Optimal Equi-Partition of Rectangular Domains for Parallel Computation

Ioannis T. Christou *   Robert R. Meyer †

February 28, 1995

Abstract

We present an efficient method for the partitioning of rectangular domains into equi-area sub-domains of minimum total perimeter. For a variety of applications in parallel computation, this corresponds to a load-balanced distribution of tasks that minimize interprocessor communication. Our method is based on utilizing, to the maximum extent possible, a set of optimal shapes for sub-domains. We prove that for a large class of these problems, we can construct solutions whose relative distance from a computable lower bound converges to zero as the problem size tends to infinity. PERIX-GA, a genetic algorithm employing this approach, has successfully solved to optimality million-variable instances of the perimeter-minimization problem and for a one-billion-variable problem has generated a solution within 0.32% of the lower bound. We report on the results of an implementation on a CM-5 supercomputer and make comparisons with other existing codes.

1 The Minimum Perimeter Problem

We consider the Minimum Perimeter Equi-partition problem MPE(M, N, P), a geometric problem with intrinsic beauty that finds numerous applications in parallel computing. It is essentially a graph partitioning problem that, when restricted to rectangular grids (the main focus of this paper), can be stated as follows: given a rectangular grid of dimensions $M \times N$ and a number of processors[1] P, where P divides MN, find the partition of the grid that minimizes the total perimeter induced subject to the constraint that each processor is assigned the same number of grid cells. Geometrically, the problem may be thought of as partitioning the grid into P equi-area regions (each of area $A := MN/P$) of minimum total perimeter.

Since graph partitioning is itself a special case of a more general problem, the so-called Quadratic Assignment Problem (QAP), it follows that MPE can be formulated as a QAP ([PRW93]). In terms of binary variables in an integer programming formulation ([NW85]) the problem may be described

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

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

[1]The words *processor* and *component* will be used interchangeably

using MNP variables and $MN+P$ constraints:

$$\text{(1)} \qquad min. \sum_{i,i'=1}^{M} \sum_{j,j'=1}^{N} \sum_{p,p'=1\ p\neq\ p'}^{P} c_{iji'j'} x_{ij}^{p} x_{i'j'}^{p'}$$

$$s.t. \begin{cases} \sum_{i=1}^{M} \sum_{j=1}^{N} x_{ij}^{p} = \frac{MN}{P} & p = 1 \dots P \\ \sum_{p=1}^{P} x_{ij}^{p} = 1 & i = 1 \dots M \quad j = 1 \dots N \\ x_{ij}^{p} \in \mathbf{B} = \{0,1\} & \end{cases}$$

$$\text{where } c_{iji'j'} = \begin{cases} 1 & \text{if } |i-i'| = 1 \text{ and } j = j' \\ 1 & \text{if } |j-j'| = 1 \text{ and } i = i' \\ 0 & \text{else} \end{cases}$$

This formulation has an objective function that is the sum of quadratic terms of boolean variables. At the expense of introducing more variables, and letting $\mathcal{I}$ denote the set of pairs of adjacent cells, we can reformulate the problem as a mixed linear integer program:

$$\text{(2)} \qquad min. \sum_{i,i'=1}^{M} \sum_{j,j'=1}^{N} \psi_{iji'j'}$$

$$s.t. \begin{cases} \sum_{i=1}^{M} \sum_{j=1}^{N} x_{ij}^{p} = \frac{MN}{P} & p = 1 \dots P \\ \sum_{p=1}^{P} x_{ij}^{p} = 1 & i = 1 \dots M \quad j = 1 \dots N \\ \psi_{iji'j'} \geq x_{ij}^{p} + x_{i'j'}^{p'} - 1 & ((i,j),(i',j')) \in \mathcal{I}, \quad p \neq p' = 1 \dots P \\ x_{ij}^{p} \ \in \mathbf{B} = \{0,1\} & \end{cases}$$

The above is a mixed integer program with $2MN-M-N$ continuous and MNP binary variables and $P(P-1)(2MN-M-N)+MN+P$ constraints. We do not attempt to solve MPE(M,N,P) using this formulation; however, we do report on comparisons between our approach and other exact methods based on Branch & Bound type algorithms [PRW93, PRRL94] for the QAP.

