SAND93–0074
Unlimited Release
Printed January 1993

Distribution
Category UC–405

Multidimensional Spectral Load Balancing

Bruce Hendrickson and Robert Leland
Sandia National Laboratories
Albuquerque, NM 87185

(Abbreviated version appeared in Proc. 6th SIAM Conf. Parallel Proc. Sci. Comput.)

Abstract

We describe an algorithm for the static load balancing of scientific computations that generalizes and improves upon spectral bisection. Through a novel use of multiple eigenvectors, our new spectral algorithm can divide a computation into 4 or 8 pieces at once. These *multidimensional* spectral partitioning algorithms generate balanced partitions that have lower communication overhead and are less expensive to compute than those produced by spectral bisection. In addition, they automatically work to minimize message contention on a hypercube or mesh architecture. These spectral partitions are further improved by a multidimensional generalization of the Kernighan–Lin graph partitioning algorithm. Results on several computational grids are given and compared with other popular methods.

This work was supported by the Applied Mathematical Sciences program, U.S. Department of Energy, Office of Energy Research, and was performed at Sandia National Laboratories, operated for the U.S. Department of Energy under contract No. DE-AC04-76DP00789.

1. Introduction. Efficient use of a distributed memory parallel computer requires that the computational load be balanced across processors in a way that minimizes interprocessor communication. This mapping requirement can be abstracted to a graph problem in which nodes represent computation, edges represent communication, and the objective is to assign an equal number of vertices to each processor in a way that minimizes the number of edges crossing between processors. Practical experience has shown that the quality of this mapping can have a substantial impact on performance, hence there is considerable interest in effective mapping algorithms.

A new heuristic for graph partitioning in the context of mapping parallel computations was recently proposed by Simon [15]. This *spectral* method recursively bisects a graph by considering an eigenvector of an associated matrix to gain an understanding of global properties of the graph. The method has received much attention because it offers a good balance between generality, quality and efficiency. The idea of using eigenvectors to partition graphs originated with work in the early 70's by Donath and Hoffman [3, 4], and Fiedler [6, 7]. In recent years these ideas have been revived and further developed by several authors [2, 12, 13, 14].

With one exception, all this previous work has been based upon the application of recursive bisection, a strategy which is limiting in several important ways. Rendl and Wolkowicz do describe an algorithm for partitioning a graph into an arbitrary number of sets without recursion [14]. However, their algorithm requires that k eigenvectors of a matrix representing the graph be determined in order to partition the graph into k sets. This renders the algorithm impractical for dividing large graphs into hundreds or thousands of sets, precisely what is required in mapping grand challenge problems onto massively parallel machines. The distinguishing feature of the work described in this paper is that it offers a practical generalization of spectral bisection to allow the division into more than two sets at once. This is especially appropriate in the context of modern parallel supercomputing.

Specifically, we describe two new algorithms, *spectral quadrisection* and *spectral octasection*, which partition into four sets using two eigenvectors and eight sets using three eigenvectors respectively. By weighing the effect of several cuts simultaneously, these new algorithms produce better partitions than spectral bisection, as measured in the *hypercube hop* (or *Manhattan*) metric. Empirical study has shown that this is the appropriate measure for modeling the performance of hypercube architecture machines since minimizing this metric corresponds to minimizing congestion within the communication network [9]. The hop metric is similarly appropriate for two and three dimensional mesh architectures (with or without wrap–around torus connections). Most distributed memory parallel machines have either mesh or hypercube architectures, so these new algorithms have wide application.

In addition to producing better partitions, these higher dimensional spectral algorithms have two other significant advantages. First, the logarithmic relationship between the number of required eigenvectors and the number of partitions means that in practice the new methods actually require less net computation than spectral bisection to define the same number of partitions. Second, by exploiting available redundancy in the solution space, higher dimensional spectral methods are able to correctly partition highly symmetric graphs which give spectral bisection trouble. Load balancing algorithms based on spectral quadrisection and octasection are thus generally more powerful, efficient and robust than those based on spectral bisection.

