Unification of Spectral and Inertial Bisection

Roy Williams

California Institute of Technology

Abstract

We discuss algorithms for graph bisection which are relevant to the distribution of tasks, such as the elements and nodes of an unstructured mesh to the processors of a parallel computer.

Starting with a cost function consisting of a part to ensure equal numbers of tasks for each processor, and a part to minimize communication time between processors, we derive the spectral bisection method, which consists of finding an eigenvector of a large sparse matrix.

Inertial bisection is derived as a statistical approximation from the spectral method, via the moments of the node density and moments of the pair distribution function.

Introduction

When a parallel computer has many tasks to perform simultaneously, each task is given to one of the processors. If we assume that each task communicates continually with a subset of the others, then the tasks take the structure of a graph. If the task graph remains constant or changes slowly, it may be advantageous to decide carefully to which processor each task should be assigned.

The resulting optimization problem is a graph-partitioning problem. The task graph is to be partitioned among the processors such that the number of *cut edges* (edges with one end in one processor and the other end in another processor) of the graph is minimized. The number of tasks per processor should be approximately equal to maximize parallelism of the computation, and the number of cut edges minimized to reduce communication costs.

In this paper, we shall assume that the graph-partitioning problem is to be solved by *recursive bisection* [8] where the graph is partitioned into two equal halves, then into halves again, and so on. We shall thus consider only the problem of partitioning the graph into two equal pieces.

There are several ways to solve approximately this graph bisection problem, and two popular algorithms are: spectral bisection, and geometric methods such as inertial bisection. The spectral method [1,2,7,9] uses properties of the task graph itself, such as the Laplacian matrix of the graph. In contrast, geometric methods [9] make a statistical hypothesis, assuming that each task is associated with a definite position in some geometric space, and ignore the detailed structure of

the task connectivity.

The Problem

Let G be a graph, with its nodes labelled by $i, j, ...$, with $|G| = N$ being the number of nodes. We shall use the notation $\sum_{(ij)}$ to mean a sum over all pairs of nodes which are connected by an edge, with *(ij)* and *(ji)* both appearing in the sum.

We wish to partition the graph into two subgraphs G_1 and G_2 as shown below. The partitioning

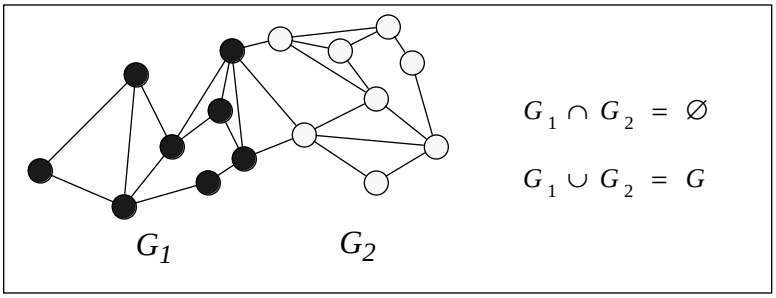

Figure 1: A graph partitioned in two equal subgraphs with three cut edges.

should be such that

- **Balance:** Each subgraph has about the same number of members, that is $|G_1| \approx |G_2|$. In Figure 1, each subgraph has eight nodes.
- **Communication:** The number of edges whose nodes are in different subgraphs is minimized, that is we wish to minimize $\sum_{(ij)} (i \in G_1) \; and \; (j \in G_2)$. In Figure 1, the number of cut edges is three.

This problem is related to the MINCES problem (minimum cut into equal-sized pieces), which demands $|G_1| = |G_2|$ and asks if there is a partitioning so that the number of cut edges is less than a given number. For more details see Reference [3].

Load Balancing

The first condition we shall refer to as the *balance* optimization, and the second as the *communication* optimization. This terminology is because the motivation for solving this graph optimization problem is the assignment of tasks to the processors of a parallel computer to minimize execution time. Each node of the graph is a task, and each edge represents a communication channel between two tasks. Partitioning the graph into two subgraphs turns out to be the key to the more general problem of partitioning into many subgraphs, using recursive bisection [8,9], so for the moment we assume there are two processors.

