For the case when a significant number of elements is moved between the groups in step 3, the second step is executed again to equalize the execution time in a group given the new computational load. For the experiments conducted thus far, step 3 has not been needed. A more detailed description of PART is in [5].

ParaPART replaces the simulated annealing in PART with the asynchronous multiple Markov chain approach of parallel simulated annealing [14]. Given $P$ processors, a straightforward implementation of the multiple Markov chain approach would be initiating simulated annealing on each of the $P$ processors with a different seed. Each processor performs moves independently and then finally the best solution from those computed by all processors is selected. In this approach, however, simulated annealing is essentially performed $P$ times which may result a better solution but not speedup.

To achieve speedup, $P$ processors perform an independent simulated annealing with a different seed, but each processor performs only $N/P$ moves ($N$ is the number of moves performed by the simulated annealing at each temperature). Processors exchange solutions at the end of each temperature. The exchange of data occurs synchronously or asynchronously. In the synchronous multiple Markov chain approach, the processors periodically exchange solutions with each other. In the asynchronous approach, the client processors exchange solutions with a server processor. It has been reported that the synchronous approach is more easily trapped in a local optima than the asynchronous [14], therefore ParaPART uses the asynchronous approach. During solution exchange, if the client solution is better, the server processor is updated with the better solution; if the server solution is better, the client gets updated with the better solution and continue from there. Processor exchanges solution with the server processor at the end of each temperature.

To ensure that each subdomain is connected, we check for disconnected components at the end of ParaPART. If any subdomain has disconnected components, the parallel simulated annealing is repeated with a different random seed. This process continues until there is no disconnected subdomains or the number

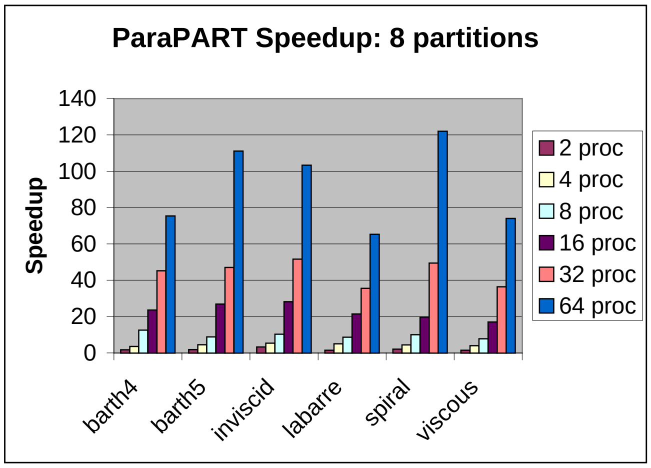

**Fig. 2.** ParaPART speedup for 8 partitions.

of trials exceed three times. A warning message is given in the output if there are disconnected subdomains.

## 4 Experimental Results

In this section, we present the results from two different experiments. The first experiment focuses on the speedup of ParaPART. The second experiment focuses on the quality of the partitions generated with ParaPART.

### 4.1 Speedup Results

**Table 1.** ParaPART execution time (seconds): 8 partitions.

| # of proc. | barth4 | barth5 | inviscid | labarre | spiral | viscous |
|---|---|---|---|---|---|---|
| 1 proc | 158.4 | 244.4 | 248.0 | 163.3 | 183.0 | 214.6 |
| 2 proc | 91.2 | 133.5 | 76.2 | 112.3 | 88.5 | 142.2 |
| 4 proc | 44.4 | 54.7 | 46.6 | 32.4 | 41.2 | 53.7 |
| 8 proc | 12.6 | 27.8 | 24.2 | 18.8 | 18.3 | 27.6 |
| 16 proc | 6.7 | 9.1 | 8.8 | 7.6 | 9.3 | 12.6 |
| 32 proc | 3.5 | 5.2 | 4.8 | 4.6 | 3.7 | 5.9 |
| 64 proc | 2.1 | 2.2 | 2.4 | 2.5 | 1.5 | 2.9 |

ParaPART is used to partition six 2D irregular meshes with triangular elements: barth4 (11451 elem.), barth5 (30269 elem.), inviscid (6928 elem.), labarre (14971

