Skewed Graph Partitioning*

Bruce Hendrickson[†] Robert Leland[‡] Rafael Van Driessche[§]

Abstract

Graph partitioning is an important abstraction used in solving many scientific computing problems. Unfortunately, the standard partitioning model does not incorporate considerations that are important in many settings. We address this by describing a generalized partitioning model which incorporates the notion of partition *skew* and is applicable to a variety of problems. We then develop enhancements to several important partitioning algorithms necessary to solve the generalized partitioning problem. Finally we demonstrate the benefit of employing several of these generalized methods to static decomposition of parallel computing problems.

1 Introduction

Many combinatorial problems in scientific computing can be conveniently phrased in terms of graphs. A *graph* $G = (V, E)$ consists of a set of *vertices* V and a set of vertex pairs E commonly referred to as *edges*. We use e_{ij} to denote an edge between vertices i and j. Vertices and edges can have weights associated with them which we denote by $w_v(i)$ and $w_e(e_{ij})$ respectively.

Very often the solution of these combinatorial problems involves the *partitioning* of the associated graph, *i.e.* the vertices of the graph must be grouped into comparably-sized sets such that the number (or weight) of edges crossing between sets is small. Although this problem is NP-complete [8], a number of heuristics have been developed and successfully applied to practical problems. However, in many applications the graph partitioning abstraction fails to capture the true problem. As with many discrete optimization problems, the model is often improved when some sort of penalty function is included. This introduces a bias or *skew* into the partition. In the next section we describe a simple generalization to the graph partitioning problem which allows for skew. The remainder of the paper describes enhancements to several important graph partitioning algorithms to address this generalized problem and demonstrates application of these methods.

---

*This paper appears in the Proc. Eighth SIAM Conf. Parallel Processing for Scientific Computing.

[†]Sandia National Labs, Albuquerque, NM 87185-1110. bah@cs.sandia.gov.

[‡]Sandia National Labs, Albuquerque, NM 87185-0441. leland@cs.sandia.gov

Hendrickson and Leland were supported by the Mathematical, Information and Computational Sciences Division of the U.S. DOE, Office of Energy Research, and work at Sandia National Laboratories, operated for the U.S. DOE under contract No. DE-AC04-94AL85000. Sandia is multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the U.S. DOE.

[§]Alcatel Telecom, Dept. XE22, F. Wellesplein 1, B-2018 Antwerp, Belgium. driesscr@sh.bel.alcatel.be. Formerly Research Assistant, Katholieke Universiteit Leuven, supported by the Belgian Incentive Program "Information Technology"—Computer Science of the Future (IT/IF/5), and by the Belgian Programme on Interuniversity Poles of Attraction (IUAP 17), initiated by the Belgian State, Prime Minister's Office for Science, Technology and Culture.

2 Skewed Partitioning

When partitioning a graph, let each vertex i have a *desire* to be in set k denoted by $d_k(i)$. We now define a *skewed* graph partitioning problem in which we try to minimize the cut edges and maximize the satisfied desires. Let $s(i)$ be the set to which vertex i is assigned. We want to find a mapping s which minimizes the following objective function.

$$\text{(1)} \qquad \text{Minimize} \sum_{e_{ij}} \begin{cases} w_e(e_{ij}) & \text{if } s(i) \neq s(j) \\ 0 & \text{otherwise} \end{cases} - \sum_{i=1}^{n} d_{s(i)}(i)$$

Several observations are in order. First, the standard graph partitioning metric is described by the first summation. This could be replaced by any desired metric on cut edges without altering the basic idea of skew. Second, the importance of satisfying the desires relative to that of minimizing edge cuts can be controlled by scaling the desire values. And third, for a particular vertex, only differences between desires are relevant.

