Optimal and Asymptotically Optimal Equi-partition of Rectangular Domains via Stripe Decomposition*

Ioannis T. Christou† Robert R. Meyer†

Abstract

We present an efficient method for assigning any number of processors to tasks associated with the cells of a rectangular uniform grid. Load balancing equi-partition constraints are observed while approximately minimizing the total perimeter of the partition, which corresponds to the amount of interprocessor communication. This method is based upon decomposition of the grid into stripes of "optimal" height. We prove that under some mild assumptions, as the problem size grows large in all parameters, the error bound associated with this feasible solution approaches zero. We also present computational results from a high level parallel Genetic Algorithm that utilizes this method, and make comparisons with other methods. On a network of workstations, our algorithm solves within minutes instances of the problem that would require one billion binary variables in a Quadratic Assignment formulation.

1 Introduction

1.1 Problem Formulation

The Minimum Perimeter Equi-partition problem (MPE) is a geometric problem with applications in scientific computing ([DTR91]), engineering and image processing ([Sch89]). In its most general form it can be stated as follows: *Given a Grid $\mathcal{G}$ of unit cells and a number of processors P, find an assignment of the grid cells to the processors so that the perimeter of the partition is minimized while the loads of the processors are as balanced as possible.* The perimeter of a partition is the sum of the lengths of the boundaries of the regions that each processor occupies , while the load of each processor is the area of the region it occupies. The problem is a special case of the (NP-complete) Graph Partitioning problem, and as such, it can be formulated as a Quadratic Assignment problem ([PRW93]), with $|\mathcal{G}|P$ binary variables and $|\mathcal{G}|+P$ constraints. Letting $\mathcal{I}$ denote the set of pairs of adjacent cells, and a_i the area for processor i, the QAP formulation is as follows:

$$\min . \sum_{i,j\in\mathcal{G}} \sum_{\substack{p,p'=1 \\ p\neq p'}}^{P} c_{ij} x_i^p x_j^{p'}$$

$$s.t. \begin{cases} \sum_{i\in\mathcal{G}} x_i^p = a_p & p = 1\ldots P \\ \sum_{p=1}^{P} x_i^p = 1 & i \in \mathcal{G} \\ x_i^p \in \mathbf{B} = \{0,1\} \end{cases}$$

$$\text{where } c_{ij} = \begin{cases} 1 & \text{if } (i,j)\in\mathcal{I} \\ 0 & \text{else} \end{cases}$$

*This research was partially supported by the Air Force Office of Scientific Research under grant F49620-94-1-0036, and by the NSF under grants CDA-9024618 and CCR-9306807.

†Center for Parallel Optimization, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin 53706.

The objective of this optimization problem is a sum of quadratic terms of binary variables, while the constraints are network constraints. An illustration of the network assignment nature of the problem is in figure 1 where each processor represents a supply node of supply a_i and each grid cell is a demand node of demand 1. The goal is to find a feasible assignment that minimizes the total perimeter.

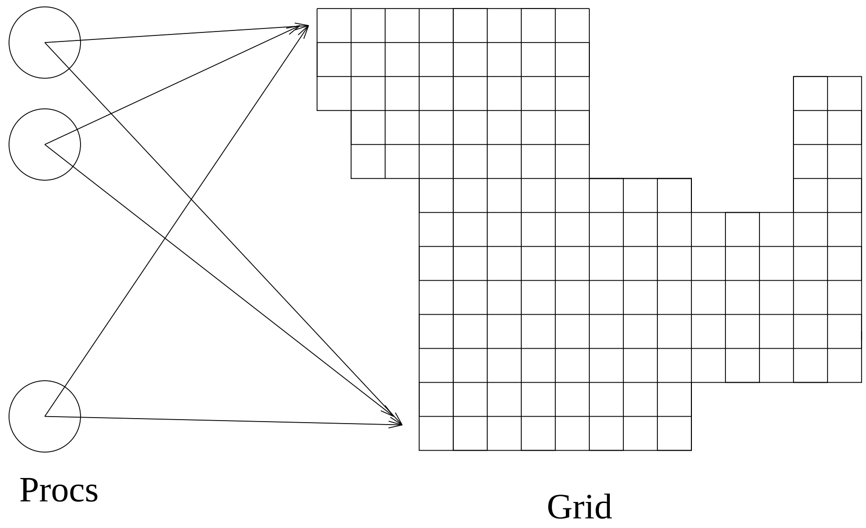

Figure 1: Network Assignment Formulation of MPE

Such problems arise naturally in the numerical solution of Partial Differential Equations (PDEs) over a given domain using finite difference schemes in parallel or distributed computing environments; in such cases, the domain is discretized thus giving rise to a grid, which then must be decomposed among the number of available processors, subject to the constraint that the load (areas) (a_i) of the processors are as balanced as possible. In the 5-point grid scheme, each cell updates its value using the values of its North, East, West and South neighboring cells ([DTR91]). Interprocessor communication occurs when a cell needs the value of another cell that does not belong to the same processor. This means that communication will occur exactly at the boundaries of the regions between the processors, and therefore, since the original boundary is a constant, minimizing the total perimeter of the partition corresponds to minimizing the overall interprocessor communication. The Minimum Perimeter problem arises also in the context of image processing ([Sch89]) and low-level computer vision, where for edge detection an image (a rectangular grid) has to be split among a number of processors in a parallel machine subject to the same load balancing constraints. Many edge detection algorithms use the 5-point grid computation scheme, so the objective of minimizing communication in the parallel machine again reduces to minimizing the perimeter of the partition. As the current trend in parallel computing is towards networks of workstations where the latency of communication between processors can be high, it is important that good solutions to the Minimum Perimeter problem can be found. In the case of networks of workstations in particular, the number of workstations connected together can be any number. Our method differs from most of the popular graph partitioning methods in that it can partition a domain into any number of processors while most methods require the number of processors to be a power of two.

Under the assumption that each processor has the same computing power, the load balancing constraints become *Equi-partitioning* constraints: the area a_i of each processor must differ by no more than 1 from the area of any other processor. In the rest of this paper we shall consider the Minimum Perimeter Equi-partitioning problem. We shall further restrict ourselves to the case where the grid $\mathcal{G}$ is a uniform rectangular grid of M rows and N columns. We shall refer to the problem of minimizing the perimeter of a partition of this grid into P processors as MPE(M, N, P).

