Repartitioning of Adaptive Meshes: Experiments with Multilevel Diffusion *

Kirk Schloegel, George Karypis, Vipin Kumar
University of Minnesota, Department of Computer Science
(kirk, karypis, kumar) @ cs.umn.edu

Abstract. For a large class of irregular grid applications, the structure of the mesh changes from one phase of the computation to the next. Eventually, as the graph evolves, the adapted mesh has to be repartitioned to ensure good load balance. If this new graph is partitioned from scratch, it will lead to an excessive migration of data among processors. In this paper, we present a new scheme for computing repartitionings of adaptively refined meshes. This scheme performs diffusion of vertices in a multilevel framework and minimizes vertex movement without significantly compromising the edge-cut.

1 Introduction

For a large class of irregular grid applications, the computational structure of the problem changes in an incremental fashion from one phase of the computation to another. An example of such an effect results with the use of adaptive meshes. Eventually, as the graph evolves, it becomes necessary to correct the partition in accordance with the structural changes in the computation and to migrate a certain amount of computation between processors. Failure to do so will lead to load imbalance, which will bog down parallel run time.

Thus, for adaptive applications, we need a partitioning or repartitioning algorithm with the following constraints. It is fast. It is scalable. It balances the graph. It minimizes edge-cut. It minimizes vertex migration time. The use of parallel graph partitioners will result in a balanced graph with small edge-cuts, but will lead to excessive movement of data among processors. In this paper, we present a new scheme for computing repartitionings of adaptively refined meshes. This scheme performs diffusion of vertices in a multilevel framework and minimizes vertex movement without significantly compromising the edge-cut.

* This work was supported by NSF CCR-9423082, by Army Research Office contract DA/DAAH04-95-1-0538, by Army High Performance Computing Research Center cooperative agreement number DAAH04-95-2-0003/contract number DAAH04-95-C-0008, by the IBM Partnership Award, and by the IBM SUR equipment grant. Access to computing facilities was provided by AHPCRC, Minnesota Supercomputer Institute. Related papers are available via WWW at URL: http://www.cs.umn.edu/~karypis

In this paper, `TotalV` is defined as the number of vertices which change partitions as the result of partitioning or repartitioning. `MaxV` is defined as the maximum number of vertices which migrate into or out of any one partition as a result of partitioning or repartitioning.

2 Repartitioning Strategies: Review of Previous Work

In partitioning or repartitioning a dynamic graph, two strategies are available. The first is to simply partition the new graph from scratch. The advantage of this strategy is that edge-cut is minimized. The second strategy is to use the existing partition as input for a repartitioning algorithm and to attempt to minimize the difference between the original partition and the output partition. This strategy has the potential benefit of reducing `TotalV` by an order of magnitude or more over partitioning the modified graph from scratch.

The second strategy can be accomplished by the following *cut-and-paste repartioning* method. Excess vertices in an overbalanced partition are simply swapped into one or more underbalanced partitions in order to bring these partitions up to balance. However, while this method will optimize the vertex migration time, it will have an excessively negative effect on the edge-cut compared with more sophisticated approaches.

Another method which reduces edge-cut degradation over cut-and-paste repartitioning, while increasing `TotalV` only moderately, is analogous to diffusion from thermal dynamics. The concept is for vertices to move from overbalanced partitions to underbalanced partitions and to eventually reach balance, just as in the analogous case, uneven temperatures in a space cause the movement of heat towards equilibrium [1].

Directed diffusion is diffusion guided by a global view of the graph. One method of computing this view (hereafter referred to as the *diffusion solution*) involves minimization of the two-norm of this solution. Hu and Blake described a method which computes the diffusion solution while optimally minimizing its two-norm [3].