This simple extension to the basic graph partitioning problem has applicability to a range of applications:

• **Circuit Placement With Terminals**. Graph partitioning is commonly applied to the problem of placing circuit elements on a chip so that total wire length is kept small. The basic idea is to divide the chip area in half, and then partition the graph (or hypergraph) describing the circuit, assigning each partition to half of the chip. With few edges crossing between the chip halves, most wires are between localized elements. The process is then repeated recursively. However, a chip generally contains some *terminals*, pins which connect electrically to the outside of the chip. Circuit elements connected to these terminals should be located near the pins to reduce wire length, but the standard graph partitioning model doesn't account for these considerations. The skewed model easily encodes this consideration by giving vertices a desire to be in the chip half which contains any neighboring pins.

• **Static Decomposition for Parallel Computing**. Graphs are commonly used to describe the structure of many scientific calculations. When performing such a calculation on a parallel computer, the workload must be divided among processors in such a way that each processor has similar total work, but the communication is kept small. This problem is commonly abstracted to graph partitioning. However, within a parallel computer, communication between architecturally distant processors consumes more wires and generates more congestion than communication between neighboring processors. An optimal decomposition should take these effects into account. Unfortunately, the standard graph partitioning model is unable to do so. The skewed model can address this problem for approaches that recursively divide the graph and assign the pieces to portions of the parallel machine. Consider an intermediate stage in this process where a subset of the graph is being divided into pieces for subsets of processors. A vertex may have edges connecting it to vertices which are assigned to other portions of the machine. If so, the vertex can be given a desire to be assigned to a processor subset which is architecturally near the processors holding its neighbors. This basic idea was described by Kernighan and Dunlop for the circuit placement problem and given the name *terminal propagation* [4]. In the parallel computing context, this approach has been advocated by Pellegrini [15] and by the current authors [13]. It has been implemented in the Chaco 2.0 [10] and Scotch [16] partitioning tools.

• **Dynamic Decomposition With Reduced Data Movement**. For many scientific computations, the work can not be accurately predicted in advance. For these problems, efficient use of a parallel machine requires that data and computational work be periodically migrated between processors. The determination of a new distribution can often be expressed in terms of graph partitioning. However, the migration of data can be quite expensive, so it is preferable to leave as much as possible in place. This preference can not be addressed by the standard graph partitioning model. But by giving each vertex a desire to stay in its current set, the skewed model can accommodate this consideration.

3 The algorithm of Kernighan/Lin and Fiduccia/Mattheyses

In 1970, Kernighan and Lin proposed a heuristic for partitioning graphs based upon greedy exchange of vertices to reduce the number of edges cut [14]. Their basic approach has been enhanced and improved through the years, most significantly by Fiduccia and Mattheyses who devised a linear time variant [5]. This approach to partitioning is often referred to as KL/FM after these authors.

The KL/FM algorithm is a technique for improving an initial, perhaps random, partition. The key notion is that of the *gain* of a vertex, the net reduction in cuts which would ensue if the vertex were moved to the other partition. The basic step is selecting and moving a vertex with the highest gain value. There are two details which add complexity and considerable power to this very simple idea. First, in order to keep sets from becoming unbalanced, only moves between equal sized sets or from the larger to the smaller are allowed. Second, the algorithm continues trying to move vertices even if doing so makes the partition temporarily worse. The hope is that this reduction in quality will be compensated for by a larger improvement later on. This was the key insight of Kernighan and Lin's paper and makes the approach superior to a simple greedy algorithm.

The algorithm thus consists of two nested loops. In the inner loop, vertices whose movement would maximally improve the partition are selected, subject to set size constraints. Once a vertex is moved, the gain values of all its neighbors are updated. A particular vertex is allowed to move just once during each pass through the outer loop. The best partition encountered in this sequence of moves is recorded, and the outer loop resets the current partition to this best partition.

In 1985 Kernighan and Dunlop devised an extension to the KL/FM approach to address the problem of terminals in circuit placement [4]. The basic idea is to add a special vertex to each partition which is not allowed to switch partitions. For each normal vertex, add an edge to the special vertices with weight equal to the desire to be in the corresponding partition. In this way, the external edge information is internalized in the connections to the special vertices.