The area of each processor then, if P divides exactly MN is the same for all processors and is simply $a_i = \frac{MN}{P}$. If, however, P does not divide MN, then we assume that $P \leq MN$, and the first $P_1 > 0$ processors will be assigned a load of A_1 and the remaining P_2 processors will get a load $A_2 = A_1 + 1$. Thus we have,

$$\begin{aligned} P_1 + P_2 &= P \\ A_2 &= A_1 + 1 \\ A_1 P_1 + A_2 P_2 &= MN \end{aligned}$$

from which we get

$$P_1 A_1 + (P - P_1)(A_1 + 1) = MN \implies PA_1 + P_2 = MN.$$

Therefore,

$$\begin{aligned} \text{for } i = 1 \ldots P_1 \;\; a_i = A_1 = MN \div P &= \left\lfloor \frac{MN}{P} \right\rfloor \\ P_1 &= P - MN \mod P \\ \text{and for } i = P_1 + 1 \ldots P \;\; a_i = A_2 = MN \div P + 1 &= \left\lceil \frac{MN}{P} \right\rceil \\ P_2 &= MN \mod P \end{aligned}$$

.

1.2 Related Work

There is a great deal of literature dealing with domain decomposition. By considering the graph of the grid, where for each grid cell, there is a vertex associated with it, and for any two neighboring cells there is an edge joining the associated vertices, one can apply graph partitioning techniques for decomposing the domain. Kernighan and Lin's heuristic ([KL70]) for partitioning a graph into two components is a very well known technique that is still used in many modern codes as a subroutine but has the disadvantage of requiring a relatively good initial partition upon which it attempts to improve. It is a standard local refinement routine incorporated in the Chaco package ([HL95a]). Pothen et. al. ([PSL90]) developed the spectral method in the context of general graph partitioning; discussion of improved spectral partitioning algorithms including spectral quadrisection or octasection can be found in [HL95b]. Laguna et. al. ([LFE94]) also developed a GRASP heuristic for partitioning a general graph into two pieces; and in [CQ95] Crandall and Quinn presented a heuristic for decomposing non-uniform rectangular grids among a number of heterogeneous processors. Also, Miller et. al. ([MTTV93]) have designed a domain decomposer for meshes based on geometric ideas.

The spectral method and its variations have received considerable attention as they are general methods for splitting a graph into two equally sized pieces while minimizing the sum of weights of the arcs with endpoints in both sub-graphs. However, extending the spectral method to decompose a graph among an arbitrary number of components is non-trivial for any number that is not a power of two. The same holds true for the geometric partitioner by Miller, et al.

Finally, Genetic Algorithm approaches to the graph partitioning problem have been proposed ([vL91]) where the length of each individual in the population is at least as big as the size of the graph. Our GA, in contrast, uses the theory of optimal shapes ([YM92a]) in a high-level approach that reduces the length of the individual to P, the number of processors. Performance comparisons of our GA with many of the above approaches is given in section 5.

2 Shapes of Optimal Regions

In [YM92a], Yackel & Meyer showed that for any given area A a processor must occupy, there exists a non-empty collection of configurations of A cells (shapes) with the property that all of the shapes in the collection have minimum perimeter $\Pi^*(A) = 2\left\lceil 2\sqrt{A}\right\rceil$, i.e. there is no configuration of A cells with less perimeter than the perimeter of the shapes in the collection. It turns out that all optimal shapes have a property called slice-convexity (which is the same as convexity in the "polyomino" literature), a consequence of which is the fact that the perimeter of any optimal shape is twice its semi-perimeter, where the semi-perimeter of any shape is the sum of the height and width of the smallest rectangle containing the shape.

It follows immediately that if P shapes from such a collection completely cover the grid, then this partition constitutes an optimal solution to the Minimum Perimeter problem and this optimal solution has a perimeter equal to $P\Pi^*(A)$. If P does not divide MN, an analogous optimal solution will have perimeter $P_1\Pi^*(A_1) + P_2\Pi^*(A_2)$.

By similar arguments, these values give lower bounds on the objective value of the MPE(M, N, P), which are tight in many cases but not always. Specifically, when the dimensions of the grid are not big enough to accommodate the relatively square optimal shapes, the lower bound fails to be tight. In particular, we have the following lemma:

Lemma 1 *Assume that $M < N$ and that the following problem $(\mathcal{P})$ is feasible:*

$$\min_{h,w} \; h + w$$
$$s.t. \begin{cases} hw \geq A \\ h \leq M \\ w \leq N \\ h, w \in I\!N. \end{cases}$$

Let $(\mathcal{P}_{rel})$ denote the relaxed problem

$$\min_{h,w} \; h + w$$
$$s.t. \begin{cases} hw \geq A \\ h, w \in I\!N. \end{cases}$$

Assume that all optimal solutions of $(\mathcal{P}_{rel})$ violate at least one of the constraints of $(\mathcal{P})$. Then, an optimal solution of $(\mathcal{P})$ is $(h^*, w^*) = (M, \lceil \frac{A}{M} \rceil)$.

Proof: For each $h = 1 \ldots M$ that corresponds to a feasible point of $(\mathcal{P})$ (i. e. satisfies $\lceil \frac{A}{h} \rceil \leq N$), the w in the range $\lceil \frac{A}{h} \rceil \ldots N$ that yields the best objective is the value $\lceil \frac{A}{h} \rceil$. Thus, we need only find the number h in the range $1 \ldots M$ that corresponds to a feasible solution and minimizes the function $f(h) = h + \lceil \frac{A}{h} \rceil$. But the function $f(h)$ in the range $1 \ldots \left\lfloor \sqrt{A} \right\rfloor$ is non-increasing. To see this assume $i < \left\lfloor \sqrt{A} \right\rfloor$; we are going to show that $i + \lceil \frac{A}{i} \rceil \geq i + 1 + \left\lceil \frac{A}{i+1} \right\rceil$, or equivalently that

$$\left\lceil \frac{A}{i} \right\rceil - \left\lceil \frac{A}{i+1} \right\rceil \geq 1.$$

But the number $d := \frac{A}{i} - \frac{A}{i+1} = \frac{A}{i(i+1)} > 1$ as $i(i+1) < \left\lfloor \sqrt{A} \right\rfloor^2 \leq A$. Therefore, the ceilings of the two numbers $\frac{A}{i}$, $\frac{A}{i+1}$ are at a distance greater than, or equal to one, so $\lceil \frac{A}{i} \rceil - \left\lceil \frac{A}{i+1} \right\rceil \geq 1$.

