

Evaluation of Parallel Domain Decomposition Algorithms

Jesús Antonio Izaguirre Paz

Department of Computer Science and Beckman Institute
University of Illinois at Urbana-Champaign
1304 West Springfield Avenue
Urbana, IL 61801-2987, U.S.A
izaguirr@cs.uiuc.edu

Abstract

In this talk we describe and evaluate several heuristics for the parallel decomposition of data into n processors. This problem is usually cast as a graph partitioning problem, requiring that each processor have equal amount of data, and that inter-processor communication be minimized. Since this problem is NP-complete, several heuristics have been developed: combinatorial, geometric, spectral, multilevel, and combinations of these. We present results of our evaluation of the algorithms found in the software packages Metis version 2.0, Chaco version 2.0, and the geometric mesh partitioning toolbox.

We use the empirical evidence from our evaluation to help decide what algorithm to use depending on load balancing requirements and the number of domains that will make up a partition. Our main results are: a) When the user wants a perfectly balanced partition, the multilevel methods implemented in the software package Metis are the best; b) When the user can allow a bounded load imbalance, Chaco multilevel methods produce better partitions; c) When coordinates are available, geometric methods are usually faster, and produce good quality partitions when linked with local refinement.

The paper presents performance summary tables, a decision chart to select a parallel domain decomposition algorithm, and pointers to the software evaluated here for the benefit of potential users of graph partitioning algorithms for parallel domain decomposition.

1 Introduction

The parallel domain decomposition problem is to partition data or processes among n processors. In graph theory language we can formulate it as follows: Find a

balanced partition of the graph G , whose vertices represent computational processes, and whose edges represent communication due to dependencies among processes. The domains usually represent processors in which the data or the processes will reside. Balanced partitions yield good workload balance among processors; minimal edge-cuts minimize the cost of communication overhead.

Recall that a graph G is a non-empty set V of vertices and a set E of edges, which are unordered pairs of elements of V . A *partition* of G comprises n mutually disjoint subsets whose union is V . We will call these subsets *domains*. A partition is *balanced* if the sizes of the domains are roughly the same. We will refer to the set of edges that connect different domains of a partition as an *edge-cut*. These are edges that, when removed from E , disconnect G into at least n pieces. Thus, the *graph-partitioning* problem is to find a balanced partition such that its edge-cut is of minimal size. Besides parallel domain decomposition, matrix factorization and VLSI layout are usually modelled as a graph-partitioning problem.

The complexity of finding an optimal solution to the graph-partitioning problem is NP-complete [Garey *et al.*, 1976]. Extensive work has led to the development of several classes of heuristics: *combinatorial*, *spectral*, *geometric*, *multilevel*, and combinations of these.

In this work we give empirical evidence to help decide which graph-partitioning algorithms to use on a given problem, depending on the *granularity* of the partition, the degree of *balance* needed, and the availability of geometric information associated with the vertices of a graph. *Granularity* refers to the size of the domains, or, equivalently, to the number of domains n into which G is partitioned. Here, we consider 2 to 8 domains a *coarse* partition; 16 to 64 domains a *medium* partition; and 128 to 1024 domains a *fine* partition. We show that different methods perform better for different granularities. By *degree of balance* of a partition we mean the percentage by which the largest domain exceeds the ideal size: $(|V|/n)$. This measure is not the only one possible, but it makes sense in parallel processing, since the slowest process would slow down the whole computation. It is

the measure used in Metis [Karypis and Kumar, 1995b].

The results presented here are a subset of our master thesis [Izaguirre, 1996]. There we tested over forty methods on ten problems, mostly finite-element meshes. The methods reported here are found in the packages Chaco version 2.0, Metis version 2.0, and the geometric mesh partitioning toolbox. Most experiments were performed on a Sparc/10 workstation. The geometric method was tested on a Silicon Graphics Power-Challenge array.