The balance optimization is to distribute the computational part of the work among the two processors, because if one has more work than the other then the calculation moves at the pace of the slowest.

The communication optimization assumes that each processor spends time dealing with the communication channels sequentially, so that the time taken by that processor is proportional to the number of communication channels.

We assume here for simplicity that each task requires equal time to execute, and that each communication channel requires equal time to service.

Problem Definition

Let us specify the graph partitioning problem in terms of a set of unknowns $s_i \in \{-1, 1\}$, where the node i is in G_1 if s_i is -1, else s_i is 1 and the node is in G_2.

Perfect balance is achieved when the sum of the s_i is zero, so we may write a cost function for the balance part as:

$$H_{balance} = \left(\sum_i s_i\right)^2$$

The cost function for the communication part may be constructed by observing that we want $s_i = s_j$ when the nodes i and j are connected by an edge, so using the sum-over-edges notation defined above we may write: the number of cut edges as:

$$H_{communication} = \frac{1}{2}\sum_{(ij)} (s_i - s_i)^2$$

We have not yet specified the optimization problem completely: we must minimize just one cost function, not two, so we may define the problem as a minimization of a linear combination of the balance and communication costs,

$$H = H_{balance} + \varepsilon H_{communication}$$

The constant ε is a measure of the cost of communication compared to calculation on the machine for which we are doing the load-balancing. It should be noted that when ε is sufficiently large (communication is very expensive), it is advantageous to put all tasks on a single processor. In this case H reaches the maximum value of $H_{balance}$, which is N^2, while $H_{communication}$ is zero, where N is $|G|$, the number of nodes in G.

If the computation is reasonably efficient, however, we expect the parallelism to be advantageous and the contribution of communication costs to be significantly less important than that of

calculation, so we shall be assuming that ε is in some sense small (to be defined below).

Spectral Bisection

The practical minimization of H is difficult because the underlying space $\{-1, 1\}^N$ is discrete. However, we shall at this point make an **approximation** that makes the minimization tractable: we replace the discrete variables s_i by continuous variables $v_i \in \Re$, but with the constraint that $\sum_i v_i^2 = 1$.

We may interpret this approximation geometrically by observing that the discrete space is a

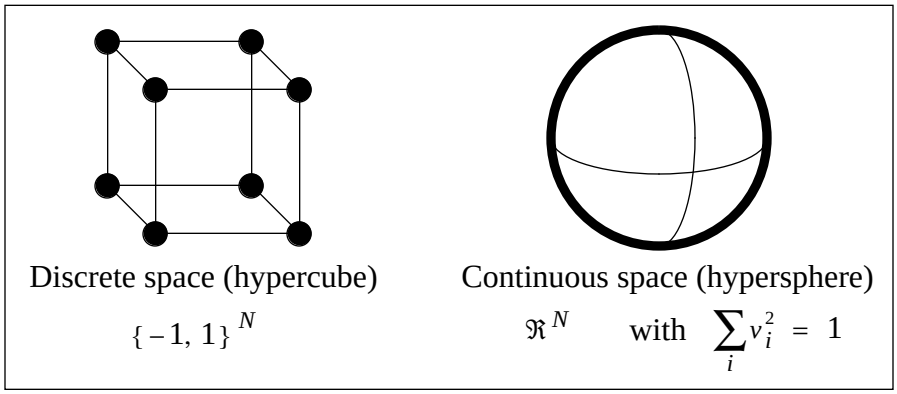

Figure 2: Domain for the discrete problem and its smoothed approximation

hypercube and the continuous space is the surface of a hypersphere, as illustrated in Figure 2. As $N \to \infty$, the density of points of the discrete space approaches the smooth distribution on the surface of the hypersphere.