As $M < \left\lfloor \sqrt{A} \right\rfloor$ (otherwise, there exists an optimal solution of $(\mathcal{P}_{rel})$ that does not violate the extra constraints of $(\mathcal{P})$ because it is shown in [YM92a] that there always exist an optimal solution of $(\mathcal{P}_{rel})$ that has $h^* = \left\lfloor \sqrt{A} \right\rfloor$.) an optimal solution of $(\mathcal{P})$ is $(h^*, w^*) = (M, \lceil \frac{A}{M} \rceil)$ and the optimal objective value of $(\mathcal{P})$ is $M + \lceil \frac{A}{M} \rceil$. ∎

The above lemma (1) implies that when the domain is a sufficiently narrow horizontal band (M being small enough) so that no optimal shape from the collection of optimal shapes fits in the domain, then the optimal perimeter is $2(M + \lceil \frac{A}{M} \rceil)$.

Motivated by the above theory of optimal shapes, we may convert the partitioning problem into a tiling problem: find a set of shapes (from the appropriate collection of optimal shapes) that can be tiled together so as to completely cover the grid with no overlap and with minimum distortion. If such a set can be found that completely covers the grid with no distortion of the shapes, then the resulting partition is *provably optimal.*

Many of the shapes in this collection consist of a rectangle of dimensions $h \times w$ plus a fringe of size f, denoted as the tuple (h, w, f). The number of such near rectangular shapes is shown in [YMC95] to grow with A as $\mathcal{O}(A^{1/4})$. In our method, we restrict the initial choices of optimal shapes to this latter subset of the collection of optimal shapes. Furthermore, it follows from [Yac93], that given an area A, and letting $k = \left\lfloor \sqrt{A} \right\rfloor$, if $k^2 = A$ then $(k, k, 0)$ is an optimal shape. Else, if $k^2 < A < k(k+1)$ then the shapes (k, k, f), and $(k+1, k-1, f')$ are optimal (with $f < k$ and $f' = A - k^2 + 1 < k + 1$). And finally, if $k(k+1) \leq A$ then $(k, k+1, f)$ and $(k+1, k, f)$ are both optimal shapes (with $f \leq k$). These shapes play a key role in proving the existence of a stripe-form solution of the MPE(M, N, M) where $M \geq N$.

Note that in the combinatorics literature ([Lin91, Mel94]), much research has been published on the generating function approach for developing expressions for the exact number of "convex polyominoes" with various properties. However, our method (described below) is based on a library comprised of near-rectangular minimum perimeter configurations for a given area, so that the full collection does not have to be counted or generated.

3 Optimal Tilings

In this section we observe that the lower bound described above for MPE may be attained if the number of processors is large relative to the area of the grid. In particular, in the case where the number of processors is such that $\frac{MN}{P} \leq 3$, it is easy to construct an optimal solution that achieves the lower bound (because any connected configuration of 1, 2, or 3 cells is an optimal shape for the respective area size). In the case that $3 < \frac{MN}{P} \leq 4$, some processors will occupy an area of three cells and some will occupy an area of four. Then, a *necessary and sufficient condition* for the existence of an optimal solution that achieves the lower bound is the existence of a subrectangle of the grid that can accommodate all the optimal shapes for the processors having area four (there is only one optimal shape of area four, namely the 2×2 square). Necessity follows immediately from the fact that the grid is a subrectangle of itself, therefore if the squares cannot fit in the original grid, the lower bound cannot be attained. The condition is also sufficient because then, we can obtain an optimal partition that achieves the lower bound by tiling all the square shapes having an area of four in the north-western corner of the grid and filling the remaining area of the grid, row by row, with shapes of area three (any connected configuration of three cells is an optimal shape of area three).

If there exists no subrectangle that can accommodate 2×2 optimal shapes for all of the regions of area 4, then an optimal solution may be obtained by filling the largest (if any) northwest subrectangle of even dimensions with 2×2 shapes, then completing the assignment of the remaining cells (which will all lie in a single row and/or column) by assigning the remaining processor indices connectively left to right then bottom to top along the unassigned border. It is easily seen that this induces the minimum possible perimeter increase in the minimum possible number of shapes of area four whose shapes must be non-optimal and assigns optimal shapes of area three.

For these optimal solutions, we require $P \geq \frac{MN}{4}$, i.e. P is at least $\frac{1}{4}$ of the area of the grid. In the next section we show how to construct asymptotically optimal solutions if P dominates the individual dimensions M, N.

4 Asymptotically Optimal Solutions via Stripe Decomposition

In [CM95], we proved the following theorem:

Theorem 2 *The MPE(M,N,P) with $P = M \geq N$ has a feasible solution whose total perimeter possesses a relative distance δ from the lower bound that satisfies*

$$\delta < \frac{1}{\left\lceil 2\sqrt{N} \right\rceil}. \tag{1}$$

The proof of this theorem is an illustration of the stripe-decomposition technique. For any integer $k \geq 0$, if the number of rows of the grid M is at least $k(k-1)$, we can always find two natural numbers a, b such that

$$M = ak + b(k+1). \tag{2}$$

Letting now $k = \left\lfloor \sqrt{N} \right\rfloor$, where N is the area of each processor for the problem MPE(M, N, M), we can decompose the rows of the grid into stripes of height k or $k+1$. Each stripe can be filled with optimal and near optimal shapes, using a stripe-filling process ([CM95]) and the perimeter of each non-optimal shape will be at most two more than the optimal. A provably optimal partition of a 200×200 grid among 200 processors that is in stripe-form is shown in figure 2.