In this paper we extend the empirical studies of [Gilbert *et al.*, 1994; Chan *et al.*, 1994; Hendrickson and Leland, 1994; Karypis and Kumar, 1995a] in two ways: First, by a more comprehensive inclusion of methods over a common set of problems; second, by a more refined analysis of performance in scenarios requiring different granularity for the partitions and different balance. Even though using slightly imbalanced partitions has been mentioned in the literature, we are quantifying it here for the first time.

We proceed as follows: in sections 2 to 6 we describe each of the heuristics, their parameters, and the variations explored in this study. In section 7 we present and discuss summary results of all the experiments. We also present a decision tree to help in selecting among algorithms. In section 9 we put forth the conclusions of our study, and we give pointers to graph partitioning packages available on the World Wide Web. We refer the reader to the appendices of [Izaguirre, 1996] where we give detailed explanation of the algorithms and results of all the experiments. We hope that our study will be useful to potential users of graph-partitioning algorithms.

2 Combinatorial

Moves vertices from one set to another, producing an overall decrease in the size of the edge-cut. Most of these methods are variations on the KL method of [Kernighan and Lin, 1970]. Particularly important are variants of KL that move vertices in the boundary of a domain only.

3 Spectral

Solves a continuous approximation to the discrete formulation of the graph-partitioning problem. The solution to this approximation is given by the first few eigenvectors of the Laplacian matrix of the graph G . Hence, *eigenvector computations for large, sparse matrices are at the core of these methods*. The theoretical foundations of these methods is found in [Pothen *et al.*, 1990]. We study the implementation in Chaco version 2.0 [Hendrickson and Leland, 1995].

4 Geometric

Limits the search space for a solution by using geometric information associated with the vertices. It imposes a “spatial” bias by looking for edge-cuts orthogo-

nal to some coordinate axes, along which grids are typically elongated and there may be a narrow point to cut. They are most naturally applied to problems from physical domains, such as finite-element meshes. We study the Geometric partitioner of [Gilbert *et al.*, 1994; Teng, 1991] and the Inertial method used in [Hendrickson and Leland, 1995] and described by [Nour-Omid *et al.*, 1986]. Gilbert and Teng’s method projects the grid onto the surface of a sphere, and then finds edge or vertex cuts that lie on a circle or a line on the sphere. It also has been proven to produce good separators for *planar* graphs (those that can be drawn without crossings) which represent VLSI circuit layouts. Geometric methods can be implemented in linear time, and Gilbert and Teng’s method has been parallelized.

5 Geo-spectral

It uses *the components of the first few eigenvectors of the Laplacian matrix of a graph as artificial coordinates*. It can be seen as a geometric method applicable to graphs for which no coordinates are available, or it can be regarded as a more sophisticated spectral method [Chan *et al.*, 1994, p. 6].

6 Multilevel

Multilevel *coarsens a graph* preserving the size of the edge-cut, then it *partitions the coarsened-graph* using a good partitioner, and finally it *uncoarsens* the graph *by projecting the partition* onto finer graphs, locally refining these projections in the process. We test the multilevel methods found in Chaco version 2.0 and Metis version 2.0. See [Hendrickson and Leland, 1993; Karypis and Kumar, 1995a; 1995c]. The advantage of coarsening the graph is twofold: First, the search space for partitions is reduced, making it more likely to succeed in finding a good one. Second, the cost of finding a good partition is lessened by solving the smaller problem. Recall that the number of different bisections of a graph G is $2^{|V(G)|}$. In multilevel methods the overall computational cost of the method is dominated by the cost of local refinement, since the recursion resembles a geometric progression. This differentiates Metis’ from Chaco’s multilevel methods. The former uses boundary local refinement methods, which are cheaper than the latter’s KL method which operates in all vertices of the graph. There are two interfaces for Metis: Pmetis, which uses a multilevel bisection algorithm, and Kmetis, which coarsens the graph to a few thousand vertices, produces a k -way partition using pmetis, and then uses a cheap k -way local refinement. Kmetis produces well balanced partitions, usually better than Pmetis. An advantage of the Chaco multilevel method is that it allows the user to relax the strict balance requirement and produce better quality partitions, though slightly imbalanced.