The minimum perimeter problem has many applications in scientific computing in parallel systems, e.g. in the solution of PDEs where a partial differential equation must be solved numerically on a grid. Such computations often require communication between each cell and its North, South, East, and West neighbors (see [DTR91]). Given a parallel/distributed computing environment, one is faced with the task of assigning to each processing element a group of grid cells subject to load balancing constraints (each processing element gets exactly the same number of grid cells) so that total inter-processor communication is minimized. As the trend in parallel computing is towards clusters of workstations where the communication between processors can be very expensive, it is important that good solutions to the minimum perimeter problem be provided. Another application is edge detection in computer vision and digital image processing employing *parallel* computations [Sch89].

2 Optimal Shapes and Lower Bounds

As Yackel and Meyer have shown in [YM92a] , there exists a mapping $\mathcal{L}$ from the set of natural numbers to the set of all sets of configurations of grid cells such that every natural number A is mapped onto a library of configurations, called the optimal shapes for A, where each shape that belongs to $\mathcal{L}(A)$ has exactly A cells and its perimeter is

$$\text{(3)} \qquad \Pi^{*}(A) = 2\lceil 2\sqrt{A} \rceil$$

The perimeter of these shapes is optimal, in the sense that there exists no configuration of A cells having a total perimeter less than $\Pi^*(A)$. Given an instance MPE(M, N, P) with $A = \frac{MN}{P}$ it follows that if any P shapes of $\mathcal{L}(A)$ can be tiled together so as to completely cover the original domain (i. e. the whole $M \times N$ grid), then the partition induced by these shapes is optimal. In any case, (3) yields *a lower bound* $P\Pi^*(A)$ *on the objective value* of our problem. Computational experience shows that this lower bound is tight for many problems, but not for all instances. For example, consider the MPE$(1, N, 2)$ with N even (shown in figure 1); the optimal partition has a total perimeter equal to $4(N/2+1)$ while the lower bound is $4\left\lceil 2\sqrt{N/2}\right\rceil$. Clearly, the relative distance defined by the ratio of the difference of the solution minus the one predicted from the lower bound over the one predicted grows as $\sqrt{N}$. This gap is due to the fact that the lower bound assumes domains large enough in both dimensions so as to fit the relatively square optimal shapes.

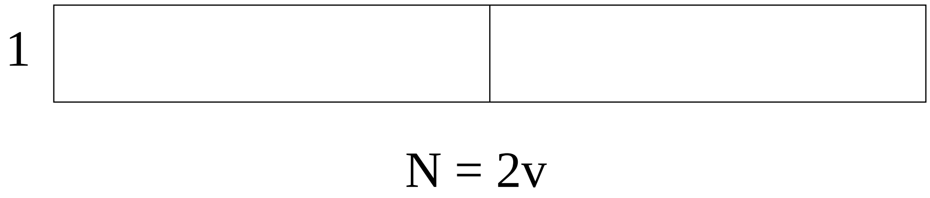

Figure 1: Optimal Partition for the MPE$(1, N, 2)$

But the lower bound can fail to be attained even for square domains, as is the case for the MPE(5,5,5), an optimal partition of which is shown in figure 2. However, for relatively square domains, we will show that the lower bound is good in an asymptotic sense.

Figure 2: Optimal Partition for the MPE(5,5,5)

Many of the optimal shapes are rectangles with a "fringe" attached to one of their sides ([YM92b]), so they can be characterized by three numbers, namely the dimensions of the rectangle h, w and the size of the fringe f. In general, the number of such near-rectangular optimal shapes is of order $A^{1/4}$, but this does not encompass all possible minimum perimeter configurations. There is a lot of literature (see for example [Lin91, Mel94]) dealing with the generating function approach for developing expressions for the exact number of "convex polyominoes" with various properties. However, our algorithm (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. Such a shape can be generated using the following technique: start with a rectangle that has perimeter $\Pi^*(A)$ and area at least A. Iteratively remove corner cells of this rectangle until the area of the remaining object is exactly A. The remaining object is an optimal shape for A. It turns out that all the optimal shapes for a given area size A can be constructed using this technique (see Yackel's PhD thesis [Yac93]).