Having made the approximation, we observe that H is a quadratic form, as is the constraint, so that the minimization of H is just an eigenvalue computation. Using a Lagrange multiplier λ for the constraint, the cost functions can be written as follows, in matrix-vector notation:

$$H_{balance} = \boldsymbol{v}^T \boldsymbol{E} \boldsymbol{v}$$

$$H_{communication} = \boldsymbol{v}^T \boldsymbol{Q} \boldsymbol{v}$$

$$H - \lambda \boldsymbol{v}^T \boldsymbol{v} = \boldsymbol{v}^T (\boldsymbol{E} + \varepsilon \boldsymbol{Q}) \boldsymbol{v} - \lambda \boldsymbol{v}^T \boldsymbol{v}$$

where $\boldsymbol{E}$ and $\boldsymbol{Q}$ are the matrices associated with balance and communication, respectively. When H is minimized with the constraint, we have

$$(\boldsymbol{E} + \varepsilon \boldsymbol{Q}) \boldsymbol{v} - \lambda \boldsymbol{v} = 0$$

Substituting, we find that $H = \boldsymbol{v}^T (\boldsymbol{E} + \varepsilon \boldsymbol{Q}) \boldsymbol{v} = \boldsymbol{v}^T \lambda \boldsymbol{v} = \lambda$, so that the eigenvalue λ is the achieved minimum, so that we are searching for the smallest eigenvalue of $\boldsymbol{E}+\varepsilon\boldsymbol{Q}$.

We have shown that an approximate solution of the graph bisection problem may be derived from the smallest eigenvalue of the matrix $\boldsymbol{E}+\varepsilon\boldsymbol{Q}$ that generates the quadratic form $H = \boldsymbol{v}^{\mathrm{T}}\boldsymbol{(E+\varepsilon Q)v}$.

First we note that each component of H is a sum of squares, and therefore $\boldsymbol{E}$ and $\boldsymbol{Q}$ are positive semi-definite. They are also symmetric, so they have a complete orthonormal set of eigenvectors.

Given the form of $H_{balance}$, we can see that the matrix $\boldsymbol{E}$ has elements which are all 1. Let us now define the vector $\boldsymbol{e} \in \Re^N$, whose components are all 1, and note that when $\boldsymbol{E}$ is applied to any vector, the result is a multiple of $\boldsymbol{e}$; that is $\boldsymbol{E}$ is a projector from $\Re^N$ to the one-dimensional space spanned by $\boldsymbol{e}$. Thus $\boldsymbol{E}$ has $\boldsymbol{e}$ as an eigenvector of eigenvalue N, and N-1 degenerate eigenvectors with eigenvalue zero.

We shall show below that the matrix $\boldsymbol{Q}$ has $\boldsymbol{e}$ as an eigenvector of eigenvalue zero. Since $\boldsymbol{e}$ is an eigenvector, the other eigenvectors of $\boldsymbol{Q}$ are orthogonal to it. Therefore the eigenspectrum of $\boldsymbol{E}+\varepsilon\boldsymbol{Q}$ is intimately related to that of $\boldsymbol{Q}$, in the following sense.

Lemma: Let the eigenvalues of $\boldsymbol{Q}$ be 0, λ_1, λ_2, ..., λ_{N-1}, with eigenvectors $\boldsymbol{e}$, $\boldsymbol{q}_1$, $\boldsymbol{q}_2$, ... $\boldsymbol{q}_{N-1}$ respectively. Then the eigenvalues of $\boldsymbol{E}+\varepsilon\boldsymbol{Q}$ are N, $\varepsilon\lambda_1$, $\varepsilon\lambda_2$, ..., $\varepsilon\lambda_{N-1}$, with the same eigenvectors.

If ε is sufficiently small, that is the parallel computation is reasonably efficient, then $\varepsilon\lambda_1 < N$, so the smallest eigenvalue of $\boldsymbol{E}+\varepsilon\boldsymbol{Q}$ corresponds to the smallest positive eigenvalue of $\boldsymbol{Q}$. The balance part of the optimization is thus exactly satisfied and the value of ε is decoupled from the load-balancing problem.