Walshaw, Cross, and Everett implemented JOSTLE, a combined partitioner and diffusion repartitioner. Their repartitioning scheme has two phases, a balancing phase based on the Hu and Blake method [3] and a refinement phase based on Kernighan-Lin refinement [5]. They obtained low `TotalV` results and low edge-cut degradation on mildly perturbed graphs [8]. Walshaw, Cross, and Everett later implemented JOSTLE-MD. This replaced the refinement phase of the original JOSTLE algorithm with multilevel refinement [2, 4]. The result was a decrease in edge-cut over the original algorithm at the cost of an increase in `TotalV` [7]. Unlike our schemes, JOSTLE-MD applies directed diffusion on the full graph. Thus, it misses the benefits of multilevel diffusion.

2.1 Multilevel Schemes for Graph Partitioning

The multilevel graph partitioning algorithm described in [4] finds high quality partitions. It has three phases, a coarsening phase, a partitioning phase, and a

refinement, or uncoarsening, phase. During the coarsening phase, a sequence of smaller graphs are constructed from an input graph by collapsing vertices together. When enough vertices have been collapsed together so that the coarsest graph is sufficiently small, a partition is found using one among a variety of methods. Finally, the partition of the coarsest graph is projected back to the original graph by refining it at each uncoarsening level. Since now each uncoarsening level contains a finer graph, each subsequent graph has more degrees of freedom than the previous one had. These degrees of freedom can be used to decrease the edge-cut and increase the graph balance at each level.

Refinement is done in this scheme by a method based on the Kernighan-Lin partition algorithm [5]. Vertices are visited randomly. Each vertex visited is checked as to whether migrating partitions would

1. decrease the edge-cut while maintaining the graph balance, or
2. maintain the edge-cut and increase graph balance.

If so, the vertex is migrated. This process is repeated until it converges [4]. We define these two conditions as the *vertex migration criteria*.

3 Multilevel Diffusion

A repartitioning algorithm as a modification of the multilevel k-way partitioning algorithm implemented in MÉTIS can be derived as follows. In the coarsening phase, only those pairs of nodes that belong to the same partition are considered for merging. Hence, the initial partitioning of the coarsest level graph is identical to the current partition of the graph that is being considered for repartioning.

We added a partition enforcement level to the refinement phase. The partition enforcement level is a constant which determines whether a partition is so overbalanced as to warrant forced vertex migration. During refinement, if the weight of the partition of the currently selected vertex is greater than the partition enforcement level and the vertex is a border vertex and at least one of the vertex's neighbor partitions is not overbalanced, then the vertex must migrate regardless of the consequences to edge-cut. If the vertex must migrate and more than one of its neighbor partitions are not overbalanced, then the vertex selects a partition to migrate to according to the vertex migration criteria described above. Thus, as the uncoarsening phase progresses, balance is automatically sought in conjunction with refinement. We use this repartitioning algorithm as a base algorithm and refer to it as R-MÉTIS.

4 Multilevel Directed Diffusion

The R-MÉTIS algorithm as described earlier can potentially balance any imbalanced graph. However, the diffusion of vertices from overbalanced partitions does not make use of the global information about the location of underbalanced partitions when determining vertex movement. It is possible to use the diffusion

solution of the graph in order to direct the movement of vertices in R-MéTiS. That is, if the weight of a partition is greater than the enforcement level, then border vertices are migrated only in accordance with the diffusion solution. In essence, vertices flow directly from overbalanced partitions to underbalanced partitions with this global guidance. Thus, balancing is potentially quicker to converge than in R-MéTiS. This repartitioning algorithm will be referred to as Rf-MéTiS.

5 Results

We evaluated the performance of our repartitioning algorithms described above. These experiments were performed using 10 different graphs arising in finite element applications. All experiments were conducted on an SGI R10000 196MHz processor. The input graphs were partitioned by MéTiS and then the weights of some selected vertices were increased so as to overbalance and underbalance certain partitions.