In much of the analysis below, rather than dealing directly with perimeter, it is more convenient to use the concept of semi-perimeter. Given a group of A connected cells, the semi-perimeter $\mathcal{S}(\mathrm{g})$ of this group is defined to be the width plus the height of the smallest rectangle enclosing the group. It is easy to show that for all optimal shapes, perimeter is twice the semi-perimeter.

3 Error Bounds for Selected Classes of Domains

A key observation is that for many instances of the problem, a stripe-decomposition of the domain is possible; that is, an optimal –or near optimal– partition exists, where the sub-domains form horizontal stripes of height approximately $\sqrt{A}$ that partition the rows of the grid (observe the stripes of figure 3). To establish this claim, we are going to prove two lemmas; these two lemmas combined, guarantee the existence of such solutions for a large class of instances of the perimeter minimization problem.

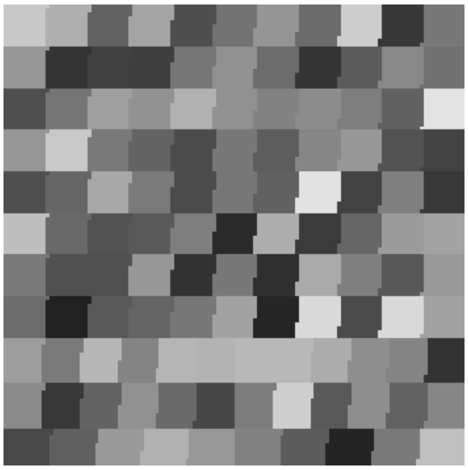

Figure 3: Optimal Partition for the MPE(122,122,122)

Lemma 1 *Given two nonnegative integers m, k there exist natural numbers a, b such that*

$$m = ak + b(k+1) \tag{4}$$

iff $r_m = 0$ *or*

$$\begin{gathered} k \leq d_m + r_m \\ \textit{where } d_m = m \div (k+1) \\ \textit{and } r_m = m \mod (k+1) \end{gathered} \tag{5}$$

Proof: The case $r_m = 0$ is trivial, so in the arguments below, we assume $r_m > 0$. From the definition of d_m and r_m we have

$$m = d_m(k+1) + r_m$$

But this can be written as

$$m = (d_m - c)(k+1) + (r_m + c + kc)$$

for any c. Now, we can write m in the desired form (4) if k divides $r_m + c$ and $0 \leq c \leq d_m$. The smallest c that satisfies this requirement is $k - r_m$, and thus, if $c = k - r_m \leq d_m$ then simply set $b = d_m - (k - r_m)$ $a = \frac{r_m + (k+1)(k - r_m)}{k} = k + 1 - r_m$. For the other direction, observe that $k - r_m$ is the smallest c that allows k to divide the number $r_m + (k+1)c$ so if this c is greater than d_m, there exists no natural such that the required decomposition is possible. ∎

A useful corollary of this lemma is the following:

Corollary 2 *Given two nonnegative integers m, k there exist natural numbers a, b such that equation 4 holds if $m \geq k(k-1)$.*

Proof: The corollary trivially holds for $k = 0$ or $k = 1$. Assume therefore $k \geq 2$. If $k(k-1) \leq m \leq k^2 - 1$, then $m = k^2 - r$ for some $r = 1 \ldots k$, and thus write $m = (r-1)k + (k-r)(k+1)$. Else $m \geq k^2$. For all m between k^2 and $k(k+1) - 1$ we have $d_m = k - 1$ and $r_m \geq 1$, so $k \leq d_m + r_m$ and the claim holds. For all m greater than or equal to $k(k+1)$ we have $d_m \geq k$ and so $k \leq d_m + r_m$ and the claim holds true again. ∎

The next lemma states that the class of problems MPE(N, N, N) is amenable to such decomposition. In other words, for all $N > 0$, we can partition the rows of the grid with a number of stripes, each of which has a height that is equal to the height of an optimal shape from the library $\mathcal{L}(N)$.