Many of the results presented here are taken from a recently completed report [10]. The purpose of this paper is to present those results in a less formal and more intuitive manner. In addition, the content of §6 and §7 has not appeared previously. In §2 of this paper, we describe spectral bisection and lay the mathematical and notational framework for the later sections. In §3 through §5 we describe our spectral quadrisection and octasection algorithms. A physical object whose behavior mimics that of the partitioning algorithms is described in §6, and a new algorithm to locally refine the spectral partition is presented in §7. Results of some sample calculations are presented in §8, followed by conclusions in §9.

2. Spectral bisection. Various formulations of spectral bisection can be found in papers by several authors [2, 12, 15]. We choose to set up the problem as follows. Define the graph G by vertex set V and edge set E. Index the n vertices with i or j, so V_i refers to the vertex with index i, $E_{i,j}$ denotes an extant edge between V_i and V_j, and $\sum_{V_i}$ and $\sum_{E_{ij}}$ specify sums over the vertices and existing edges, respectively. Now assign a variable x_i to each V_i such that $x_i = \pm 1$ and $\sum_{V_i} x_i = 0$. The first condition stipulates a partition into two distinct sets. The second requires that the sets be of equal size, assuming an even number of vertices. (We call a vector x whose elements satisfy these conditions an *indicator* vector since it indicates the set assignment of each graph vertex.)

The next step is to notice that the function $f(x) = \frac{1}{4}\sum_{E_{ij}}(x_i - x_j)^2$ counts the number of edges crossing between sets since $(x_i - x_j)^2$ contributes nothing to the sum if x_i and x_j have the same sign, and contributes 4 if they have opposite sign. Now that we have an objective function to minimize, we convert $f(x)$ to matrix form since that will make the solution more apparent. We start by defining the *adjacency* matrix

$$A_{i,j} = \begin{cases} 1 & \text{if } (V_i, V_j) \in E \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

We also define the *degree* matrix $D = diag(d_i)$, where d_i is the graph degree of V_i, i.e. the number of edges incident upon vertex V_i.

The conversion then proceeds as follows. Write

$$\sum_{E_{ij}}(x_i - x_j)^2 = \sum_{E_{ij}}(x_i^2 + x_j^2) - \sum_{E_{ij}} 2x_i x_j. \tag{2}$$

and recast the terms

$$\sum_{E_{ij}} 2x_i x_j = \sum_{V_i}\sum_{V_j} x_i A_{ij} x_j = \sum_{V_i} x_i \sum_{V_j} A_{ij} x_j = x^T A x. \tag{3}$$

$$\sum_{E_{ij}}(x_i^2 + x_j^2) = \sum_{E_{ij}} 2 = 2|E| = \sum_{V_i} x_i^2 d_i = x^T D x. \tag{4}$$

Now define the *Laplacian* matrix of the graph G

$$L_{i,j} = \begin{cases} -1 & \text{if } (V_i, V_j) \in E \\ d_i & \text{if } i = j \\ 0 & \text{otherwise,} \end{cases} \tag{5}$$

note that $L(G) = D - A$, and conclude $f(x) = \frac{1}{4}x^T Lx$. Coupling this with the constraints on x, we define the *discrete* bisection problem

$$\begin{aligned} &\text{Minimize} \quad && \frac{1}{4}x^T Lx \\ &\text{Subject to :} && x^T\mathbf{1} = 0, \qquad x_i = \pm 1, \end{aligned} \tag{6}$$

where $\mathbf{1}$ is the n-vector $(1, 1, 1, \ldots)^T$. Since graph partitioning is an NP-hard problem, we expect that there is no practical way to solve this problem. Undeterred, we relax the discreteness constraint on x and define the *continuous* bisection problem

$$\begin{aligned} &\text{Minimize} \quad && \frac{1}{4}x^T Lx \\ &\text{Subject to :} && x^T\mathbf{1} = 0, \qquad x^T x = n, \end{aligned} \tag{7}$$

in which the elements of x may take on any values satisfying the norm constraint. This continuous problem is only an approximation to the discrete problem, and the values defining its solution must be mapped back to ± 1 by some appropriate scheme to define a partition. Let us emphasize that the relaxation of the discreteness constraint is the crucial approximation in application of spectral methods to graph partitioning. Ideally, the solution to the continuous problem will have entries clustered near ± 1 so that it is a good approximation to the discrete problem.