An equivalent, more elegant approach is possible when using a Fiduccia/Mattheyses implementation in which gain values are only computed once and are updated incrementally thereafter. The desires are simply included in the initial gain calculations while the rest of the code remains unchanged. The advantage of this approach is that the basic KL/FM loop need not be modified at all.

4 Multilevel Partitioning

4.1 Standard Multilevel Methods

The primary shortcoming of the KL/FM algorithm is that it enacts only local modifications to a partition. Although it is quite effective at finding local minimums, its solution may be quite far from the global optimum. This is particularly true for large graphs.

One possible remedy is to initialize KL/FM with a partition generated by another algorithm, for example the spectral bisection method discussed in §5. An alternate approach, suggested independently by several authors [1, 11] is to apply KL/FM on different scales. One way to think of this is as an algebraic multigrid technique in which KL/FM serves as the smoother.

Such a *multilevel* algorithm consists of three phases. First, a sequence of successively smaller graphs is generated from the original graph. Next, the smallest graph in the sequence is partitioned using some technique. This partition is then propagated back through the sequence of intermediate graphs, with KL/FM refinement being applied to refine the partitions of the intermediate graphs.

It is important that the small graphs represent their larger counterparts as accurately as possible. In the partitioning context, there are two properties we would like to preserve in the construction of the smaller graphs: the cost of a partition, and the set sizes so that a balanced partition of the small graph is also a balanced partition of the larger graph. These properties are preserved by the algorithm discussed here and in [11].

The key mechanism in the construction of a small graph is an operation known as *edge contraction*. In this step, two vertices joined by an edge are merged, and the resulting vertex is given edges to the union of the neighbors of the two merged vertices. The new vertex is assigned a weight equal to the sum of the weights of its constituent vertices. Edge weights are not changed unless both merged vertices are adjacent to the same neighbor. In this case, the new edge that represents the two original edges is assigned a weight equal to the sum of the weights of the edges it replaces. So, for example, contracting one edge of a triangle with unit edges and vertex weights would yield a graph with a vertex of weight one and a vertex of weight two, joined by an edge of weight two.

The attractive feature of this contraction step is that it preserves cut and set sizes in a *weighted* sense. A partition of a small graph implies a partition of a larger graph since each vertex in the small graph is merely an amalgamation of vertices of the larger one. The total weight of small graph edges that are cut in the partition will be precisely equal to the total weight of the edges cut in the larger graph. Similarly, the total weight of vertices in each of the two small graph sets is exactly equal to the weight of the vertices in the corresponding partition of the large graph.

To construct a small graph from a larger one we need to contract a number of edges. Ideally, these edges will be well distributed throughout the large graph so the overall shape of the small graph will be similar to that of its larger counterpart. One way to do this is to select a maximal set of edges that share no vertices. Such a set is known as a maximal matching, and can be easily generated in linear time.

4.2 Skewed Multilevel Partitioning

The multilevel approach can be enhanced to include skew in a straightforward way. We are apply the skewed variant of KL/FM on the smaller graphs. The only question is how to compute desires for the vertices of the small graphs. Recall that a vertex of a small graph is a union of large graph vertices. The net desire of the union is simply the sum of the individual desires. Hence, when contracting an edge, the resulting vertex should be assigned a desire value which is the sum of the desires of the original vertices.

5 Skewed Spectral Bisection

An important class of partitioning algorithms known as *spectral methods* uses eigenvectors of a matrix associated with the graph to generate a partition. This surprising connection dates back to work in the early 70s by Fiedler [6, 7] and Donath and Hoffman [2, 3]. A particular spectral method that has come to be known as spectral bisection gained widespread acceptance in the parallel computing community following the work of Pothen, Simon and Liou [17] and Simon [18]. In this section we generalize spectral bisection to incorporate skew. A more detailed presentation of this material can be found in [19, 12].