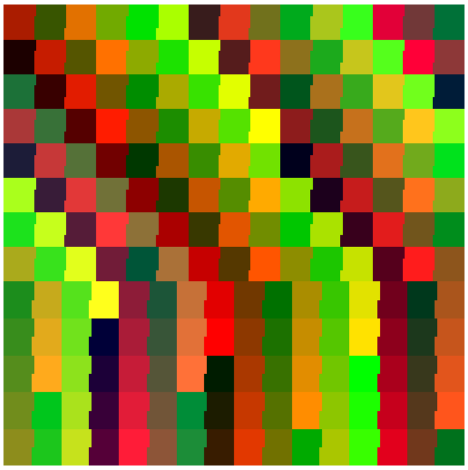

Figure 2: Optimal Partition in Stripe-Form for MPE(200,200,200)

The following theorem establishes an error bound for stripe-decomposition that improves on a related result in [CM95]:

Theorem 3 *Assuming P divides MN and that $P \geq max(M,N)$ the perimeter minimization problem MPE(M,N,P) has a feasible solution whose relative distance δ from the lower bound satisfies*

$$\delta < \frac{1}{\sqrt{A}} + \frac{1}{A}. \tag{3}$$

Thus the error bound δ converges to zero as A (the area of each processor) tends to infinity.

Proof: The grid is shown in figure 3. Note that $A \leq \min\{M, N\}$ and write $N = wA + d$

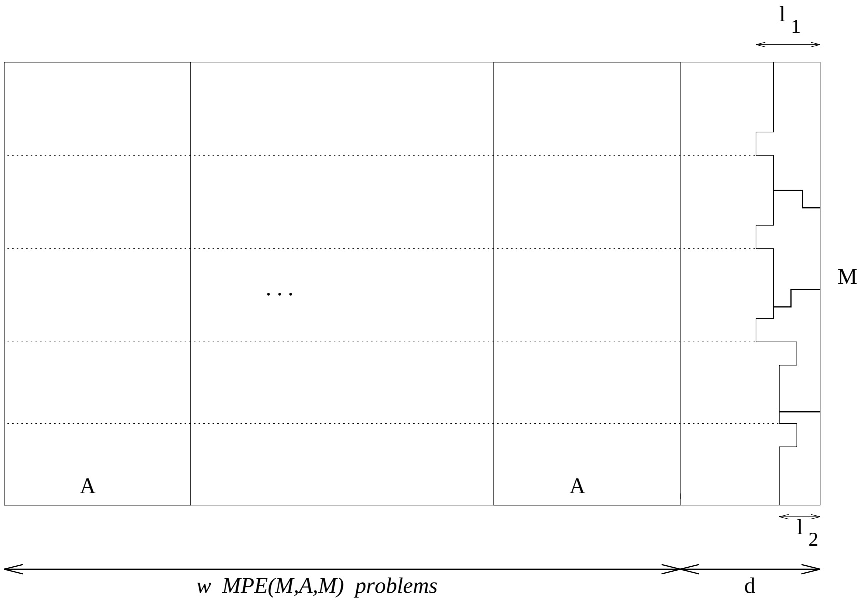

Figure 3: MPE(M, N, P), $P \geq max(M, N)$

for some naturals $w \geq 1$ and $d < A$. Define $k = \lfloor \sqrt{A} \rfloor$. Observe that the problem can be decomposed into w MPE(M, A, M) problems, and a MPE$(M, d, Md/A)$. In each of the w problems MPE(M, A, M), use the stripe decomposition method of theorem 2 to get a total absolute perimeter error $e < 2wM$. This striping technique (which partitions the rows of the grid into $r \leq M/k$ stripes of height $h_1, \ldots, h_r$) is continued over the last d columns in each stripe until no additional shape can be placed in the stripe. Let p denote the number of processors that have not been assigned. The stripe decomposition in the last d columns thus placed $\frac{Md}{A} - p$ processors, each of which may have an error in perimeter of no more than two.

The stripe decomposition for MPE(M, A, M) uses at most two different shapes. Arrange the stripes of the grid so that all stripes that use the first shape are used in the top rows of the grid which we will refer to as area 1, and all the stripes that use the second shape are in the (remaining) bottom rows which we will refer to as area 2. Let l_i $i = 1, 2$ denote the maximum number of columns in area i that contain unassigned grid cells, and without any loss of generality, assume that $l_1 \geq l_2 \geq 0$.

We place the last p processors in the remaining area using the following "orthogonal stripe filling" algorithm that approximates the optimal shapes established in lemma 1: starting from the top row of the grid, keep assigning the unassigned cells row-wise (interchanging left to right and then right to left) until the processor has A cells.

To compute the error bound in perimeter of the last p processors that were placed in the grid using this "orthogonal stripe filling" algorithm we compute the length of the boundary enclosing the region they occupy, plus the length of the border between processors, then subtract the lower bound. Thus, the maximum error in this region is

$$e < (2M + l_1 + l_2) + (2r - 1 + l_1 - l_2) + 2\left[(p-1)l_1 + (p-1)\right] - 2(p\left\lceil 2\sqrt{A}\right\rceil)$$

. The first six terms in the RHS of the inequality account for the left, right, top and bottom borders of this region, the next two terms account for the inner borders, and the last term is the lower bound. Note that the perimeter of the left border includes four terms $(M + (2r-1) + l_1 - l_2)$ because it is not a straight line. Thus the total relative distance of the perimeter of the solution from the lower bound satisfies:

$$\delta < \frac{2\left[wM + \frac{Md}{A} - p + (p-1)l_1 + p - 1 - p\left\lceil 2\sqrt{A}\right\rceil\right] + 2M + l_1 + l_2 + (2r-1) + l_1 - l_2}{2M\frac{wA+d}{A}\left\lceil 2\sqrt{A}\right\rceil}$$

or

$$\delta < \frac{wM + \frac{Md}{A} + (p-1)l_1 - 1 - p\left\lceil 2\sqrt{A}\right\rceil + M + l_1 + r}{M\frac{wA+d}{A}\left\lceil 2\sqrt{A}\right\rceil}.$$