Lemma 3 *Given N, we can always find r shapes (h_i, w_i, f_i) –not necessarily distinct– from the library of optimal shapes $\mathcal{L}(N)$ such that*

$$\sum_{i=1}^{r} h_i = N$$

.

Proof: We are going to show that we can always find two optimal shapes (h_1, w_1, f_1) and (h_2, w_2, f_2) where $f_1 < h_1$, $f_2 < h_2$, such that $ah_1 + bh_2 = N$ for some natural numbers a, b. Let $k = \left\lfloor \sqrt{N} \right\rfloor$.

- Assume $k(k+1) > N$. The previous discussion of this section implies that $(k, k, N - k^2)$ is an optimal shape and its semi-perimeter is $2k+1$ (unless $N = k^2$ in which case the semi-perimeter is $2k$, and we can get a perfect partition using the optimal shape $(k, k, 0)$). Furthermore, trying the rectangle $(k+1, k-1)$ we get

$$(k+1)(k-1) = k^2 - 1 < N$$

  and $f = N - k^2 + 1 < k + 1$ because $N < k(k+1)$ so the shape $(k+1, k-1, N - k^2 + 1)$ is also an optimal shape. Both of these optimal shapes have fringe size less than the height of the corresponding block. Since $N \geq k^2$, by corollary 2 we can find two naturals a, b, such that $N = ak + b(k+1)$.

- Next assume that $k(k+1) = N$. This simply means that the MPE(N, N, N) has an optimal shape that is a rectangle and thus we can obtain a perfect partition using N rectangles of dimensions $k \times (k+1)$ all oriented in the same way.

- Finally, assume $k(k+1) < N$. Observe that $(k+1)^2 > N$ from the definition of k. Now, the shapes $(k+1, k, f)$ and $(k, k+1, f)$ where $f = N - k(k+1) < k+1$ belong to $\mathcal{L}(N)$. Again, because $N > k^2$ corollary 2 applies and the required decomposition of the rows of the grid is possible. Note that if $f = k$ the rectangle $k \times (k+2)$ is an optimal shape, and a perfect partition using N such rectangles all oriented the same way is possible.

∎

Lemma 3 proves that for the MPE(N,N,N) we can partition the rows of the grid into r stripes of height h, where h is the height of an optimal shape for the problem. Motivated by this fact, we present next a general *stripe-filling* process, which, given an optimal shape (h,w,f) of area A and a stripe of height h and width A fills the stripe with exactly h such shapes assigning them processor indices $1\ldots h$. We first state this in the form of pseudo-code and then describe it in more detail.

```
stripe_fill(h,A:integer; var str: grid)
/* input: h,A - the dimensions of the stripe
   output: str - the processor index assignments of the cells
*/
begin
proc = 1;
area[proc] = 0;
for col = 1 to A
  for row = 1 to h
    str[row,col] = proc;
    area[proc] = area[proc] + 1
    if (area[proc] = A)
      proc = proc + 1;
      area[proc] = 0
    endif
  endfor
endfor
end;
```

The effect of this process is to place the block-part of the current optimal shape so that its leftmost column occupies the first completely unassigned column of the stripe. If there exist any unassigned cells in column to the left of this column, the method places as much of the fringe as possible there. If there is some part of the fringe that does not fit there, the algorithm places this remainder in the immediate right neighboring column of the block. However, if all fringe cells are assigned to the left and there remain neighboring cells on the left that are not assigned, the algorithm alters the shape by removing cells from the rightmost column of its block and using them to fill the residual left neighbors of the block. Figure 4 illustrates the placement of the first two shapes of a given input string for the MPE(17, 17, 17) problem (grid dimensions are 17×17 to be partitioned among 17 processors) within a stripe. Note that each of these shapes is a 4×4 rectangle accompanied by a fringe cell. The second shape has been modified by the stripe-filling process, but its total perimeter is still optimal (equal to 18).