We now begin the solution of (7) by noting that the Laplacian matrix has several important properties. If $u_1, u_2 \ldots u_n$ are the normalized eigenvectors of L with corresponding eigenvalues $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$, the following theorem is proved in [10].

THEOREM 2.1. *The matrix L has the following properties.*
(I) L is symmetric positive semi-definite.
(II) The u_i are pairwise orthogonal.
(III) $u_1 = n^{-\frac{1}{2}}\mathbf{1}$, $\lambda_1 = 0$.
(IV) If G is connected, then λ_1 is the only zero eigenvalue of L.

Next express x in terms of the eigenvectors of L: $x = \sum_i \alpha_i u_i$ where the α_i are real constants such that $\sum_{i=1}^n \alpha_i^2 = n$. Property II ensures that this is always possible since a set of n pairwise orthogonal vectors must span $\mathbb{R}^n$. By substitution for x we find $f(x) = \frac{1}{4}(\alpha_2^2\lambda_2 + \alpha_3^2\lambda_3 + \cdots + \alpha_n^2\lambda_n)$ since $\lambda_1 = 0$. Clearly

$$(\alpha_2^2 + \alpha_3^2 + \cdots + \alpha_n^2)\lambda_2 \leq \alpha_2^2\lambda_2 + \alpha_3^2\lambda_3 + \cdots + \alpha_n^2\lambda_n, \tag{8}$$

given the ordering of the eigenvalues, so $f(x) \geq n\lambda_2/4$. Notice that we can *achieve* $f(x) = n\lambda_2/4$ by choosing $x = \sqrt{n}u_2$. Notice also that this choice of x satisfies the balance constraint since $x^T\mathbf{1} = u_2^T u_1 = 0$, by properties II and III. Therefore, since

$x = \sqrt{n}u_2$ satisfies the constraints and minimizes $f(x)$, it is a solution to the continuous problem. If $\lambda_2 \neq \lambda_3$, this solution is unique ($x = -\sqrt{n}u_2$ defines the same partition).

There remains the task of mapping the solution of the continuous problem to a discrete partition. In the case of bisection there is a simple and natural way to do this. Find the median of the x_i values and then map vertices with corresponding x_i above the median to one set, and those below to the other. If several vertices share the median value they are assigned in a way that retains balance. This solution is the nearest discrete point to the continuous optimum.

An immediate corollary of the reasoning used to solve the continuous minimization problem is that $n\lambda_2/4$ is a lower bound on the number of cuts produced by any balanced partitioning of the graph. That is because the solution space of the continuous problem subsumes the solution space of the discrete problem. This result can be slightly improved using insight from the higher dimensional partitioning schemes, as we show in [10]. Unfortunately, while these lower bounds are simple and computable, they tend to be rather loose in practice.

A theoretical result of more practical interest involves a theorem due to Fiedler [6, 7]. This says that following a bisection, if the median value is larger than zero, then the subgraph consisting of all lower valued vertices is connected. Similarly, if the median value is less than zero, then the subgraph consisting of all higher valued vertices is connected. In both cases the other subgraph may be disconnected. A disconnected subgraph is problematic because we intend to apply the method recursively, and the method can fail badly if applied to a disconnected graph since Theorem 2.1(IV) breaks down. We therefore monitor connectivity of the subgraphs at each stage of recursion. If a disjoint subgraph is detected, we add a minimal number of *phantom edges* to establish connectivity, partition this subgraph and then remove the phantom edges. We find that in practice several subgraphs do become disconnected in the course of the partitioning of a typical large graph, and our phantom–edge strategy does noticeably improve overall results in those cases.

3. Spectral quadrisection. In order to partition a graph into four sets we need a second indicator vector, y, so that we can associate two bistate coordinates with each vertex. To make set assignment explicit, we define a mapping from the indicator vector to binary digits $\hat{x}_i = \frac{1}{2}(x_i - 1)$, $\hat{y}_i = \frac{1}{2}(y_i - 1)$ and then assign v_i to set 0, 1, 2 or 3 by interpreting $\hat{x}_i\hat{y}_i$ as a binary number. The question is then how we should choose x and y in order to achieve a good mapping.