The Laplacian Matrix *Q*

The matrix $\boldsymbol{Q}$ is called the Laplacian matrix of the graph G. It is defined by:

- If $i \neq j$: Q_{ij} = -1 iff there is an edge connecting i and j
- If $i = j$: Q_{ii} is the number of edges at node i

Thus the sum across any row of $\boldsymbol{Q}$ is zero, so that $\boldsymbol{Qe} = 0$ as claimed above. This matrix is closely related to the adjacency matrix of the graph, which has matrix element 1 when two nodes are connected by an edge, and zero on the diagonal.

If we partition a connected graph in two equal pieces while minimizing the cut edges, we would intuitively expect each half to be a connected subgraph. This is indeed the case, as supported by a theorem of Fiedler [4,5].

Recovering the Discrete Solution

Once the relevant eigenvector $\boldsymbol{q}$ of $\boldsymbol{E}+\varepsilon\boldsymbol{Q}$ has been computed, we need to convert it from the (approximate) continuous form of the vector $\boldsymbol{q}$ back to the discrete form $\boldsymbol{s}$. Let $s_i = \mathrm{sign}(q_i - V)$,

where V is a ***median*** of the distribution of q_i, that is so that V is defined by:

$$\sum_i s_i = \sum_i \mathrm{sign}\,(q_i - V) = 0$$

Such a median value V generally will be possible to find when the number of nodes N is even. If N is odd, the RHS of the equation above can only be -1 or 1.

Notice that the eigenvector $\boldsymbol{q}$ that we have computed is orthogonal to the trivial eigenvector $\boldsymbol{e}$ of $\boldsymbol{Q}$. The orthogonality condition may be written $\sum q_i = 0$, so that in a continuum sense the balance condition is satisfied. However the orthogonality does not imply that $\sum \mathrm{sign}\,(q_i) = 0$, which is why the median computation is necessary.

Geometric Bisection

We have approximated the hypercube range of the solution by the surface of a hypersphere, reducing a difficult discrete optimization to the computation of an eigenvector of the Laplacian matrix of the graph. If the tasks being distributed are small, and/or the load balancing is done frequently, it may be that computing this eigenvector of an NxN sparse matrix is more work than is necessary. One solution to this is aggregating graph nodes [1], so that a smaller graph is partitioned, but each node of the smaller graph represents many nodes of the original graph.

In this section, we present another approach for reducing the computational effort of the partitioning, which assumes that each node of the graph is associated with a point in geometric space, and that connections between nodes are short-range in this space. We shall make the statistical approximation of the node distribution by means of a continuous ***density of nodes***, and the connectivity of the graph by a continuous ***pair distribution*** function.

Suppose that each node i of the graph is associated with a point $\boldsymbol{r}_i$ in a d-dimensional space, with the dimensions labelled by α, β, ... We may write the density of nodes ρ and the pair distribution function Π as:

$$\rho\,(\boldsymbol{r}) = \sum_i \delta\,(\boldsymbol{r} - \boldsymbol{r}_i)$$

$$\Pi\,(\boldsymbol{r}, \boldsymbol{r}') = \sum_{(ij)} \delta\,(\boldsymbol{r} - \boldsymbol{r}_i)\,\delta\,(\boldsymbol{r}' - \boldsymbol{r}_j)$$

where δ is the Dirac delta function. The continuum approximation consists of replacing the discrete space of node positions by a continuous geometric space, so that functions on the space are differentiable.

We can write the load-balancing cost function in integral form as

$$H = H_{balance} + \varepsilon H_{communication}$$

$$H_{balance} = \left[\int v(\boldsymbol{r})\,\rho\, d\boldsymbol{r}\right]^2$$

$$H_{communication} = \int \Pi(\boldsymbol{r}, \boldsymbol{r}')\,\left[v(\boldsymbol{r}) - v(\boldsymbol{r}')\right]^2 d\boldsymbol{r} d\boldsymbol{r}'$$

The analysis proceeds similarly to the discrete case. As before, there is a trivial solution $v(\boldsymbol{r}) = 1$ corresponding to the situation where communication is so expensive that all work is done on one processor. Taking solutions which are explicitly orthogonal to this means that

$$\int v(\boldsymbol{r})\,\rho\, d\boldsymbol{r} = 0$$

and we shall call such functions *zero-average* functions.