Label	Description	Parameters default/modified
<i>Cx-ML</i>	Chaco Multilevel	Vertices-to-coarsen-to = 100 Partitioner = Fast-Spectral Local-refinement = KL
C2-ML-100	Chaco v.2.0 Multilevel	
C2-ML-0.49	Chaco v.2.0 Multilevel	KL.Imbalance = 0.011
C2-ML-0.45	Chaco v.2.0 Multilevel	KL.Imbalance = 0.080
Pmetis2-ML	Metis Multilevel	Vertices-to-coarsen-to = 100 Heavy_Match = True Local-refinement = Boundary Greedy Method+KL
Kmetis2-ML	Metis Multilevel	Vertices-to-coarsen-to = 2000 Heavy_Match = True Local-refinement = Boundary Greedy Method
C2-INerti	Chaco v.2.0 Inertial	
Geometric	Circle separator	Number of trials = 50 Code = Matlab <i>meshpart</i> Coordinates = Cartesian
Geo-Spec	Circle separator	Coordinates = Fiedler vector U_2
C2-[F]Spec	Chaco Spectral	Eigensolver = Multilevel/RQI Coarsen-To = 10
C2-KL-FSpec	Same as C2-FSpec, but	local refinement = KL

Table 1: Description of methods tested and their parameters

Package	URL or email; other remarks
Metis 2.0	http://www.cs.umn.edu/karypis/metis
Chaco version 2.0	http://www.cs.sandia.gov/bahendr/partitioning.html . License necessary, at no cost. Contact bah@cs.sandia.gov.
Geometric	ftp://parcftp.xerox.com/pub/gilbert/meshpart.uu Matlab code
Jostle	http://www.gre.ac.uk/c.walshaw/jostle Parallel algorithm
ParMETIS	http://www.cs.umn.edu/karypis/parmetis Contact karypis@cs.umn.edu
Party	hniwww.uni-paderborn.de/graduierete/preis/party.html Library. Contact preis@hni.uni-paderborn.de
Scotch	http://www.labri.u-bordeaux.fr/Equipe/ALiENor/membre/pelegrin/scotch Claims to be faster than Metis
NPAC	Scalable Libraries for Graph Partitioning Contact Sanjay Ranka, ranka@top.cis.syr.edu

Table 2: Graph Partitioning Software for Parallel Domain Decomposition

7 Results

All methods were tested on ten problems. The experiments were run on a Sparc 10/Model 30 with 64 Megabytes of RAM, with the exceptions of Geometric and Geo-spectral, which were run on a Silicon Graphics PowerChallenge array. The operating system was Solaris 2.4. All runs were made with no extra load: the methods tested were the only application running. The codes for all these methods are written in C.

7.1 Problems

The common problem suite consists of finite-element meshes. We had coordinates for all of them and were able to test geometric methods. Recall that $|V|$ is the number of vertices in a graph and $|E|$ is the number of edges. Our problems had $|V|$ rank from 4,720 to 29,681; $|E|$ from 13,722 to 81,795, for two and three dimensional meshes, such as airfoils and triangulations of spheres.

7.2 List of Algorithms

Table 1 summarizes the methods and variations tested in our experiments. A label identifies a method (and the same label is used in all tables). A label comprises one or more of the following acronyms or values:

- Graph partitioning package (e.g., C2 stands for “Chaco Version 2.0”)
- Method class (e.g., ML stands for “Multilevel”)
- Parameter values (e.g., 0.08 for the balance factor q in Chaco)

Every variation of a method differs in at least one parameter. Default values for all other variations within a class of methods are indicated at the beginning of each frame. This parameter is included in the header of the Cx-ML class in the third column, “parameters default/modified.”