Consider the objective function $f(x, y) = \frac{1}{4}(x^T Lx + y^T Ly)$. This counts hypercube hops, as illustrated in Fig. 1. To understand this, notice that $E_{1,2}$, which crosses the $x = 0$ plane, contributes 1 to the value of $f(x, y)$ through the $x^T Lx$ term. Similarly, $E_{2,3}$ crosses the $y = 0$ plane and contributes 1 to $f(x, y)$ through the $y^T Ly$ term. An edge like $E_{1,3}$, however, crosses both planes and hence contributes 2 to the value of $f(x, y)$.

We also need to revise the minimization constraints. In the continuous bisection problem we had the single constraint $x^T\mathbf{1} = 0$ to ensure load balance in the x indicator vector. In the quadrisection case we clearly need to add $y^T\mathbf{1} = 0$ to ensure balance in

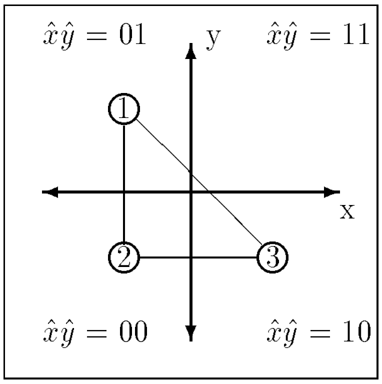

Fig. 1. The function $f(x, y) = \frac{1}{4}(x^T Ax + y^T Ay)$ counts hops in the plane.

the y indicator vector. A less obvious but necessary constraint proved in [10] is that $\sum_{V_i} x_i y_i = x^T y = 0$. This constraint serves to prevent a node distribution like that shown in Fig. 2, which is balanced with respect to both the $x = 0$ and $y = 0$ planes independently, but does not represent four balanced sets; a correct partitioning would assign two nodes to each quadrant.

Note that by way of physical analogy with a point–mass distribution, the $x^T \mathbf{1} = 0$ and $y^T \mathbf{1} = 0$ constraints specify a coordinate system in which the origin is placed at the center of mass. We can extend the analogy by considering the inertia tensor

$$T = \begin{pmatrix} \sum_{V_i} y_i^2 & -\sum_{V_i} x_i y_i \\ -\sum_{V_i} x_i y_i & \sum_{V_i} x_i^2 \end{pmatrix}. \tag{9}$$

The $x^T y = 0$ constraint combined with norm constraints $x^T x = y^T y = n$ stipulate a distribution in which T is a constant multiple of the identity matrix, i.e. there is no preferred axis of rotation. Intuitively, these conditions describe a roughly spherical mass distribution which is divided naturally into four balanced sets by the orthogonal axes. We might infer that for higher dimensional partitionings we must specify that higher moments are zero in order to preserve balance. This is in fact the case, as we prove algebraically in [10].

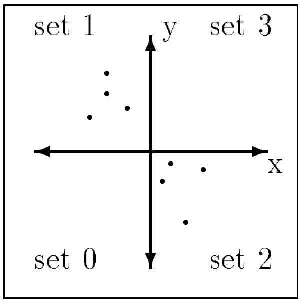

Fig. 2. An unbalanced but allowed quadrisection unless the constraint $x^T y = 0$ is enforced.

With these amendments, we can now follow the same progression from discrete to continuous model used in the bisection case and write continuous quadrisection

$$
\text{(10)} \qquad \begin{array}{ll} \text{Minimize} & \dfrac{1}{4}(x^T L x + y^T L y) \\ \text{Subject to :} & x^T \mathbf{1} = y^T \mathbf{1} = x^T y = 0, \qquad x^T x = y^T y = n. \end{array}
$$

The solution of (10) is actually very simple. We first note that if x and y are chosen to be any pair of distinct eigenvectors of L not including u_1 and scaled by $\sqrt{n}$, the constraint equations are all satisfied. That is because, by Theorem 2.1, all such eigenvectors are orthogonal and are also orthogonal to $\sqrt{n}u_1 = \mathbf{1}$. By extension of the algebraic argument used in (8) for the bisection case, we can prove that the minimum possible value of $f(x, y) = n(\lambda_2 + \lambda_3)/4$ is obtained by setting $x = \sqrt{n}u_2$ and $y = \sqrt{n}u_3$, and that $n(\lambda_2 + \lambda_3)/4$ is a lower bound on the number of hops induced by any balanced partition into four sets. This is a special case of the general solution theorem proved in [10].