By using the stripe-decomposition and stripe-filling results, we will show that for large classes of MPEs, there exist solutions whose relative distance from the theoretical lower bound converges to zero as the problem size tends to infinity. One such class of problems is MPE(M,N,M) where $M\geq N$. This error bound behavior is a clear indication of the quality of the theoretical lower bound we are using. The proof of the theorem is by construction, meaning that we present a fast algorithm that computes such approximate solutions.

Theorem 4 *The MPE(N,N,N) problem (partition an $N\times N$ grid into N components) has a solution whose relative distance δ from the lower bound satisfies*

$$\delta < \frac{1}{\lceil 2\sqrt{N}\rceil} \qquad (6)$$

Proof: Let (h_i,w_i,f_i) denote an optimal shape of height h_i, width w_i and fringe size f_i for area size $A=\frac{N\times N}{N}=N$ from the library of optimal shapes $\mathcal{L}(N)$, where $f_i<h_i$.

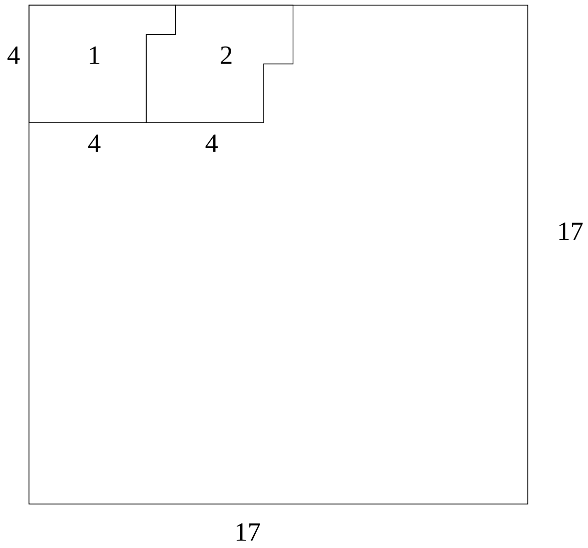

Figure 4: Placement of the initial two shapes

As we have already shown in lemma 3, we can always find r shapes –not necessarily distinct– such that $\sum_{i=1}^{r} h_i = N$.

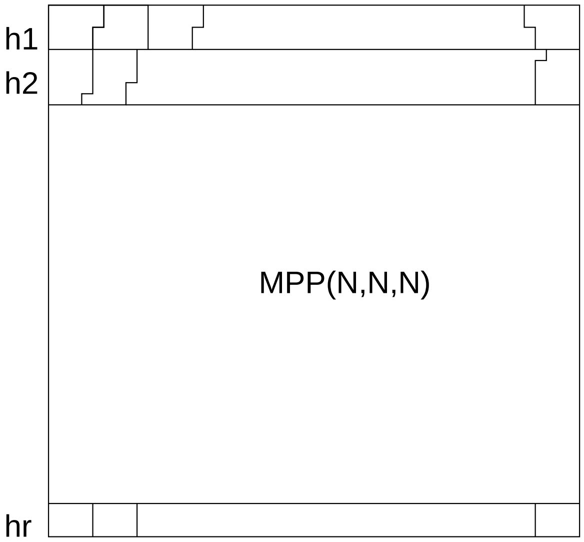

Figure 5: Stripe Form of the Partition

These numbers $h_1, \ldots, h_r$ induce a partition on the rows of the grid (see figure 5). The first h_1 rows of the grid are called the first stripe, the following h_2 rows are called the second stripe etc.

Now, each stripe, say stripe-i, can be filled with h_i components using the shape (h_i, w_i, f_i). In order to do this simply use the "stripe-filling" algorithm described in section 2. In this manner, stripe-i is filled using exactly h_i components, because the area of the stripe is $h_i N$, and the total area of h_i components is $h_i N$ also.