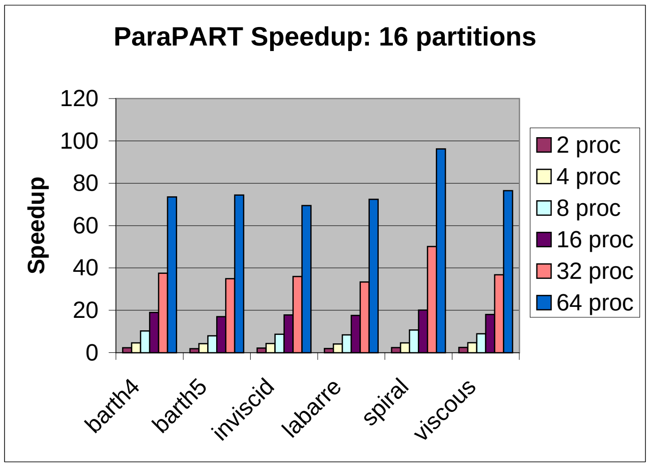

**Fig. 3.** ParaPART speedup for 16 partitions.

elem.), spiral (1992 elem.), and viscous (18369 elem.). The cost function of ParaPART is the estimate of the execution time of the WHAMS2D. The running time of using ParaPART to partition the six irregular meshes into 8, 16, and 100 subdomains are given in Table 1, 2, and 3, respectively. It is assumed that the subdomains will be executed on a distributed system consisting of two IBM SPs, with equal number of processors but different processor performance. Further, the machines are interconnected via vBNS for which the performance of the network is given in Table 4 (discussed in Section 4.2). In each table, column 1 is the number of client processors used by ParaPART, and columns 2 to 6 are the running time of ParaPART in seconds for the different meshes. The solution quality of using two or more client processors is within 5% of that of using one client processor. In this case, the solution quality is the estimate of the execution time of WHAMS2D.

**Table 2.** ParaPART execution time (seconds): 16 partitions.

| # of proc. | barth4 | barth5 | inviscid | labarre | spiral | viscous |
|---|---|---|---|---|---|---|
| 1 proc | 772.4 | 781.6 | 729.6 | 717.0 | 615.9 | 749.6 |
| 2 proc | 340.2 | 416.5 | 340.5 | 375.6 | 260.1 | 355.8 |
| 4 proc | 168.4 | 183.5 | 168.5 | 176.3 | 135.1 | 160.9 |
| 8 proc | 75.7 | 98.2 | 83.8 | 85.7 | 57.7 | 84.2 |
| 16 proc | 40.7 | 46.0 | 41.0 | 40.8 | 30.7 | 41.7 |
| 32 proc | 20.6 | 22.4 | 20.3 | 21.5 | 12.3 | 20.5 |
| 64 proc | 10.5 | 10.6 | 10.6 | 9.9 | 6.4 | 9.8 |

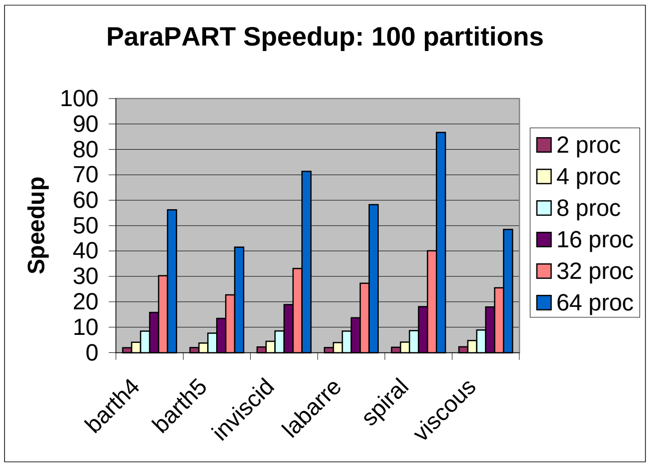

**Fig. 4.** ParaPART speedup for 100 partitions.

**Table 3.** ParaPART execution time (seconds): 100 partitions.

| # of proc. | barth4 | barth5 | inviscid | labarre | spiral | viscous |
|---|---|---|---|---|---|---|
| 1 proc | 16212.5 | 17688.9 | 13759.2 | 16953.1 | 5433.4 | 18993.9 |
| 2 proc | 8603.3 | 8997.9 | 6142.1 | 8626.6 | 2607.8 | 8260.7 |
| 4 proc | 3982.2 | 4666.4 | 3082.3 | 4273.5 | 1304.4 | 3974.0 |
| 8 proc | 1910.5 | 2318.6 | 1610.7 | 2007.8 | 630.3 | 2140.2 |
| 16 proc | 1025.9 | 1310.7 | 728.1 | 1234.7 | 299.9 | 1056.4 |
| 32 proc | 535.4 | 777.9 | 415.6 | 620.8 | 135.5 | 744.3 |
| 64 proc | 288.3 | 426.3 | 192.8 | 291.2 | 62.7 | 391.7 |

Figure 2, 3, and 4 are graphical representations of the speedup of ParaPART relative to one client processor. The figures show that when the meshes are partitioned into 8 or 16 subdomains, superlinear speedup occurs in all cases when the number of client processors is more than 4. When the meshes are partitioned into 100 subdomains, superlinear speedup occurs only in the cases of two smallest meshes, spiral and inviscid. Other cases show slightly less than perfect speedup. This superlinear speedup is attributed to the use of multiple client processors conducting a search, for which all the processors benefit from the results. Once a good solution is found by any one of the clients, this information is given to other clients quickly, thereby reducing the effort of continuing to search for a solution. The superlinear speedup results are consistent with that reported in [21].