The constant ε is again not important (unless it is very large), so we can safely ignore $H_{balance}$. The problem is one of minimizing $H_{communication}$, while keeping the norm of v fixed, which is a minimization of:

$$H_{communication} - \lambda \int v(\boldsymbol{r})^2 \rho\, d\boldsymbol{r}$$

We now examine the form of the communication cost function. First we put $\boldsymbol{s} = \boldsymbol{r}' - \boldsymbol{r}$, then use Taylor's theorem for v, assuming that the pair distribution function is sufficiently short range to allow truncation at the linear term:

$$H_{communication} = \int d\boldsymbol{r} \int d\boldsymbol{s}\, \Pi(\boldsymbol{r}, \boldsymbol{r}+\boldsymbol{s})\,\left[v(\boldsymbol{r}+\boldsymbol{s}) - v(\boldsymbol{r})\right]^2$$

$$= \int d\boldsymbol{r} \int d\boldsymbol{s}\, \Pi(\boldsymbol{r}, \boldsymbol{r}+\boldsymbol{s}) \left[\sum_\alpha \frac{\partial v}{\partial r_\alpha} s_\alpha\right]^2$$

$$= \sum_{\alpha\beta} \int d\boldsymbol{r} \frac{\partial v}{\partial r_\alpha} \frac{\partial v}{\partial r_\beta} \int d\boldsymbol{s}\, \Pi(\boldsymbol{r}, \boldsymbol{r}+\boldsymbol{s})\, s_\alpha s_\beta$$

where the α and β indices range over the geometric dimension. The latter integral is a second moment of the pair distribution function, and depends on $\boldsymbol{r}$. We define the ***pair distribution moment matrix*** as:

$$\pi_{\alpha\beta}(\boldsymbol{r}) = \int d\boldsymbol{s}\Pi(\boldsymbol{r}, \boldsymbol{r}+\boldsymbol{s}) s_\alpha s_\beta$$

and thus the optimization problem may be expressed as follows. Find the zero-average function that minimizes:

$$\sum_{\alpha\beta}\int d\boldsymbol{r}\frac{\partial v}{\partial r_a}\frac{\partial v}{\partial r_\beta}\pi_{\alpha\beta}(\boldsymbol{r}) - \lambda\int v(\boldsymbol{r})^2\rho\, d\boldsymbol{r}$$

Inertial Bisection

Rather than minimizing the above quantity over all zero-average functions, we shall minimize over only linear functions, $v(\boldsymbol{r}) = c + \sum p_\alpha r_\alpha$. For simplicity, let us assume that the origin of the geometric coordinates has been moved to the center of mass of the nodes, so that $\int r_\alpha \rho\, d\boldsymbol{r} = 0$, and consequently we may take c to be zero.

Substituting, we need to minimize:

$$\sum_{\alpha\beta} p_\alpha p_\beta \int d\boldsymbol{r}\pi_{\alpha\beta}(\boldsymbol{r}) - \lambda\sum_{\alpha\beta} p_\alpha p_\beta \int d\boldsymbol{r} r_\alpha r_\beta$$

Writing the average of the pair distribution moment matrix as $P_{\alpha\beta} = \int d\boldsymbol{r}\pi_{\alpha\beta}(\boldsymbol{r})$, and the ***moment of inertia matrix*** as $I_{\alpha\beta} = \int d\boldsymbol{r} r_\alpha r_\beta$ we arrive at the eigenvalue problem:

$$\boldsymbol{P}\boldsymbol{p} = \lambda \boldsymbol{I}\boldsymbol{p}$$

The solution of such an eigenvalue problem may be interpreted geometrically as shown in Figure 3. In this case the graph we are trying to partition is the rectangular unstructured mesh.