Notice that if we set $x = \sqrt{n}u_2 \cos\theta + \sqrt{n}u_3 \sin\theta$ and $y = -\sqrt{n}u_2 \sin\theta + \sqrt{n}u_3 \cos\theta$, then x and y still satisfy the constraint equations and also produce the same minimum value of $f(x, y)$. Hence there is actually a family of solutions to (10) corresponding to the various choices of the free parameter θ. Different values of θ correspond to different solutions to the continuous problem that have the same value but specify different partitionings. An obvious question is therefore how we might exploit this rotational degree of freedom to advantage. We do so by choosing θ in a way that partially recovers the accuracy lost when we relaxed the discrete optimization model to the continuous one. That is, we attempt to minimize the discrepancy between the two models by making the continuous optimum as nearly discrete as possible. Specifically, we minimize $g(\theta) = \sum_{V_i}(1 - x_i^2)^2 + (1 - y_i^2)^2$ over $\theta \in (0, 2\pi)$. After substituting the trigonometric expansions for x and y, this reduces to minimizing a constant coefficient quartic equation in sines and cosines of θ. The construction of the coefficients in this equation requires $O(n)$ work, but the cost of the resulting minimization problem is independent of n and is not significant when partitioning large graphs. This is actually a global optimization, but in our experience the number of values of θ corresponding to local minima is small, so a good solution can be found in most cases by a short sequence of local minimizations from random starting points.

As in spectral bisection, there remains the problem of mapping the continuous, rotated solution back to discrete space. Unfortunately a simple generalization of the median technique used there is no longer adequate. Our guiding principle is once again to find the discrete solution nearest the continuous optimum. First we define a distance function from a continuous point (x_i, y_i) to a discrete point $(\pm 1, \pm 1)$. We wish to assign one fourth of the continuous points to each of the discrete points in such a way that the sum of the distances from the continuous values to their discrete assignments is minimized. This is an instance of a *minimum cost assignment* problem, for which efficient algorithms are known. In our code, we implemented an algorithm from [17] that runs in $O(n \log(n))$ time.

4. Spectral octasection. The development of the octasection algorithm follows closely that of quadrisection. We define a third indicator vector z mapped to a bit by $\hat{z}_i = \frac{1}{2}(z_i - 1)$ and assign each v_i to one of eight sets by interpreting $\hat{x}_i\hat{y}_i\hat{z}_i$ as a binary number. We then minimize the function $f(x, y, z) = \frac{1}{4}(x^T Lx + y^T Ly + z^T Lz)$ which counts hypercube hops on a three dimensional cube. To retain balance we must do this subject to the additional constraint that a particular third moment of the distribution is zero, i.e. $\sum_{V_i} x_i y_i z_i = 0$. (The physical analogy with mass distribution does not extend easily, although it is still true that the inertial tensor is a multiple of the identity.) Relaxing the discreteness constraint as before, we arrive at the continuous octasection problem

$$\begin{aligned}
(11) \qquad & \text{Minimize} \quad && \frac{1}{4}(x^T Lx + y^T Ly + z^T Lz) \\
& \text{Subject to :} \quad && x^T\mathbf{1} = y^T\mathbf{1} = z^T\mathbf{1} = x^T y = x^T z = y^T z = \sum_{V_i} x_i y_i z_i = 0, \\
& && x^T x = y^T y = z^T z = n.
\end{aligned}$$

Ignoring temporarily the third moment constraint, one solution to the problem is $x = \sqrt{n}u_2$, $y = \sqrt{n}u_3$ and $z = \sqrt{n}u_4$ with corresponding minimum (and hence lower bound on the number of hops induced by any balanced partition into eight sets) of $n(\lambda_2 + \lambda_3 + \lambda_4)/4$. Again there is redundancy in the solution space since any rotation of these eigenvectors generates another solution of equal value. Since we are working in 3D space, there are three rotational degrees of freedom. We use this freedom to now satisfy the triple product constraint while, as in quadrisection, also trying to make the continuous optimum as nearly discrete as possible. This yields a constrained optimization problem in three variables involving a constant coefficient quartic polynomial in sines and cosines of the angular parameters. The coefficients are computed in $O(n)$ time, after which the cost of the minimization is independent of n. The continuous, rotated solution is then mapped back to a discrete solution using the same algorithm for the minimum cost assignment problem employed in spectral quadrisection.