### 4.2 Quality of Partition

Our testbed includes two IBM SP machines, one located at Argonne National Laboratory (ANL) and the other located at the San Diego Supercomputing Center (SDSC). These two machines are connected by vBNS (very high speed Backbone Network Service). We used Globus [9] software for communication between the two SPs. Macro benchmarks were used to determine the network and processor performance. The results of the network performance analysis are given in Table 4. Further, experiments were conducted to determine that the SDSC SP processors nodes were 1.6 times as fast as the ANL ones. Our target application is an explicit, 2-D finite element code, WHAMS2D [4]. The computational complexity of WHAMS2D is linear with the size of the problem.

**Table 4.** Values of $\alpha$ and $\beta$ for the different networks.

| | | |
|---|---|---|
| ANL SP Vulcan Switch | $\alpha_1 = 0.00069$s | $\beta_1 = 0.000093$s/KB |
| SDSC SP Vulcan Switch | $\alpha_2 = 0.00068$s | $\beta_2 = 0.000090$s/KB |
| vBNS | $\alpha_3 = 0.059$s | $\beta_3 = 0.0023$ s/KB |

Using the partitions generated by ParaPART as input, the WHAMS2D application is executed on the ANL and SDSC IBM SPs. We used 4 processors from ANL SP and 4 processor from SDSC SP due to limitations of coscheduling of resources on the two sites. The execution of WHAMS2D is given in Table 5. Column one identifies the irregular meshes. Column two is the execution time resulting from the partitions from ParaPART. Column three is the execution time resulting from METIS which takes computing power into consideration (each processor's compute power is used as one of the inputs to the METIS program). The results in Table 5 show that the execution time is reduced by up to 12% as compared to METIS; this is consistent with partitions generated with PART. The reduction of 12% is significant given that we derived in [7] that the best reduction by taking network performance into consideration is 15%. This reduction comes from the fact that even on a high speed network such as the vBNS, the message start up cost on remote and local communication still has a large difference.

## 5 Conclusion

In this paper, we demonstrate the use of a parallel mesh partitioning tool, ParaPART, for distributed systems. The novel part of ParaPART is that it uses the asynchronous multiple Markov chain approach of parallel simulated annealing for mesh partitioning. ParaPART is used to partition six irregular meshes into 8, 16, and 100 subdomains using up to 64 client processors on an IBM SP2 machine.

**Table 5.** Execution time (seconds) using the vBNS on 8 processors: 4 at ANL, 4 at SDSC.

| Mesh | ParaPART | Proc. Perf. (METIS) |
|---|---|---|
| barth5 | 242.0 | 266.0 |
| viscous | 150.0 | 172.0 |
| labarre | 138.0 | 146.0 |
| barth4 | 111.0 | 120.0 |

Results show superlinear speedup in most cases and nearly perfect speedup for the rest.

We used Globus software to run an explicit, 2-D finite element code using mesh partitions from ParaPART. Our testbed includes two geographically distributed IBM SP machines. Experimental results indicate up to 12% reduction in execution time compared with using partitions that only take processor performance into consideration; this is consistent with the results from using PART. This reduction comes from the fact that even on a high speed network such as the vBNS, the message start up cost on remote and local communication still has a large difference.