If $f_i = 0$ then no error occurs in stripe-i. If $f_i > 0$, the error in this stripe can be no more than $f_i - 1$. To see this observe that each shape is either optimally placed (if it occupies $w_i + 1$ columns of the stripe) or its semi-perimeter is suboptimal by 1 (if the fringe part of the shape is split between the immediate left and right columns of the block). So, we can measure the error in the stripe by counting the number of the suboptimal shapes, or equivalently, by counting the "surplus" columns corresponding to regions that occupy $w_i + 2$ columns.

In the stripe, assume there are e_0 shapes that fill completely $w_i - 1$ columns of the stripe, and occupy part of their immediate left and right neighboring columns, e_0^+ shapes that fill w_i columns of the stripe and occupy part of one immediate neighboring column, and e_i shapes that are suboptimal , that is they fill w_i columns of the stripe, and they occupy part of both their immediate neighboring columns. Thus, letting t denote the number of columns containing more than one component index, we have

$$e_0 + e_0^+ + e_i = h_i \tag{7}$$

$$h_i w_i + f_i = N \tag{8}$$

$$e_0(w_i - 1) + e_0^+ w_i + e_i w_i + t = N \tag{9}$$

from which, after substitution, we conclude that

$$t = f_i + e_0$$

Let us now associate each of the t doubly indexed columns with the component corresponding to the block to its left. Then the shapes corresponding to e_0 each contribute to t as do the e_i suboptimal shapes and the first e_0^+ shape at the left end of the stripe. Therefore, $e_0 + 1 + e_i \leq t$. Combining this with the preceding equation implies

$$e_i \leq f_i - 1$$

Therefore, the semi-perimeter error in each stripe is not more than $f_i - 1$ and the stripes cover the grid completely and with no overlap using a total of $\sum_{i=1}^{r} h_i = N$ components, so the relative error is bounded by

$$\delta \leq \frac{1}{N\left\lceil 2\sqrt{N}\right\rceil} \sum_{i=1}^{r}(f_i - 1)$$

Defining $\pi_i \equiv \frac{f_i}{h_i}$, $\pi = \max_i \pi_i$, we have $\pi_i \in [0, 1)$, $\pi \in [0, 1)$ so substituting in the above we get

$$\delta \leq \frac{1}{N\left\lceil 2\sqrt{N}\right\rceil} \sum_{i=1}^{r}(f_i - 1) = \frac{1}{N\left\lceil 2\sqrt{N}\right\rceil} \sum_{i=1}^{r}(\pi_i h_i - 1) \leq \frac{\pi N - r}{N\left\lceil 2\sqrt{N}\right\rceil} < \frac{1}{\left\lceil 2\sqrt{N}\right\rceil}$$

∎

Before we generalize theorem 4, it is worth focusing on it. The theorem shows that the quality of the theoretical lower bound we are using must be very good as there exist solutions whose total perimeter differs only "slightly" from the lower bound as the problem size gets larger. Furthermore, the constructive proof we have given implies a fast algorithm for constructing such good approximate solutions. In fact, this technique is the basis of the PERIX algorithm that we describe in sec. 4.

Note also that in the case when the fringe f_i of an optimal shape (h_i, w_i, f_i) divides its height h_i exactly, then there can be no surplus columns and therefore the error in a stripe using this shape is zero. The same zero error behavior of stripes occurs when $f_i \leq 1$. This implies that it is not unlikely in the best near-optimal solutions to observe a large number of stripes of zero total error.

We now prove a generalized version of the theorem for the class of problems MPE(M, N, M) where $M \geq N$.

Theorem 5 *The MPE(M,N,M) with $M \geq N$ has a solution whose total perimeter possesses a relative distance δ from the lower bound that satisfies*

$$\delta < \frac{1}{\left\lceil 2\sqrt{N} \right\rceil} \tag{10}$$

Proof: The proof is very similar to the proof of theorem 4. We are going to partition the M rows of the grid into $r_1 + r_2$ stripes having lengths $h_1, \ldots, h_{r_1}$, $h_1^s, \ldots, h_{r_2}^s$. The first r_1 stripes will be filled with optimal shapes (h_i, w_i, f_i) while the last r_2 stripes will use sub-optimal shapes (h_i^s, w_i^s, f_i^s). These sub-optimal shapes have an area size equal to N, but their semi-perimeter is $\mathcal{S}(N) + 1$ (off by 1). The only case in which we use these shapes is when N is a perfect square $N = k^2$. In this case, the sub-optimal shape we are going to use is the shape $(k+1, k-1, 1)$ which has an area of $(k+1)(k-1) + 1 = N$ and a semi-perimeter equal to $2k+1$.