5. Higher dimensional spectral partitioning. If a d–dimensional partitioning scheme divides a mesh into 2^d parts at once, we have presented spectral schemes for $1 \leq d \leq 3$. Although some of the implementation details become more difficult, these ideas extend naturally to the $d = 4$ case. When $d > 4$ the moment constraints outnumber the rotational degrees of freedom, so it will not generally be possible to construct a balanced partition from the $d + 1$ lowest eigenvectors of L [10].

6. A physical model of spectral partitioning. Our intuition has been enhanced by the observation that there is a simple physical object whose behavior mimics spectral partitioning. Imagine a horizontal table into which n rigid rods have been embedded. The rods all extend perpendicularly from the table. Now place a bead on each rod that can slide vertically without friction. The beads, each with mass m, correspond to vertices in the graph to be partitioned. If there is an edge between vertices V_i and V_j, connect beads i and j with an ideal spring whose rest length is zero.

We will ignore gravity and let all spring constants have an equal value k. We will denote the position of bead i above the plane by $z(i)$, the horizontal separation between rods i and j by H_{ij}, and the extension of spring between beads i and j by $R_{ij} = \{H_{ij}^2 + (z(i) - z(j))^2\}^{\frac{1}{2}}$. In this case, $F(i)$, the force on bead i in the z direction is simply the combination of the forces of all the incident springs, so

$$F(i) = \sum_{j \text{ adj } i} kR_{ij} \frac{z(j) - z(i)}{R_{ij}} = k \sum_{j \text{ adj } i} z(j) - z(i). \tag{12}$$

In matrix terms this can be expressed $F = -kLz$, where L is the Laplacian matrix of the graph introduced earlier. Using Newton's second law we obtain a differential equation describing the oscillatory modes of the structure, $\ddot{z} = -\frac{k}{m}Lz$. The solution for mode l is $z(t) \propto u_l \cos(\sqrt{\frac{k\lambda_l}{m}}(t - t_o))$, where (λ_l, u_l) is an eigenpair of L. This modal solution consists of two factors: the cosine term represents a time oscillation, while the spatial distribution is described by the u_l factor. This spatial distribution is precisely the same as that used in spectral bisection. Hence spectral bisection partitions a graph based on the spatial envelope of the vibrational modes of the corresponding spring structure.

Note that lowest eigenvector of the Laplacian matrix is $n^{-\frac{1}{2}}\mathbf{1}$, which is a spatially flat mode. The lowest frequency *oscillatory* mode of this spring and bead structure therefore corresponds to the second lowest eigenvector of the Laplacian matrix. Spectral bisection simply divides the graph based upon this mode; the values above the median displacement get mapped to one set, and those below get placed in the other. Intuitively, the mode defines a long direction in the graph, and the algorithm divides the graph by cutting orthogonally to this direction.

It is important to note that, subject to the assumptions outlined above, this analysis is completely independent of the actual locations of the rods. The modes are a topological property of the graph, not a geometrical property of the physical object.

Spectral quadrisection and octasection are somewhat more difficult to interpret. Quadrisection uses the second lowest mode in addition to the lowest to divide the graph into 4 pieces, while octasection employs the three lowest modes to divide into 8 pieces. These algorithms form linear combinations of the low modes in a way that is difficult to describe physically.

7. Locally refining the partition. As the results in §8 will show, the spectral partitioning algorithms described above typically produce better partitions than competitive algorithms. This success is due to their ability to find promising regions of the graph in which to cut. However, this global strength is combined with a local weakness. Spectral methods often do poorly in the fine details of a partition. This observation suggests that spectral methods could be improved by coupling them with a heuristic to improve the partition locally.

One such heuristic is the graph bisection algorithm due to Kernighan and Lin (KL) [11]. This algorithm is quite good at finding locally optimal answers, but unless it is initialized with a good global partition, the local optimum can be far from the best possible. The synergy between the global strength of spectral methods and the local finesse of KL leads to an algorithm that is significantly better than either alone.