One way to describe a partition is to assign a value of $+1$ to all the vertices in one set and a value of -1 to all the vertices in the other. If we denote the value assigned to vertex i by $x(i)$, then the simple function $(x(i) - x(j))^2/4$ is equal to 1 if vertices i and j are in different partitions and 0 otherwise. This allows us to write the standard graph partitioning problem as

$$\text{(2)} \qquad \text{Minimize } \; f(x) = \frac{1}{4} \sum_{e_{ij} \in E} w_e(e_{ij})(x(i) - x(j))^2$$
$$\text{Subject to}$$
$$\text{(a)} \quad \sum_i w_v(i)x(i) \approx 0 \qquad \text{and} \qquad \text{(b)} \quad x(i) = \pm 1.$$

Constraint (a) is an algebraic way of saying that each partition must have about half the total vertex weight. We do not specify it as a precise equality since it may not be possible to divide the vertices into two sets of precisely equal weight.

We now need to enhance the objective function to include skew. Letting $p(i) = d_{+1}(i) - d_{-1}(i)$ be the net preference for vertex i to be in the set denoted by $+1$, the new problem we want to solve is

$$\text{(3)} \qquad \text{Minimize } \; f(x) = \frac{1}{4} \sum_{e_{ij} \in E} w_e(e_{ij})(x(i) - x(j))^2 - \frac{1}{2} \sum_{i \in V} p(i)x(i)$$
$$\text{Subject to}$$
$$\text{(a)} \quad \sum_{i \in V} w_v(i)x(i) \approx 0 \qquad \text{and} \qquad \text{(b)} \quad x(i) = \pm 1.$$

We now make an approximation that makes this algebraic problem much easier to solve. Rather than insisting that all x's be exactly ± 1, we allow them to take on any value and consequently replace constraint (b) with a norm condition on the vector x of values $x(i)$. Once we solve the resulting continuous problem, we can find the ± 1 vector which is nearest to the continuous optimum, and use this to partition the graph.

$$\text{(4)} \qquad \text{Minimize } \; f(x) = \frac{1}{4} x^T L x - \frac{1}{2} p^T x$$
$$\text{Subject to}$$
$$\text{(a)} \quad w_v^T x = 0 \qquad \text{and} \qquad \text{(b)} \quad x^T x = n.$$

We now make a change of variables to simplify the expression. First define $s(i) = \sqrt{w_v(i)}$ and $t(i) = 1/s(i)$. Let $y = \text{Diag}(s)x$, and let $A = \text{Diag}(t)^T L \text{Diag}(t)$. Since the x values are relaxations of ± 1, the appropriate normalization for the y vector is $y^T y = \sum_i w_v(i)$, which we denote by ω_v. Letting $h = \text{Diag}(t)p$ and multiplying the objective function by 4, we have

$$\text{(5)} \qquad \text{Minimize } \; f(y) = y^T A y - 2h^T y$$
$$\text{Subject to}$$
$$\text{(a)} \quad s^T y = 0 \qquad \text{and} \qquad \text{(b)} \quad y^T y = \omega_v.$$

To solve (5) introduce Lagrange multipliers η and μ, and look for stationary points of the function

$$F(y, \eta, \mu) = y^T A y - 2h^T y + \eta(s^T y) + \mu(\omega_v - y^T y). \tag{6}$$

Setting the partial derivative of F with respect to η or μ equal to zero yields the two constraint equations. Taking the derivatives with respect to the components of y, we obtain

$$2Ay - 2h + \eta s - 2\mu y = 0. \tag{7}$$

We can calculate η by left multiplying (7) by s^T. Since s is orthogonal to y and s is a zero eigenvector of A, we discover that $\eta = 2s^T h/\omega_v$. We now define

$$g = h - \frac{s^T h}{\omega_v} s, \tag{8}$$

which allows us to rewrite (7) as

$$Ay = \mu y + g. \tag{9}$$

This *extended eigenproblem* must be solved subject to the constraints in (5). Although this problem generally has multiple solutions, Van Driessche and Roose have shown that the solution which minimizes the objection function is always the y vector associated with the smallest possible value for μ [20]. Note that in the case that all net desires are zero, g becomes the zero vector and (9) reduces to the standard spectral bisection formulation. As with the standard spectral bisection approach, once a solution to (9) is computed, it is transformed back to a solution of (4), from which a nearby discrete solution can be found.