## References

1. P. Banerjee, M. H. Jones, and J. S. Sargent. Parallel simulated annealing algorithms for cell placement on hypercube multiprocessors. *IEEE Trans. on Parallel and Distributed Systems*, 1(1), January 1990.
2. M. J. Berger and S. H. Bokhari. A partitioning strategy for nonuniform problems on multiprocessors. *IEEE Trans. Comput.*, C-36:570–580, May 1987.
3. J. A. Chandy, S. Kim, B. Ramkumar, S. Parkes, and P. Banerjee. An evaluation of parallel simulated annealing strategies with application to standard cell placement. *IEEE Trans. on Comp. Aid. Design of Int. Cir. and Sys.*, 16(4), April 1997.
4. H. C. Chen, H. Gao, and S. Sarma. WHAMS3D project progress report PR-2. Technical Report 1112, University of Illinois CSRD, 1991.
5. J. Chen and V. E. Taylor. Part: A partitioning tool for efficient use of distributed systems. In *Proceedings of the 11th International Conference on Application Specific Systems, Architectures and Processors*, pages 328–337, Zurich, Switzerland, July 1997.
6. J. Chen and V. E. Taylor. Mesh partitioning for distributed systems. In *Proceedings of the Seventh IEEE International Symposium on High Performance Distributed Computing*, Chicago, IL, July 1998.
7. J. Chen and V. E. Taylor. Mesh partitioning for distributed systems: Exploring optimal number of partitions with local and remote communication. In *Proceedings of Ninth SIAM Conference on Parallel Processing for Scientic Computing*, San Antonio, TX, March 22-24 1999. to appear.
8. C. Farhat. A simple and efficient automatic fem domain decomposer. *Computers and Structures*, 28(5):579–602, 1988.

9. I. Foster, J. Geisler, W. Gropp, N. Karonis, E. Lusk, G. Thiruvathukal, and S. Tuecke. A wide-area implementation of the Message Passing Interface. *Parallel Computing*, 1998. to appear.
10. I. Foster and C. Kesselman. Computational grids. In I. Foster and C. Kesselman, editors, *The Grid: Blueprint for a New Computing Infrastructure*, pages 15–52. Morgan Kaufmann, San Francisco, California, 1986.
11. A. George and J. Liu. *Computer Solution of Large Sparse Positive Definite Systems.* Prentice-Hall, Englewood Cliffs, New Jersey, 1981.
12. D. R. Greening. Parallel simulated annealing techniques. *Physica*, D(42):293–306, 1990.
13. A. S. Grimshaw, W. A. Wulf, and the Legion team. The legion vision of a worldwide virtual computer. *Communications of the ACM*, 40(1), January 1997.
14. G. Hasteer and P. Banerjee. Simulated annealing based parallel state assignment of finite state machines. *Journal of Parallel and Distributed Computing*, 43, 1997.
15. B. Hendrickson and R. Leland. The chaco user's guide. Technical Report SAND93-2339, Sandia National Laboratory, 1993.
16. J. Jamison and R. Wilder. vbns: The internet fast lane for research and education. *IEEE Communications Magazine*, January 1997.
17. G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. Technical report, Department of Computer Science TR95-035, University of Minnesota, 1995.
18. G. Karypis and V. Kumar. Multilevel k-way partitioning scheme for irregular graphs. Technical report, Department of Computer Science TR95-064,University of Minnesota, 1995.
19. B. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal*, 29:291–307, 1970.
20. S. A. Kravitz and R. A. Rutenbar. Placement by simulated annealing on a multiprocessor. *IEEE Trans. Comput. Aided Design*, CAD-6:534–549, July 1987.
21. V. Kumar, A. Grama, A. Gupta, and G. Karypis. *Introduction to Parallel Computing: Design and Analysis of Algorithms.* Benjamin/Cummings, 1994.
22. S.Y. Lee and K.G. Lee. Asynchronous communication of multiple markov chain in parallel simulated annealing. In *Proc. of Int. Conf. Parallel Processing*, volume III, pages 169–176, St. Charles, IL, August 1992.
23. B. Nour-Omid, A. Raefsky, and G. Lyzenga. Solving finite element equations on concurrent computers. In A. K Noor, editor, *Parallel Computations and Their Impact on Mechanics*, pages 209–227. ASME, New York, 1986.
24. H. D. Simon. Partitioning of unstructured problems for parallel processing. *Computing Systems in Engineering*, 2(2/3):135–148, 1991.
25. H. D. Simon and C. Farhat. Top/domdec: a software tool for mesh partitioning and parallel processing. Technical report, Report RNR-93-011, NASA, July 1993.
26. H. D. Simon, A. Sohn, and R. Biswas. Harp: A fast spectral partitioner. In *Proceedings of the Ninth ACM Symposium on Parallel Algorithms and Architectures*, Newport, Rhode Island, June 22-25 1997.
27. A. Sohn. Parallel n-ary speculative computation of simulated annealing. *IEEE Trans. on Parallel and Distributed Systems*, 6(10), October 1995.
28. C. Walshaw, M. Cross, S. Johnson, and M. Everett. *JOSTLE: Partitioning of Unstructured Meshes for Massively Parallel Machines, In N. Satofuka et al, editor, Parallel Computational Fluid Dynamics: New Algorithms and Applications.* Elsevier, Amsterdam, 1995.