7.3 Metrics

We describe the metrics used to test the quality of partitions or of the methods themselves. When graph partitioning is used within the context of parallel processing there is a relation between these metrics and their impact on performance. In the following summary we describe both the metrics and their relevance:

- **Edge-cut size.** It affects the communication overhead generated by a given partition. Larger edge-cuts result in greater inter-processor communication overhead. When we speak about the “quality of a partition,” we mean how close it is to the smallest edge-cut possible, or at least best available to us.
- **Balance of a partition.** It affects the processor utilization efficiency of a partition. An imbalanced partition may have time penalties whose severity will vary across systems and applications.

- **Time to produce a partition.** Some applications can afford more time to compute a good partition (for example when a partition is going to be reused many times). Other applications may settle for a faster partition of lower quality. The time reported in these experiments corresponds to the time for partitioning a graph, including recursion overhead. I/O times have not been included.

These “raw” metrics are not reported here, but can be found in our master’s thesis. We report “averaged” metrics that give a global characterization of the behavior of each method.

- Average time (\overline{Time}). Computes the average time spent on all problems and all granularities. It is a good characterization of the time complexity of methods.
- Average balance (\overline{Bal}). It does for balance what the above does for time.
- Relative deviation from the best (Rel_dev). It is a percent measure normalized between 0 and 100. It measures how close partitions (of a given granularity) are to the best partition found.
- Average Relative deviation from the best ($\overline{Rel_dev}$). It is an average of the Rel_dev across all problems. It gives all problems more or less equal weighting in the average.

8 Discussion

In Tables 3 to 5 we present these global metrics: the $\overline{Rel_dev}$, \overline{Time} , and \overline{Bal} for each method. Our analysis is summarized in Figure 1: a decision tree for graph partitioning algorithms, according to the *granularity* of the partition desired, the degree of balance of the partition¹, and the availability of coordinates and parallel processing. Wherever two methods are listed in a single box, the first one is considered the best choice, but the second one is also competitive (in edge-cut size and execution time). We have considered for each method a tradeoff between quality and time to partition, avoiding extremes like C2-FSpec, which produces good quality at a tremendous cost, or C2-INertial, which is extremely fast but has poor quality. We have chosen the best parameters (from several tests not reported here) for each method tested.

¹“Tight balance” is when a user wants a perfectly balanced partition. “Loose balance” means less than 15% imbalance.