The KL algorithm has an outer loop that continues as long as improved partitions are being discovered. Each outer loop begins by computing a preference value q for each vertex. The preference value is the reduction in the number of edges between the two sets if the vertex were to switch sets. That is,

$$q(V_i) = \sum_{(V_i,V_j)\in E} \begin{cases} 1 & \text{if } P(V_i) \neq P(V_j) \\ -1 & \text{otherwise,} \end{cases} \tag{13}$$

where $P(V_i)$ is the current partition for vertex V_i. The algorithm then enters a loop in which it swaps two vertices between partitions, and updates the preference values of their neighbors. It always swaps the vertices with the largest preferences, and once a vertex is moved, it is not reconsidered. After each swap, the resulting partition is evaluated and the best one encountered is recorded and becomes the new partition upon exiting the inner loop. Note that since vertices are just swapped, if the initial partition is balanced then all generated partitions will be balanced as well. Each of these outer loops can be implemented to run in time proportional to the number of edges in the graph [5]. In practice, the number of passes required is quite small, usually fewer than 5, so the algorithm is effectively linear time. It also requires space proportional to the number of vertices in the graph. The algorithm is sketched in more detail in Fig. 3.

```
Until no better partition is discovered
    Best Partition := Current Partition
    S1 := Vertices in Partition 1
    S2 := Vertices in Partition 2
    For All Vi
        Compute preference value q(i)
    While S1 ≠ ∅ and S2 ≠ ∅
        Vi := vertex in S1 with largest preference
        P(Vi) := 2
        S1 := S1 \ Vi
        Update preferences of neighbors of Vi
        Vj := vertex in S2 with largest preference
        P(Vj) := 1
        S2 := S2 \ Vj
        Update preferences of neighbors of Vj
        If Current Partition better than Best Partition Then
            Best Partition := Current Partition
    Current Partition := Best Partition
```

Fig. 3. The Kernighan–Lin bipartition refinement algorithm.

Our spectral quadrisection and octasection algorithms divide a graph into four or eight pieces at once, so a local refinement of these partitioning algorithms should be able to handle more than just two sets. A generalization of KL to four sets is described by Suaris and Kedem for circuit layout problems [16], and we have extended this idea to an arbitrary number of sets. In addition, we are interested in minimizing the total number of hops spanned by graph edges spanning different processors, so we want a refinement algorithm that can handle arbitrary cost metrics for edges connecting different partitions.

We have developed an algorithm that generalizes Kernighan-Lin (GKL) to address these issues. Our algorithm differs from the original KL in several ways. First, instead of a single value, preferences need to be computed for a vertex transfer to any of the other sets, values we denote by $q^k(i)$. Second, the preference values include a factor for the inter-set metric, which for our purposes is chosen to be the hop metric. Third, moves are made singly instead of in pairs, since we might now want to move vertices in something other than a strict swap fashion. Fourth, unlike the original KL algorithm, partitions can be generated that are not balanced, so move selection tries to encourage the generation of balanced sets. This is accomplished by considering only moves from sets of at least average size to those that are at or below average size. And fifth, since intermediate partitions can be unbalanced, partitions are only considered for recording if they satisfy a balance criteria. The GKL algorithm is sketched below in Fig. 4. For η sets, this algorithm can be implemented to run in $O(\eta^2|E|)$ time and requires $O(\eta|V|)$ space.

```
Until no better partition is discovered
    Best Partition := Current Partition
    For All k ∈ {1,...,η}, S_k := Vertices in Partition k
    For All V_i
        Compute η − 1 preferences q^k(i)
    While allowed moves exist
        V_i := vertex with largest allowed preference
        l := set vertex V_i wants to go to
        S_P(V_i) := S_P(V_i) \ V_i
        P(V_i) := l
        Update preferences of neighbors of V_i
        If Current Partition balanced and better than Best Partition Then
            Best Partition := Current Partition
    Current Partition := Best Partition
```

Fig. 4. The generalized Kernighan–Lin multipartition refinement algorithm for improving a partition into η sets.

8. Results. We have empirically compared our recursive spectral quadrisection (RSQ) and recursive spectral octasection (RSO) algorithms to recursive spectral bisection (RSB) and another bisection algorithm in common use, the *inertial* method proposed by Nour-Omid [15]. We report here results for partitioning three representative application meshes.