But for all $A \geq 2$, $l_1 \leq \left\lfloor\sqrt{A}\right\rfloor + 2 \leq \left\lceil 2\sqrt{A}\right\rceil$ which implies that $(p-1)l_1 + l_1 = pl_1 \leq p\left\lceil 2\sqrt{A}\right\rceil$, and since $r \leq \frac{M}{\lfloor\sqrt{A}\rfloor}$ we get

$$\delta < \frac{M(wA+d) + MA + \frac{MA}{\lfloor\sqrt{A}\rfloor}}{M(wA+d)\left\lceil 2\sqrt{A}\right\rceil} = \frac{1}{\left\lceil 2\sqrt{A}\right\rceil} + \frac{A}{(wA+d)\left\lceil 2\sqrt{A}\right\rceil} + \frac{A}{(wA+d)\left\lceil 2\sqrt{A}\right\rceil\left\lfloor\sqrt{A}\right\rfloor}.$$

It's easy to show that $\forall x \geq 1 \;\; x^2 \leq \lceil 2x\rceil\lfloor x\rfloor$ so (since $A \leq M$, $A \leq N$)

$$\delta < \frac{2}{\left\lceil 2\sqrt{A}\right\rceil} + \frac{1}{N} < \frac{1}{\sqrt{A}} + \frac{1}{A}.$$

∎

The following theorem extends the previous discussion to the case in which P does not divide MN.

Theorem 4 *Assume P does not divide MN and that $P \geq max(M,N)$; the perimeter minimization problem MPE(M,N,P) has a feasible solution whose relative distance δ from the lower bound satisfies*

$$\delta < \frac{1}{\sqrt{A_1}} + \frac{1}{\sqrt{A_2}} + \frac{1}{A_1} \tag{4}$$

where $A_1 = \lfloor MN/P\rfloor$ and $A_2 = \lceil MN/P\rceil$. Thus the error bound δ converges to zero as A_1, A_2 (the areas of the processors) tend to infinity.

Proof: Decompose the grid among the P processors using the stripe decomposition and orthogonal stripe filling techniques discussed in the proof of theorem 3, initially assigning an area $A_1 = MN \div P$ to *all* P processors. Note that N can be written as $wA_1 + d$ for some naturals $w > 0$ and $0 \leq d < A_1$. The w MPE(M, A_1, M) problems cover the first wA_1 columns of the grid. The stripe filling is continued over the last d columns in the same manner as in the previous

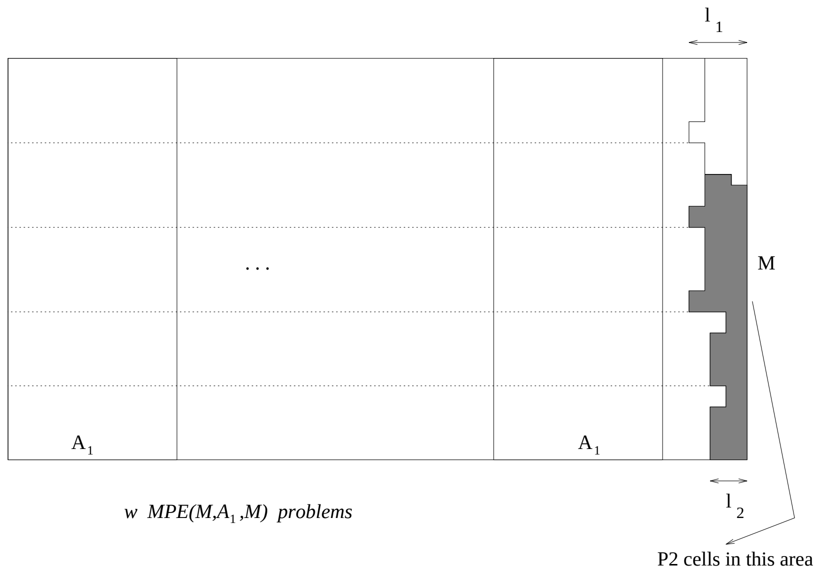

Figure 4: MPE(M, N, P), $P \geq max(M, N)$, $MN \mod P \neq 0$

theorem. Orthogonal stripe filling is used for the remaining processors (if there are any). This leaves $P_2 = MN \mod P$ grid cells unassigned near the bottom right corner of the grid (the gray area in figure 4). Assign each of these cells to the last P_2 processors; the perimeter of these cells is at most $4P_2$. Recall that the lower bound in perimeter is a non-decreasing function of the area of each processor, and therefore, the absolute perimeter error of the last P_2 processors (having area $A_2 = A_1 + 1$) can be no larger than the absolute error computed in the stripe-decomposition of their first A_1 cells plus 4 (for the extra cell). Thus, the relative distance of the perimeter of the partition from the lower bound is

$$\delta < \frac{1}{\sqrt{A_1}} + \frac{1}{A_1} + \frac{4P_2}{2P_2 \lceil 2\sqrt{A_2} \rceil}$$

(since the lower bound is $2(P_1 \lceil 2\sqrt{A_1} \rceil + P_2 \lceil 2\sqrt{A_2} \rceil) \geq 2P_2 \lceil 2\sqrt{A_2} \rceil$) and therefore

$$\delta < \frac{1}{\sqrt{A_1}} + \frac{1}{\sqrt{A_2}} + \frac{1}{A_1}.$$

∎

Essentially, the preceding theorems guarantee the existence of good quality solutions to the Minimum Perimeter problem as long as the number of processors is bigger than the dimensions of the grid. But the proof of the theorems also provide a method (stripe decomposition) for constructing such solutions. Furthermore, the theorems ensure the quality of the theoretical lower bounds.

The obvious drawback is that in environments where communication latencies can be high –like networks of workstations– the number of available processors may not be as large as the domain (which is assumed big enough to require the use of parallel processing for the efficient solution of the problem in hand). Nevertheless, the technique used to fill the last d columns of the grid in theorem 3, can be used to show that in the case where $N \leq P < M$ there exists a partition whose total perimeter approaches the lower bound (asymptotically) if $\frac{M}{PN}$ tends to zero. In particular, we have the following theorem:

Theorem 5 *If M, N and P satisfy $N \leq P < M$ and P divides MN, then the minimum perimeter problem MPE(M, N, P) has a feasible solution whose relative distance δ from the lower bound satisfies* $\delta < \frac{1}{\lceil 2\sqrt{A} \rceil} + \frac{1}{N} + \frac{A}{N\lceil 2\sqrt{A} \rceil}$.