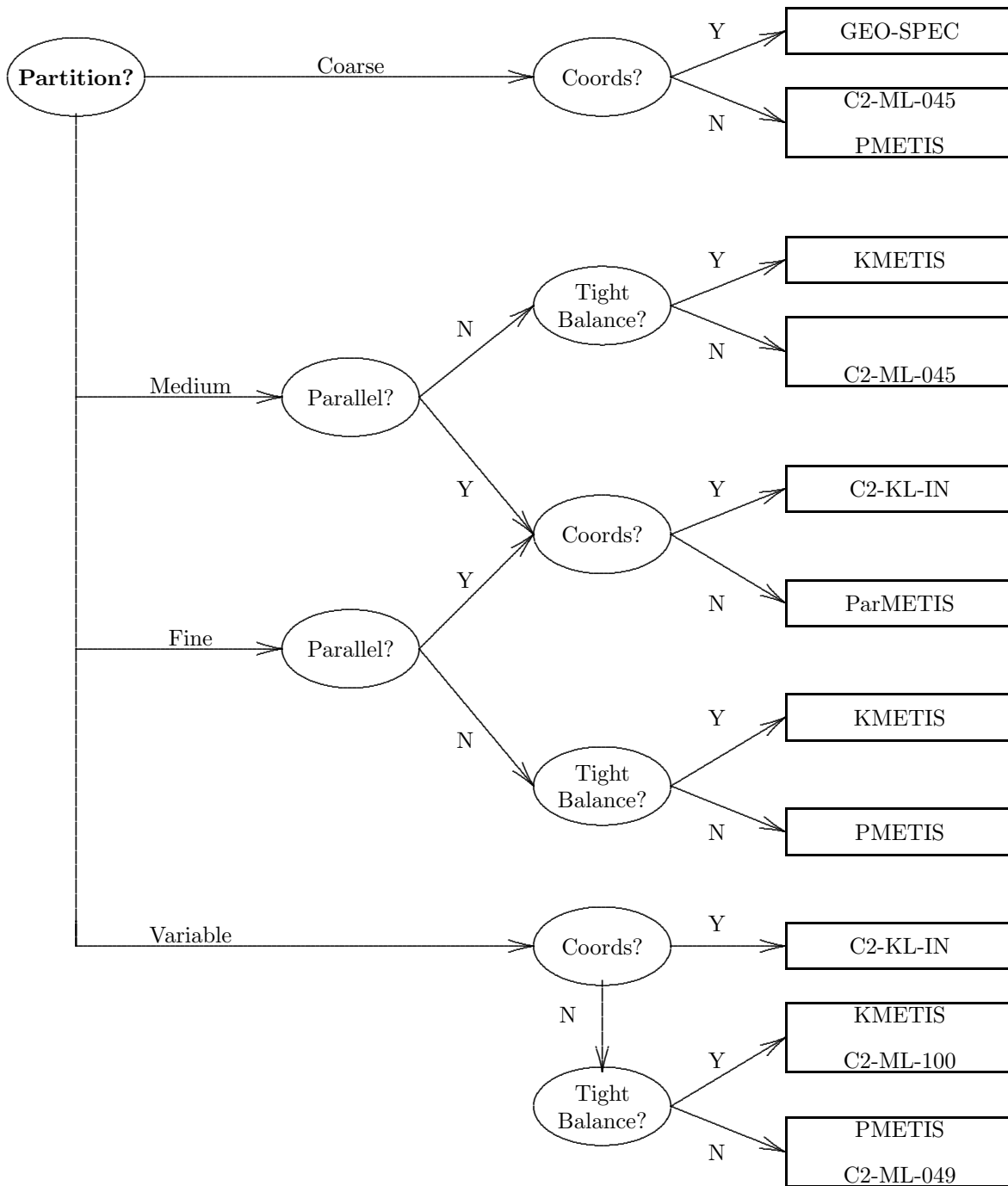


Figure 1: Graph Partitioning Algorithm Decision Tree

Method	$\overline{Rel_dev}$	\overline{Time}	\overline{Bal}
C2-ML-100	13.2 (4)	2.15	1.0
C2-ML-0.49	11.4 (3)	2.20	1.00
C2-ML-0.45	[9.2 (1)]	2.21	1.04
Pmetis2-ML	11.2 (2)	1.77	1.0
Kmetis2-ML	20.7 (7)	1.07	1.0
Geometric	20.5 (6)	—	1.0
Geo-Spec	24.5 (8)	—	1.0
C2-FSpec	25.8 (9)	15.16	1.0
C2-INerti	53.8 (11)	0.34	1.0
C2-KL-IN	32.4 (10)	1.37	1.0
C2-KL-FSpec	14.1 (5)	17.00	1.0

Table 3: Performance on 2 to 8 domains. For this and the following two tables, numbers in bold in the $\overline{Rel_dev}$ column correspond to those methods with smallest $\overline{Rel_dev}$. The number to the left is the metric itself, and the number to the right is the ranking. Enclosed between brackets is the best method.