The first example grid is a 2D finite element meshing of a multi-element airfoil provided by Barth and Jesperson of NASA Ames which has been commonly used in testing partitioners [1]; we have actually constructed and partitioned the dual ($|V| = 8034$, $|E| = 11813$) of this mesh since that is more appropriate for some finite element calculations. The second is another 2D CFD mesh generated by Hammond at RIACS [8] ($|V| = 4720$, $|E| = 13722$). The third is a three-dimensional finite difference mesh of a complex manufacturing component generated at Sandia ($|V| = 6661$, $|E| = 55600$). Table 1 shows the results obtained when four partitioning methods were used to divide each sample mesh into eight pieces. The methods are listed in rank order by hop count, which, as we mentioned earlier, has been shown to closely correlate with the overhead due to communication for these sorts of applications [9]. We have also shown another common measure of partition quality, the number of *cuts*, i.e. the total number of edges crossing between processor sets. Division into more than eight sets can be accomplished by recursive application of any of these methods.

Table 1. Performance of partitioning algorithms on sample meshes.

	Barth Mesh		*Hammond Mesh*		*Sandia Mesh*	
Method	cuts	hops	cuts	hops	cuts	hops
Inertial	317	396	880	1349	4652	5594
RSB	212	286	469	772	3425	5084
RSO	221	224	439	453	4138	4735
RSOKL	197	200	422	425	3140	3420

The suffix KL in RSOKL refers to appending the generalized Kernighan–Lin algorithm described in §7 to the spectral output. The resulting partition is clearly the best with respect to both hops *and* cuts. Notice also that the cut and hop totals are nearly equal for this algorithm, indicating that most communication is between neighboring processors.

9. Conclusions. We have described a static load balancing algorithm that is well suited to partitioning unstructured grids for problems in scientific computation. Our algorithm generalizes spectral bisection to allow for the division into four or eight sets at once. This multidimensional approach is faster than spectral bisection, and explicitly accounts for message congestion in mesh, torus or hypercube architectures. Several authors have found spectral bisection to produce better mappings of unstructured meshes than other techniques [15, 18], and in sample problems our new algorithm significantly outperforms spectral bisection.

REFERENCES

[1] T. Barth. Personal Communication, December 1991.

[2] R. Boppana, *Eigenvalues and graph bisection: An average case analysis*, in Proc. 28th Annual Symposium on Foundations of Computer Science, IEEE, 1987, pp. 280–285.

[3] 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.

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

[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] S. Hammond. Personal Communication, November 1992. Available through anonymous ftp at riacs.edu, file /pub/grids/3elt.grid.

[9] ———, *Mapping unstructured grid computations to massively parallel computers*, PhD thesis, Rensselaer Polytechnic Institute, Dept. of Computer Science, Troy, NY, 1992.

[10] B. Hendrickson and R. Leland, *An improved spectral graph partitioning algorithm for mapping parallel computations*, Tech. Rep. SAND 92-1460, Sandia National Laboratories, Albuquerque, NM, September 1992.

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

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

[13] D. Powers, *Graph partitioning by eigenvectors*, Lin. Alg. Appl., 101 (1988), pp. 121–133.

[14] F. Rendl and H. Wolkowicz, *A projection technique for partitioning the nodes of a graph*, Tech. Rep. CORR 90-20, University of Waterloo, Faculty of Mathematics, Waterloo, Ontario, November 1990.

[15] H. D. Simon, *Partitioning of unstructured problems for parallel processing*, in Proc. Conference on Parallel Methods on Large Scale Structural Analysis and Physics Applications, Pergammon Press, 1991.

[16] P. Suaris and G. Kedem, *An algorithm for quadrisection and its application to standard cell placement*, IEEE Trans. Circuits and Systems, 35 (1988), pp. 294–303.

[17] T. Tokuyama and J. Nakano, *Geometric algorithms for a minimum cost assignment problem*, in Proc. 7th Annual Symposium on Computational Geometry, ACM, 1991, pp. 262–271.

[18] R. Williams, *Performance of dynamic load balancing algorithms for unstructured mesh calculations*, Concurrency, 3 (1991), pp. 457–481.