The mesh is anisotropic, with the pair distribution function being longer range in the vertical direction, and shorter range in the horizontal direction. Thus the matrix $\boldsymbol{P}$ is shown by the set of vertical ellipses, which are contours of the quadratic form induced by $\boldsymbol{P}$. Since the density of nodes in the rectangle is uniform, the inertia matrix $\boldsymbol{I}$ follows the shape of the rectangle. The eigenvector we are searching for is then the direction that maximizes the quadratic form from $\boldsymbol{P}$

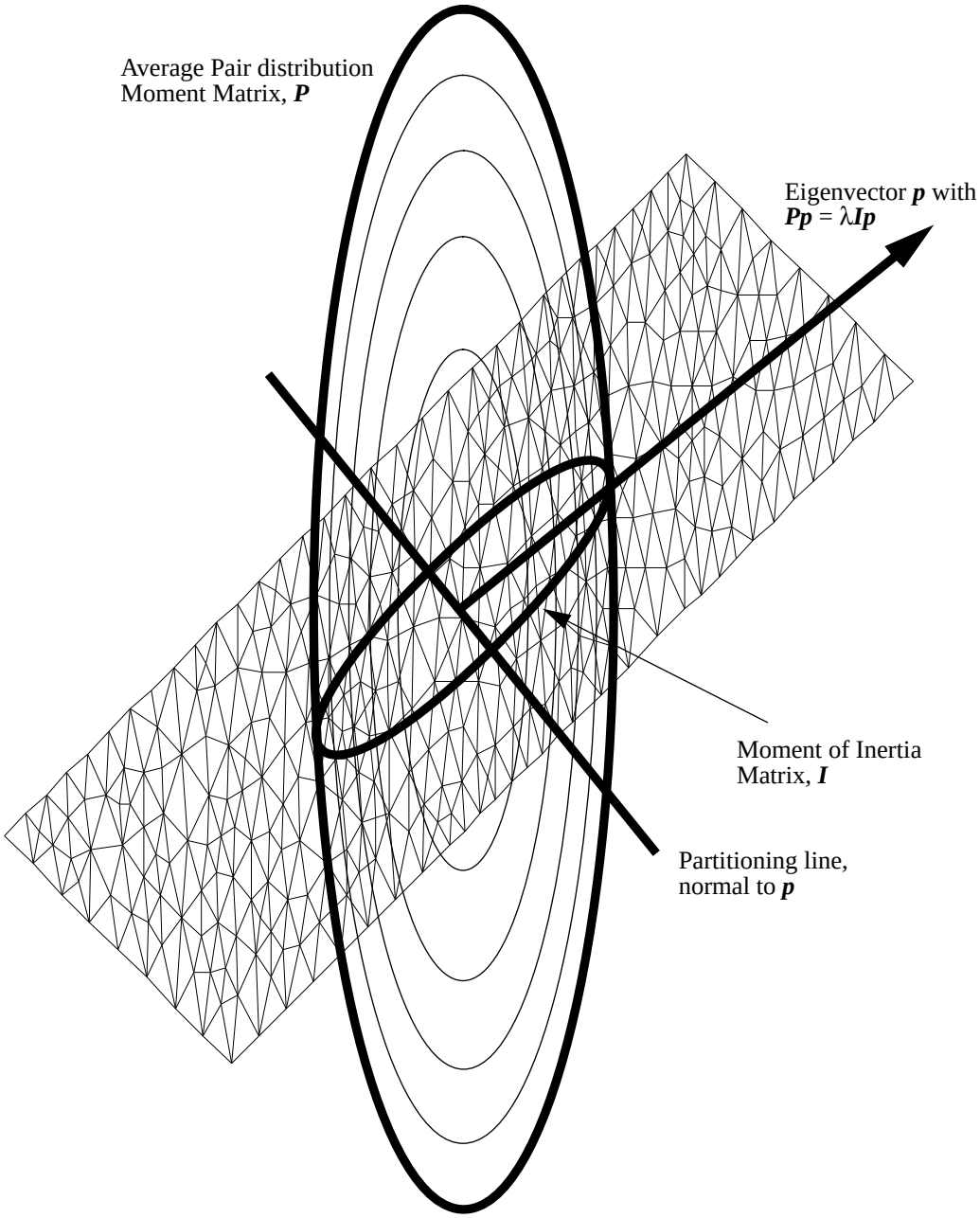

Figure 3. Geometric illustration of a graph to be bisected (the rectangle of unstructured mesh), the pair distribution function moment matrix (the vertical ellipse from the anisotropy of the mesh), and the moment of inertia matrix of the nodes (the ellipse at 45 degrees). Solution of an eigenproblem with these matrices (the arrow) leads to the optimal splitting plane for the graph (normal to the arrow).

with a constant value of the quadratic form from **I**, and is shown with an arrow. The cutting plane is then normal to this eigenvector.