Table 1 summarizes the results from these experiments. In particular, it shows the edge-cut, `TotalV`, `MaxV`, and run times for MéTiS, our multilevel repartitioner (R-MéTiS), our multilevel directed diffusion repartitioner (Rf-MéTiS), and a directed diffusion algorithm (DD-Repart). DD-Repart is essentially our implementation of the JOSTLE algorithm. The resulting value for each metric was normalized against the result from MéTiS. The means of 400 experiments are presented.

Table 1: Overview of Results

Algorithm	Edge Cut	`TotalV`	`MaxV`	Run Time
MéTiS	1.000	1.000	1.000	1.000
R-MéTiS	1.146	0.121	0.471	0.980
Rf-MéTiS	1.139	0.116	0.462	0.969
DD-Repart	1.455	0.173	0.504	2.880

Table 2: MDUAL, 4 Sources, 4 Sinks

Algorithm	Edge Cut	`TotalV`	`MaxV`	Run Time
MéTiS	34,317	252,747	5,499	15.0
Rf-MéTiS	35,262	36,237	2,912	16.1
DD-Repart	44,780	49,402	3,137	197.5

The R-MéTiS and Rf-MéTiS algorithms produced results within a few percent of each other for all four metrics. They both produced substantially lower results for `TotalV` and `MaxV` than the MéTiS partitioner while maintaining comparable

edge-cuts. They also proved to be faster and more effective than the directed diffusion algorithm. In particular, the directed diffusion algorithm (DD-Repart) produced greater `TotalV` and `MaxV` results and larger edge cuts than either R-METIS or Rf-METIS.

Table 2 shows the results obtained on one particular experiment for MDUAL with a 128-way partition. The results from partitioning from scratch (METIS), our multilevel directed diffusion repartitioning (Rf-METIS), and the directed diffusion algorithm (DD-Repart) are compared on this table. Again edge-cut, `TotalV`, `MaxV`, and run time are compared. Here, however, only the unnormalized results of a single experiment are given.

6 Conclusions

Our multilevel directed diffusion repartioning algorithm is an excellent and robust repartitioning algorithm. It has been thoroughly tested on a variety of graphs for a wide variety of possible imbalance structures and has been found to be fast, scalable, and effective on them. Furthermore, it is highly parallel in nature. We have parallelized an optimized version of this algorithm described in [6]. In our early results we have obtained high-quality repartitionings of eight million vertex graphs in under three seconds using a 256-processor Cray T3D.

References

1. G. Cybenko. Dynamic load balancing for distributed memory multiprocessors. *Journal of Parallel and Distributed Computing*, 7(2):279–301, 1989.
2. Bruce Hendrickson and Robert Leland. An improved spectral graph partitioning algorithm for mapping parallel computations. Technical Report SAND92-1460, Sandia National Laboratories, 1992.
3. Y. F. Hu and R. J. Blake. An optimal dynamic load balancing algorithm. Technical Report DL-P-95-011, Daresbury Laboratory, Warrington, UK, 1995.
4. G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. Technical Report TR 95-035, Department of Computer Science, University of Minnesota, 1995. Also available on WWW at URL http://www.cs.umn.edu/~karypis/papers/mlevel_serial.ps. A short version appears in Intl. Conf. on Parallel Processing 1995.
5. B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal*, 1970.
6. Kirk Schloegel, George Karypis, and Vipin Kumar. Multilevel diffusion schemes for repartitioning of adaptive meshes. Technical Report TR 97-013, University of Minnesota, Department of Computer Science, 1997. http://www.cs.umn.edu/~karypis.
7. C. Walshaw, M. Cross, and M. G. Everett. Dynamic load-balancing for parallel adaptive unstructured meshes. *Parallel Processing for Scientific Computing*, 1997.
8. C. Walshaw, M. Cross, and M. G. Everett. Dynamic mesh partitioning: A unified optimisation and load-balancing algorithm. Technical Report 95/IM/06, Centre for Numerical Modelling and Process Analysis, University of Greenwich, London, UK, December 1995.

This article was processed using the LaTeX macro package with LLNCS style