Method	$\overline{Rel_dev}$	\overline{Time}	\overline{Bal}
C2-ML-100	6.5 (5)	7.75	1.0
C2-ML-0.49	5.6 (2)	7.85	1.01
C2-ML-0.45	[3.1 (1)]	7.45	1.10
Pmetis2-ML	5.7 (3)	4.60	1.00
Kmetis2-ML	5.8 (4)	2.17	1.0
Geometric	22.1 (9)	—	1.0
Geo-Spec	83.4 (11)	—	1.0
C2-INerti	53.8 (10)	0.34	1.0
C2-FSpec	19.2 (7)	43.27	1.0
C2-KL-IN	20.5 (8)	4.05	1.0
C2-KL-FSpec	7.7 (6)	48.34	1.0

Table 4: Performance on 16 to 64 domains

8.1 Coarse partitions

For coarse partitions (2 to 8 domains) many methods work well. In particular, it is worth considering methods that allow some imbalance, like C2-ML-0.45, since the balance control works very well. Geometric methods also usually do a very good job in these dimensions. We present some comments:

1. C2-ML-0.45 (which had an average imbalance of 4.2%) and C2-ML-0.49 (with less than 1% average imbalance) were the best multilevel methods.
2. Pmetis is better than Kmetis for coarse partitions.
3. C2-KL-FSpec. This method is 9.6 times slower than Pmetis, although it was consistently good.
4. *Geo-spectral* found the best partitions for most problems, but did poorly in problems with many edges.

Method	$\overline{Rel_dev}$	\overline{Time}	\overline{Bal}
C2-ML-100	5.4 (5)	27.47	1.0
C2-ML-0.49	5.1 (2)	27.46	1.04
C2-ML-0.45	[0.1 (1)]	17.36	1.13
Pmetis2-ML	5.3 (4)	9.81	1.05
Kmetis2-ML	5.1 (3)	11.48	1.0
Geometric	13.8 (9)	—	1.0
Geo-Spec	99.3 (11)	—	1.0
C2-INerti	53.8 (10)	0.34	1.0
C2-FSpec	13.7 (8)	81.81	1.0
C2-KL-IN	7.7 (7)	12.07	1.0
C2-KL-FSpec	5.7 (6)	90.51	1.0

Table 5: Performance on 128 to 1024 domains

5. Geometric is the best of the geometric methods.
6. C2-KL-In. It is 29% faster than Pmetis, which makes it attractive.
7. C2-INertial. It is 5.5 times faster than Pmetis, but has the lowest quality of all methods reported here.

8.2 Medium partitions

Again, multilevel methods with relaxed balance control and Kmetis are the best methods overall. These are our comments:

1. Chaco multilevel methods with relaxed balance control (*C2-ML-0.45* and *C2-ML-0.49*) produce the highest quality partitions. See discussion about balance of partitions immediately below.
2. Kmetis is 2.1 times faster than Pmetis and the quality is comparable. The imbalance of the partitions produced by C2-ML-0.49 is comparable to Pmetis's. C2-ML-0.45 produces has a high average imbalance of 10%. Chaco methods are 62% slower than Pmetis.
3. C2-KL-FSpec. It still produces good partitions, but it is still 10.5 times slower than Pmetis. We recommend it over C2-FSpec.
4. Geo-Spec. Geo-spectral becomes one of the worst methods after 8 domains. The reason seems to be that we only used one Fiedler vector as coordinate, and these did not give enough information to partition the graph into more than eight regions.

8.3 Fine partitions

In fine partitions, highly loose imbalance factors produce unacceptable imbalance, as would be expected. C2-ML-0.49 and Kmetis are the best methods. Kmetis is 2.4 times faster than C2-ML-0.49, and about as fast as Pmetis. All other methods rankings were consistent with those in medium partitions.

9 Conclusions

Our study shows that a careful consideration of the characteristics of problems, the granularity of the partition to be produced, and the introduction of a controlled imbalance give greater chances to discern among many of the methods currently available.

The multilevel methods in Chaco and Metis are outstanding methods. However, a careful selection of parameters is required to obtain a locally optimal quality and execution time tradeoff. Chaco methods with imbalance are extremely powerful. They gave some of the best partitions (in terms of small edge-cut size) in reasonable times. C2-ML-0.45 was the best for a wide range of partitions and many problems. The resulting imbalance was generally high for fine partitions. C2-ML-0.49 produced only slightly imbalanced partitions for most partitions and problems. PMetis, the default bisection version of Metis is generally faster than Chaco's multilevel methods. It is not as effective for fine partitions. For these partitions, KMetis is faster and produces higher quality and more balanced partitions.