An efficient, Lanczos based procedure for solving the extended eigenproblem can be found in [9], [19] or [12], but is too lengthy to include here.

6 Results

The algorithms described in the previous sections have been implemented in Chaco 2.0 [10], and we report some experimental results here for the static partitioning problem. All the runs were performed a Sun Sparcstation 20 with a 50MHz clock and 64 Mbytes of RAM. We will describe results from four different algorithms: **ML** and **MLS** the multilevel method from §4.1 with and without skew, and **SFM** and **SSFM** spectral bisection with and without skew followed by FM with and without skew. For the spectral algorithms we solved to residual tolerances of 10^{-3}. In the skewed case we Lanczos with selective orthogonalization modified to handle the *extended* eigenproblem in which the additional constant vector appears. In the non-skewed case we used a variant of Barnard and Simon's multilevel RQI/SYMMLQ algorithm. For the multilevel partitioner and the multilevel eigensolver, the smallest graph had at most 200 vertices.

We monitored four metrics of partitioner quality. First was the number of edges cut, or *cuts*, which corresponds closely to the total communication volume in a parallel application. Second was *hops* in which we multiply each cut edge by the architectural distance between the two processors owning the endpoints. Third was *messages* which is the total number of messages required in a step of an iterative solver using the decomposition. The final metric was the *time* required to produce the decomposition.

Our first example graph is *barth5*, a 2D finite element grid with triangular elements containing 15606 vertices, and 45878 edges[1] The results of partitioning and mapping this graph to a 6-dimensional hypercube are presented in Table 1.

[1] This, and and other meshes, can be obtained via anonymous ftp to riacs.edu in the directory /pub/grids.

	ML	MLS	SFM	SSFM
cuts	2844	3187	2959	3530
hops	4832	3594	5052	3892
messages	288	322	282	360
time (s)	10.1	12.2	30.8	32.6

TABLE 1

Results of different partitioning algorithms on the barth5 mesh for a 6–dimensional hypercube.

As expected, the skewed algorithms significantly improve the data locality as evidenced by the significant reduction in hops. The average distance a datum has to travel is reduced from 1.7 to 1.1 in both algorithms. This comes at the cost of a modest increase in communication volume as reflected by the increase in the cuts metric, as well as an increase in number of messages. The time required to perform the partitioning is slightly increased when skew is incorporated.

Next, we partitioned the *ocean* mesh among the processors of a 10×20 mesh. This is a 3D finite difference grid of the world's oceans comprised of about 143K vertices and 410K edges. The results are presented in Table 2. Note that for this problem, we need to be able to bisect into two sets of unequal size. This is straightforward to do with the multilevel method, and a generalization to this case of spectral bisection with skew is described in [20].

	ML	MLS	SFM	SSFM
cuts	37847	45715	38365	52292
hops	103672	60210	103870	63163
messages	1652	1174	1614	1104
time (s)	157.6	186.7	690.3	477.5

TABLE 2

Results of different partitioning algorithms on the ocean mesh for a 10×20 *grid.*

Again we observe that skew significantly improves locality, reducing the average number of wires traversed by a message from 2.7 to 1.3 in the multilevel algorithm, and from 2.7 to 1.2 in the spectral method. As before this locality is paid for by an increase in communication volume. However, unlike the previous problem the number of messages is significantly reduced by the skewed formulation. Since communication is local and meshes have many fewer processors in their neighborhood than hypercubes, this result isn't surprising.

7 Conclusions

From a number of experiments like those reported above, we have drawn the following conclusions.

- Sophisticated partitioning algorithms can be successfully modified to incorporate the notion of skew.
- In the case of static partitioning for parallel computing, incorporation of skew significantly improves data locality. In particular, the skewed graph partitioning

approach generates decompositions which have significantly reduced communication congestion, and only modestly increased volume.