Proof: It is shown in [CM95] that if $M \geq k(k-1)$, then there exist two natural numbers a, b such that $M = ak + b(k+1)$. Now, let k (as before) be $\lfloor \sqrt{A} \rfloor$. From the hypothesis, $P \geq N$, and since $A = \frac{MN}{P}$ we get $M \geq A \geq k^2 \geq k(k-1)$, so M can be written as $ak + b(k+1)$. This means that the grid can be decomposed exactly like the grid of figure 3 except of the first wA columns which do not exist. Following exactly the same arguments for computing the perimeter of the solution in the last d columns of figure 3 then, gives us

$$\delta < \frac{2\left[(P-p)+(p-1)l_1+(p-1)\right]+(2M+l_1+l_2)+(2r-1)+(l_1-l_2)-2p\left\lceil 2\sqrt{A}\right\rceil}{2\frac{MN}{A}\left\lceil 2\sqrt{A}\right\rceil}$$

from which, after substitution ($l_1 \leq \lceil 2\sqrt{A} \rceil$ and $r \leq \frac{M}{\lfloor \sqrt{A} \rfloor}$) we get

$$\delta < \frac{1}{\left\lceil 2\sqrt{A}\right\rceil} + \frac{1}{N} + \frac{A}{N\left\lceil 2\sqrt{A}\right\rceil}.$$

∎

It is easy to check that if M, N and P grow large in such a manner so that $\frac{M}{PN} \to 0$ then $\delta \to 0$ too. For example, by letting $P = N = A^{\frac{1}{2}+\epsilon}$, and $M = A$, then as A tends to infinity, $\delta \to 0$.

In the case where P does not divide MN, we have $A_1 = \lfloor \frac{MN}{P} \rfloor$ and because $\frac{N}{P} \in (0,1)$ we have $M > A_1 \geq k^2 > k(k-1)$ and thus the grid can be decomposed as shown in figure 4, except the first wA_1 columns which do not exist. Using the stripe decomposition method as described in theorem 4, we initially assign an area $A_1 = MN \div P$ to *all* processors, and fill the last bottom-right P_2 unassigned cells with the remaining cell of each of the P_2 processors. The error in perimeter of these last P_2 processors can be no larger than the absolute error computed in the stripe decomposition of their first A_1 cells plus four (for the extra cell). Using theorem 5, the relative distance of the perimeter of the partition from the lower bound is less than $\frac{1}{\lceil 2\sqrt{A_1} \rceil} + \frac{1}{N} + \frac{A_1}{N\lceil 2\sqrt{A_1} \rceil} + \frac{4P_2}{2P_2\lceil 2\sqrt{A_2} \rceil}$ (since the lower bound on perimeter is greater than $2P_2\lceil 2\sqrt{A_2} \rceil$) and so we have established the following

Theorem 6 *If M, N and P satisfy $N \leq P < M$ and P does not divide MN, then the minimum perimeter problem MPE(M, N, P) has a feasible solution whose relative distance δ from the lower bound satisfies* $\delta < \frac{1}{\lceil 2\sqrt{A_1} \rceil} + \frac{1}{N} + \frac{A_1}{N\lceil 2\sqrt{A_1} \rceil} + \frac{1}{\sqrt{A_2}}$ *where* $A_1 = \lfloor MN/P \rfloor$ *and* $A_2 = \lceil MN/P \rceil$.

Finally, note that there are instances of problems for which the best solution that can be found via the stripe-decomposition method are not optimal. For example, the MPE(17, 17, 17) has a provably optimal solution that achieves the lower bound (see figure 5) yet the best solution found by stripe-decomposition is at distance 0.65% from the lower bound. The optimal solution was found after a swapping heuristic was applied to a solution produced by PERIX-GA (to be discussed below).

5 Computational Results

Based on the observation that the partitioning problem can be viewed also as a tiling problem when restricted to the class of uniform 5-point grids, we developed PERIX, an algorithm that given a set of P optimal shapes from the appropriate library of optimal shapes attempts to tile the grid. To

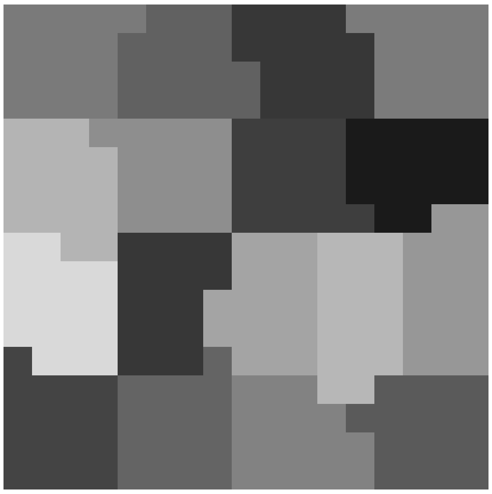

Figure 5: Optimal Solution for MPE(17, 17, 17)

achieve this goal, PERIX maintains a list of maximal free rectangles of the grid (a structure used successfully by Yackel in [YM92b] for another tiling problem), into which it attempts to place the next optimal shape, one at a time. The optimal shapes in our library are blocks accompanied by a fringe. PERIX attempts to place the block part of the optimal shape first, then attempts to place the fringe heuristically next to it with as little modification as possible.

To search the huge search space of input combinations to the PERIX algorithm (in order to find the best partition –not only the best stripe-form–) we have developed PERIX-GA, a high level repair Genetic Algorithm ([Hol92, Mic94]). PERIX-GA works with a population of individuals each of which is an array of shape indices to be tiled together by the PERIX algorithm. PERIX-GA breeds a population of such individuals for a certain number of generations using a modified crossover and mutation operator to take advantage of the fact that many good solutions of the problem are in stripe form (and therefore to encourage their appearance). Specifically, each allele in an individual has a tag associated with it, indicating whether the corresponding shape was placed at the beginning of a row or not. Crossover then, occurs at positions that that are marked by both parents as beginning of (possibly) a new stripe. Similarly, the mutation operator attempts to alter all "genes" with the same shape-index between two positions whose shapes are placed in the beginning of a new row. The parents may replace their offspring in the next generation if the offspring's fitness is worse than the worse fitness of the individuals of the parent generation or if the parents are the best individuals found so far in the evolution process (this elitist survival policy ensures the best individual found is not lost in subsequent generations and that the worse fitness value of the population monotonically improves).