We were also able to show that geometric methods, in particular Geometric and C2-KL-Inertial produce very good results. However, they are not applicable in problems with no coordinates. We explored the possibility of using spectral coordinates, which can be generated relatively cheaply using multilevel-RQI iterations in Chaco version 2.0. Geo-spectral method was a big surprise: for coarse partitions was undoubtedly very good for medium difficulty problems. On medium and fine partitions, however, it performed badly. This result opens a question: whether the poor performance of Geo-spectral in these dimensions was due to our having used only one Fiedler vector or whether it is rather a defect of the method? We think it is the former, and that using more Fiedler vectors should produce better results.

Currently we are conducting tests on larger graphs, where we expect the multilevel methods to do even better. Also, we are testing parallel graph partitioning methods, particularly ParMetis and Scotch. We give reference to the latter two packages on table 2, where as a service to potential users of graph partitioning algorithms for parallel-domain decomposition, we provide pointers to existing packages and a theoretical resource site.

References

- [Chan *et al.*, 1994] Tony F. Chan, John R. Gilbert, and Shang-Hua Teng. Geometric spectral partitioning. Report in Xerox site ftp.parc.xerox.com, 1994.
- [Garey *et al.*, 1976] M. Garey, D. Johnson, and L. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1:237–267, 1976.
- [Gilbert *et al.*, 1994] John R. Gilbert, Gary L. Miller, and Shang-Hua Teng. Geometric mesh partitioning: Implementation and experiments. Report in Xerox site ftp.parc.xerox.com, 1994.
- [Hendrickson and Leland, 1993] Bruce Hendrickson and Robert Leland. A multilevel algorithm for partitioning graphs. Technical Report SAND93-1301, Sandia National Laboratories, Albuquerque, NM, 1993.
- [Hendrickson and Leland, 1994] Bruce Hendrickson and Robert Leland. An empirical study of static load balancing algorithms. In *Scalable High-Perf. Comput. Conf.*, pages 682–685. IEEE, 1994.
- [Hendrickson and Leland, 1995] Bruce Hendrickson and Robert Leland. The Chaco user's guide, version 2.0. Technical Report SAND94-2692, Sandia National Laboratories, Albuquerque, NM, 1995.
- [Izaguirre, 1996] Jesús A. Izaguirre. An empirical evaluation of graph partitioning algorithms. Master's thesis, University of Illinois at Urbana-Champaign, Urbana, Illinois, 1996.
- [Karypis and Kumar, 1995a] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. Technical Report 95-035, University of Minnesota, Department of Computer Science, Minneapolis, MN 55455, July 1995.
- [Karypis and Kumar, 1995b] George Karypis and Vipin Kumar. Metis: Unstructured graph partitioning and sparse matrix ordering system, version 2.0. Paper at <http://www.cs.umn.edu/karypis>, August 1995.
- [Karypis and Kumar, 1995c] George Karypis and Vipin Kumar. Multilevel k -way partitioning scheme for irregular graphs. Paper at <http://www.cs.umn.edu/karypis>, August 1995.
- [Kernighan and Lin, 1970] Brian W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal*, 49:291–307, February 1970.
- [Nour-Omid *et al.*, 1986] B. Nour-Omid, A. Raefsky, and G. Lyzenga. *Solving finite element equations on concurrent computers*, pages 291–307. American Soc. Mech. Eng., 1986.
- [Pothen *et al.*, 1990] A. Pothen, H. Simon, and K. Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM J. Matrix Anal. Appl.*, 11:430–452, 1990.
- [Teng, 1991] Shang-Hua Teng. *Points, Spheres, and Separators: A Unified Geometric Approach to Graph Partitioning*. PhD thesis, Carnegie-Mellon University, Pittsburgh, PA, August 1991.