- For static partitioning, the skewed formulation can significantly reduce the number of messages on mesh topologies.
- The additional computational cost associated with incorporating skew in the multilevel and spectral partitioning algorithms is modest.
- As in the unskewed case, the skewed multilevel algorithm seems to produce better answers significantly faster than the skewed spectral algorithm.

References

[1] T. Bui and C. Jones, *A heuristic for reducing fill in sparse matrix factorization*, in Proc. 6th SIAM Conf. Parallel Processing for Scientific Computing, SIAM, 1993, pp. 445–452.

[2] W. Donath and A. Hoffman, *Algorithms for partitioning of graphs and computer logic based on eigenvectors of connection matrices*, IBM Technical Disclosure Bulletin, 15 (1972), pp. 938–944.

[3] ——, *Lower bounds for the partitioning of graphs*, IBM J. Res. Develop., 17 (1973), pp. 420–425.

[4] A. E. Dunlop and B. W. Kernighan, *A procedure for placement of standard-cell VLSI circuits*, IEEE Trans. CAD, CAD-4 (1985), pp. 92–98.

[5] C. M. Fiduccia and R. M. Mattheyses, *A linear time heuristic for improving network partitions*, in Proc. 19th IEEE Design Automation Conference, IEEE, 1982, pp. 175–181.

[6] M. Fiedler, *Algebraic connectivity of graphs*, Czechoslovak Math. J., 23 (1973), pp. 298–305.

[7] ——, *A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory*, Czechoslovak Math. J., 25 (1975), pp. 619–633.

[8] M. Garey, D. Johnson, and L. Stockmeyer, *Some simplified NP-complete graph problems*, Theoretical Computer Science, 1 (1976), pp. 237–267.

[9] G. Golub and U. von Matt, *Quadratically constrained least squares and quadratic problems*, Numer. Math., 59 (1991), pp. 561–580.

[10] B. Hendrickson and R. Leland, *The Chaco user's guide, version 2.0*, Tech. Rep. SAND94–2692, Sandia National Laboratories, Albuquerque, NM, October 1994.

[11] ——, *A multilevel algorithm for partitioning graphs*, in Proc. Supercomputing '95, ACM, December 1995.

[12] B. Hendrickson, R. Leland, and R. Van Driessche, *Enhancing data locality by using terminal propagation*, tech. rep., Sandia National Laboratories, Albuquerque, NM, May 1995.

[13] ——, *Enhancing data locality by using terminal propagation*, in Proc. 29th Hawaii Conf. System Sciences, January 1996.

[14] B. Kernighan and S. Lin, *An efficient heuristic procedure for partitioning graphs*, Bell System Technical Journal, 29 (1970), pp. 291–307.

[15] F. Pellegrini, *Static mapping by dual recursive bipartitioning of process and architecture graphs*, in Proceedings of SHPCC'94, Knoxville, IEEE, May 1994, pp. 486–493.

[16] ——, Scotch 3.1 *User's guide*, Technical Report 1137-96, LaBRI, Université Bordeaux I, August 1996.

[17] A. Pothen, H. Simon, and K. Liou, *Partitioning sparse matrices with eigenvectors of graphs*, SIAM J. Matrix Anal., 11 (1990), pp. 430–452.

[18] H. D. Simon, *Partitioning of unstructured problems for parallel processing*, in Proc. Conf. Parallel Methods Large Scale Structural Anal. & Physics Appl., Pergammon Press, 1991.

[19] R. Van Driessche, *Algorithms for Static and Dynamic Load Balancing on Parallel Computers*, PhD thesis, Katholieke Universiteit Leuven (Belgium), November 1995.

[20] R. Van Driessche and D. Roose, *A spectral algorithm for constrained graph partitioning I: The bisection case*, TW Report 216, Dept. Computer Science, Katholieke Universiteit Leuven, Belgium, October 1994.