We have run our algorithm (PERIX-GA), which is written in C, on a cluster of 33 SUN 20 SPARC-SERVER workstations (COW) using the host-node paradigm (one workstation serving as the host which co-ordinates the selection process, and the other 32 workstations being the nodes; all are connected via Ethernet). Each node maintains 2 individuals, thus the total population size is 64. The communication between workstations used the PVM 3.3.7 message-passing system ([GBD+94]) (before that, we had run earlier versions of our algorithm on a CM-5 with 32 nodes using the CMMD message-passing library ([Thi93]), and we reported the results in [CM95]). We run our algorithm for 20 generations except for the last case (1000×1000 grid) which we run for only 10 generations due to time limitations.

A metric of the size of the test problems is shown in table 1. The column "LIP" indicates the size of the problem formulated as a Mixed Linear Integer Program (MNP of the total variables are binary, the rest are continuous). The QAP dimension of each problem is MN, and the GA dimension is the length of the individuals, i.e. the number of processors to be assigned to the domain.

In table 2 we have compared our algorithm with a GRASP heuristic for the QAP ([LPR94]), run on one node of the cluster of workstations. This GRASP code is a state-of-the-art code, but as the dimension of the problem grows as the product of the dimensions of the grid, it has difficulties dealing with the larger problems in our test-suite.*

PROBLEM			QAP	LIP		GA	Stripe bnd
M	N	P	DIM	VARS	CONSTR	VARS	(%)
7	7	7	49	427	3584	7	16.66
13	13	13	169	2509	48854	13	12.50
17	17	17	289	5457	148274	17	11.11
32	30	64	960	63298	7492480	64	32.48
32	31	8	992	9857	108576	8	-
32	31	256	992	255873	125404128	256	141.06
100	100	8	10000	99800	1118808	8	-
101	101	101	10201	1050501	204030302	101	4.76
128	128	128	16384	2129664	528531584	128	4.34
200	200	200	40000	8079600	3.168E+09	200	3.44
256	256	256	65536	1.690E+07	8.523E+09	256	3.12
512	512	512	262144	1.347E+08	1.369E+11	512	2.17
1000	1000	1000	1000000	1.001E+09	1.996E+12	1000	1.56

Table 1: Test Problem Sizes under Various Formulations

PROBLEM			GRASP (COW: 1)		PERIX-GA (COW: 33)		
M	N	P	Err bnd(%)	Time	Err bnd(%)	Gens	Time
7	7	7	0.00	153.5	0.0	1	196.1
13	13	13	25.96	10327.6	0.0	15	227.8
17	17	17	-	-	0.0	9	268.6
32	31	8	-	-	2.17	2	201.0
32	31	256	-	-	0.0	5	230.2
101	101	101	-	-	0.04	15	219.1
200	200	200	-	-	0.0	7	261.0
512	512	512	-	-	0.66	5	402.3
1000	1000	1000	-	-	0.45	5	1660.5

Table 2: Computational Results: PERIX-GA and GRASP-QAP

We also compared our algorithm against two popular graph-partitioning methods, namely the (recursive) spectral bisection method ([PSL90]) with a Kernighan - Lin local refinement procedure applied, and the geometric mesh partitioning method ([MTTV93]). We obtained an implementation of the geometric mesh partitioner (in MATLAB) as described in [GMT95], and we used the Chaco package version 2.0 ([HL95a]) for the spectral bisection (Chaco is entirely written in ANSI C). We ran our experiments on these two graph partitioning methods on a SUN-20 workstation. Both of these methods require the number of processors to be a power of two, and we tabulated the results of the comparison from another test-suite in table 3. The column labeled "Time Best" indicates the time it took PERIX-GA to find the best solution. An asterisk in table 3 indicates the fact

*in the QAP literature, problems of dimension more than 30 are considered large, difficult problems.

that the partition found was not balanced (i.e. there were components that had at least two more nodes than other components). Also, note that for the last problem, both the geometric and the spectral method ran out of memory when trying to construct the adjacency matrix of the graph. The times are all in seconds. For this comparison, we ran PERIX-GA on an 9-node partition of

PROBLEM			SPECTRAL		GEOMETRIC		PERIX-GA		
M	N	P	Time	Err (%)	Time	Err (%)	Time	Time Best	Err(%)
32	31	8	1.8	6.52	43.6	5.43	84.0	20.9	2.71
32	31	256	4.3	6.73	152.3	-2.73*	80.4	4.1	0.00
32	30	64	3.0	6.25	90.4	6.25	50.9	43.2	0.00
100	100	8	9.0	9.33	111.0	7.39	81.9	12.3	2.64
128	128	128	85.5	14.13	539.9	7.13	67.6	16.9	1.65
256	256	256	227.8	13.25	3304.2	4.15	105.1	4.1	0.00
512	512	512	-	-	-	-	279.0	111.6	1.63

Table 3: PERIX-GA on 9 procs. vs Spectral and Geometric Partitioning

the COW (that is, 8 workstations of the cluster were serving as nodes, and one workstation was the host). The results show that the quality of the partition of our method is significantly better when the assumptions of theorem 3 are satisfied; but importantly, even when the assumption on the number of available processors is violated, our method still produces better partitions than spectral or geometric partitioning. The geometric mesh partitioning method proved to be rather fast on smaller problems, as its serial version gives times comparable to our algorithm running on 9 SUN-20 SPARC SERVER workstations except on the three largest problems. The (recursive) spectral bisection was the fastest method on many of the smaller problems. However, the quality of the resulting solutions is not as good as the quality of the partitions given by the other methods in the comparison test. It is also interesting to note that both the spectral bisection and the geometric partitioner fail to find the (provably optimal) partition of the 256×256 grid partitioned among 256 processors which is simply 256 squares of size 16×16 tiled together. This particular test-problem is the only one in our suite for which an optimal solution was known to exist a-priori.