In the special case where the pair distribution is isotropic, the matrix **P** is proportional to the unit matrix, and the solution of the above eigenvalue problem reduces to finding the eigenvectors of the inertia matrix, also known as the principle axes of inertia. In this case, the bisecting plane should be normal to the ***axis of minimum moment of inertia***.

Conclusions

We have started *ab initio* with a cost function for the graph bisection problem, consisting of balance and communication components. By a continuum approximation we have demonstrated the spectral bisection method, and that achieving perfect balance of tasks need not impact the optimization of the communication part of the cost-function.

We have naturally derived the geometric graph bisection algorithm, and in the case of a task graph which is on average isotropic, the inertial bisection algorithm. Thus the physically intuitive bisection of an object normal to the axis of minimum inertia is vindicated.

Further work in this area would be:

- Considering the effects of load-balance ***inertia***, also called remapping [6], which is relevant for adaptive computations; here we take into the cost-function a term expressing the time taken to migrate tasks from the existing processors to new processors. Thus a task is in some sense attached to the processor in which it currently resides.
- Accounting for ***latency*** effects: if two processors communicate a small amount of data, then additional data in the message takes the same time to communicate, so it is the number of processor-processor messages that is important rather than the number of task-task messages.
- Extending the geometric treatment to ***multigrid*** meshes, where communication occurs both within a mesh and between layers of mesh. The concept of locality in the pair distribution function must be redefined to efficiently bisect such structures.

Acknowledgements

This work was supported in part by Caltech, by the Concurrent Supercomputing Consortium, and by the Center for Research on Parallel Computation (a National Science Foundation Science and Technology Center, Grant CCR-8809615).

References

1 S. T. Barnard and H. D. Simon, *A Fast Implementation of Recursive Spectral Bisection for Partitioning Unstructured Problems,* p. 711 in R. F. Sincovec et. al., ed., Parallel Processing for Parallel Computing, SIAM, 1993. Also on the World Wide Web at
`http://www.nas.nasa.gov/RNR/People/sbarnard/RNR-92-033/RNR-92-033.html`

2 R. B. Boppana, *Eigenvalues and Graph Bisection: an Average Case Analysis*, in 28th Annual Symp. Found. Comp. Sci, 1987.

3 S. Even, *Graph Algorithms*, Computer Science Press, 1979, ISBN 0-914894-21-8.

4 M. Fiedler, *Algebraic Connectivity of Graphs*, Czech. Math. J., **23** (1973) 298.

5 M. Fiedler, *A Property of Eigenvectors of Non-negative Symmetric Matrices and its Application to Graph Theory*, Czech. Math. J., **25** (1975) 619.

6 C.-W. Ou, S. Ranka, and G. Fox. *Fast Mapping and Remapping Algorithm for Irregular and Adaptive Problems*. Proceedings of the 1992 International Conference on Parallel and Distributed Systems, pages 279-283, December 1993.

7 A. Pothen, H. D. Simon and K. P. Liu, *Partitioning Sparse Matrices with Eigenvectors of Graphs*, SIAM J. Mat. Anal. Appl., **11** (1990) 430.

8 H. D. Simon and S.-H. Teng, *How Good is Recursive Bisection?*, NASA Ames report RNR-93-012, obtainable on the World Wide Web at
`http://www.nas.nasa.gov/RNR/People/hsimon/RNR-93-012/RNR-93-012.ps`

9 R. D. Williams, *Performance of Dynamic Load Balancing Algorithms for Unstructured Mesh Calculations*, Concurrency 3 (1991) 457, also on the World Wide Web at
`http://www.ccsf.caltech.edu/~roy/annealing.ps.Z`