6 Conclusions and Future Directions

Stripe decomposition is a fast and efficient method for constructing very good quality partitions of rectangular grids among processors as long as the number of processors is bigger than both dimensions of the grid. We have developed PERIX, an algorithm that tiles together an input set of near rectangular optimal shapes while seeking to minimize modification of the input shapes. (For suitable inputs, a stripe decomposition is produced). PERIX-GA is a parallel Genetic Algorithm that runs on a network of workstations and searches the space of inputs to PERIX with genetic operators that encourage the occurrence of stripe-form solutions. The computational results indicate that problems that are intractable for many methods because of their size, can be solved by PERIX-GA with good accuracy within minutes on a network of workstations. The quality of the partitions produced by our code is superior to the quality of the solutions provided by other popular codes.

Recently, we extended PERIX to work on arbitrary uniform grids and one immediate goal is to evaluate the performance of our algorithm in general grids. First results on elliptical domains look very promising. Improving the quality of the lower bounds is another goal. Finally, we would like to find ways to relax the assumption on the number of available processors in the theorems on stripe decomposition. This relaxation should be possible as indicated by the computational results for problems with small number of processors.

7 Acknowledgments

We would like to thank P.M. Pardalos and M. Resende for providing us with a version of their high-quality GRASP heuristic for the QAP. We obtained the MATLAB code for the geometric mesh partitioner from the anonymous ftp site indicated in Gilbert et al. ([GMT95]) to whom we are thankful for the easy access to their code. Finally, our thanks to B. Hendrickson and R. Leland for providing us with version 2.0 of the Chaco package.

References

[CM95] I. T. Christou and R. R. Meyer. Optimal equi-partition of rectangular domains for parallel computation. Technical Report MPTR 95-04, University of Wisconsin - Madison, February 1995. To appear in the *Journal of Global Optimization*.

[CQ95] P. Crandall and M. Quinn. Non-uniform 2-d grid partitioning for heterogeneous parallel architectures. In *Proceedings of the 9th International Symposium on Parallel Processing*, pages 428–435, 1995.

[DTR91] R. DeLeone and M. A. Tork-Roth. Massively parallel solution of quadratic programs via succsessive overrelaxation. Technical Report 1041, University of Wisconsin - Madison, August 1991.

[GBD+94] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam. *PVM 3 User's Guide and Reference Manual.* Oak Ridge National Laboratory, 1994.

[GMT95] J. R. Gilbert, G. L. Miller, and S. H. Teng. Geometric mesh partitioning: Implementation and experiments. In *Proceedings of the 9th International Symposium on Parallel Processing*, pages 418–427, 1995.

[HL95a] B. Hendrickson and R. Leland. *The Chaco User's Guide Version 2.0.* Sandia National Laboratories, July 1995.

[HL95b] B. Hendrickson and R. Leland. An improved spectral graph partitioning algorithm for mapping parallel computations. *SIAM J. on Sci. Comput.*, 16:452–469, 1995.

[Hol92] John Holland. *Adaptation in Natural and Artificial Systems.* MIT Press, 1992.

[KL70] B. W. Kernighan and S. Lin. An effective heuristic procedure for partitioning graphs. *Bell Systems Tech. Journal*, pages 291–308, February 1970.

[LFE94] M. Laguna, T. A. Feo, and H. C. Elrod. A greedy randomized adaptive search procedure for the two - partition problem. *Operations Research*, July - August 1994.

[Lin91] K. Y. Lin. Exact solution of the convex polygon perimeter and area generating function. *J. Phys. A. Math Gen.*, 24:2411–2417, 1991.

[LPR94] Y. Li, P. M. Pardalos, and M. G. C. Resende. A grasp for the qap. In P. M. Pardalos and H. Wolkowicz, editors, *Quadratic Assignment and Related Problems.* DIMACS Series Vol. 16, American Mathematical Society, 1994.

[Mel94] M. Bousquet Melou. Codage des polyominos convexes et equation pour l'enumeration suivant l'aire. *Discrete Applied Mathematics*, 48:21–43, 1994.

[Mic94] Zbigniew Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs.* Springer-Verlag, 1994.

[MTTV93] G. L. Miller, S. H. Teng, W. Thurston, and S. A. Vavasis. Automatic mesh partitioning. In A. George, J. R. Gilbert, and J. W. H. Liu, editors, *Graph Theory and Sparse Matrix Computation*. Springer-Verlag, 1993.

[PRW93] P. M. Pardalos, F. Rendl, and H. Wolkowicz. The quadratic assignment problem: A survey and recent developments. In P. M. Pardalos and H. Wolkowicz, editors, *Quadratic Assignment and Related Problems*. American Mathematical Society, 1993.

[PSL90] A. Pothen, H. D. Simon, and K. P. Liu. Partitioning sparse matrices with eigenvectors of graphs. *SIAM Journal on Matrix Analysis and Applications*, 11:430–452, 1990.

[Sch89] R. J. Schalkoff. *Digital Image Processing and Computer Vision*. John Wiley & Sons, 1989.

[Thi93] Thinking Machines Corporation. *CMMD Reference Manual*, May 1993.

[vL91] von Laszewski. Intelligent structural operators for the k-way graph partitioning problem. In R. Belew and L. Booker, editors, *Proceedings of the Fourth Intl. Conference on Genetic Algorithms*, pages 45–52. Morgan Kaufmann Publishers, Los Altos, CA, 1991.

[Yac93] J. Yackel. *Minimum Perimeter Tiling in Parallel Computation*. PhD thesis, University of Wisconsin - Madison, August 1993.

[YM92a] J. Yackel and R. R. Meyer. Minimum perimeter decomposition. Technical Report 1078, University of Wisconsin - Madison, February 1992.

[YM92b] J. Yackel and R. R. Meyer. Optimal tilings for parallel database design. In P. M. Pardalos, editor, *Advances in Optimization and Parallel Computing*, pages 293–309. North - Holland, 1992.

[YMC95] J. Yackel, R. R. Meyer, and I. T. Christou. Minimum-perimeter domain assignment, February 1995. Submitted to the Math. Programming Journal.