Let $k = \left\lfloor \sqrt{N} \right\rfloor$.

- Assume first $k(k+1) > N$. Furthermore, assume $N \neq k^2$. Under these assumptions, the shapes $(k, k, N - k^2)$ and $(k+1, k-1, N - k^2 + 1)$ can be used to partition the grid. Applying the technique described in the previous theorem, in the i-th stripe we can place h_i shapes, and the error in each of them is

  $$e_i \leq f_i - 1$$

  It only remains to prove that we can find nonnegative integers a, b such that

  $$M = ak + b(k+1) \tag{11}$$

  But since $M \geq N \geq k^2$, from corollary 2, we have that equation 11 holds.

  Now, assume that $N = k^2$. In this case, the shape $(k+1, k-1, 1)$ is sub-optimal by 1 as its semi-perimeter is $2k+1$. Nevertheless, the area size of this shape is N, and we can use $k+1$ shapes to fill a stripe of height $h_i^s = k+1$. The absolute error in such a stripe will be

  $$e_i = h_i^s = k + 1.$$

  where the term h_i^s comes from the fact that each shape used in this stripe has a semi-perimeter that is suboptimal by one. Note that since $f_i^s = 1$, there exist no surplus columns in such a stripe. From the discussion above, we have that $M = ak + b(k+1)$ for some $a, b \in \mathbb{N}$, so we can partition the rows of the grid as desired. Now setting $r_1 = a$ and $r_2 = b$ we have that the total relative distance must be

  $$\delta \leq \frac{1}{M \left\lceil 2\sqrt{N} \right\rceil} \sum_{i=1}^{r_2} h_i^s$$

  and since

  $$\sum_{i=1}^{r_2} h_i^s = M - r_1 k$$

  we get

  $$\delta \leq \frac{1}{M \left\lceil 2\sqrt{N} \right\rceil} \left[M - \left\lfloor \sqrt{N} \right\rfloor r_1 \right]$$

- Next, assume that $N = k(k+1)$. This means that $(k+1, k, 0)$ and $(k, k+1, 0)$ are optimal rectangles. Since $M \geq N \geq k^2$ by corollary 2 we can always write $M = ak + b(k+1)$ for some $a, b \in \mathbb{N}$. Note, that the error in each stripe is zero, which results in a perfect partition.

- Finally, in the case $N > k(k+1)$, the shapes $(k+1, k, f)$ and $(k, k+1, f)$ are optimal shapes for the MPE(M, N, M). Note that $f = N - k(k+1)$, and if $f = k$ then the shape $(k, k+1, k)$ is really the optimal rectangle $(k, k+2, 0)$. Using the same arguments again, we can partition the rows of the grid by finding $a, b \in \mathbb{N}$ such that $M = ak + b(k+1)$. The error in the i-th stripe will be
$$e_i \leq f_i - 1$$

So we have shown that in all cases there exists a solution whose total perimeter has a relative distance from the theoretical lower bound that is

$$\delta \leq \frac{\pi M - r}{M\left\lceil 2\sqrt{N}\right\rceil}$$

for some $\pi \in [0, 1)$ and $r \in \mathbb{N}$ ∎

The next theorem, based on the previous discussions, asserts that there exist solutions to the *general* MPE(M, N, P), whose relative distance from the lower bound approaches zero as the problem size tends to infinity as long as the number of components is big enough, i.e., $P \geq max(M, N)$.

Theorem 6 *If $P \geq max(M, N)$ then the perimeter minimization problem MPE(M,N,P) has a solution whose relative distance δ from the lower bound satisfies*

$$\delta < \frac{9}{\left\lceil 2\sqrt{A}\right\rceil} \tag{12}$$

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

Proof:

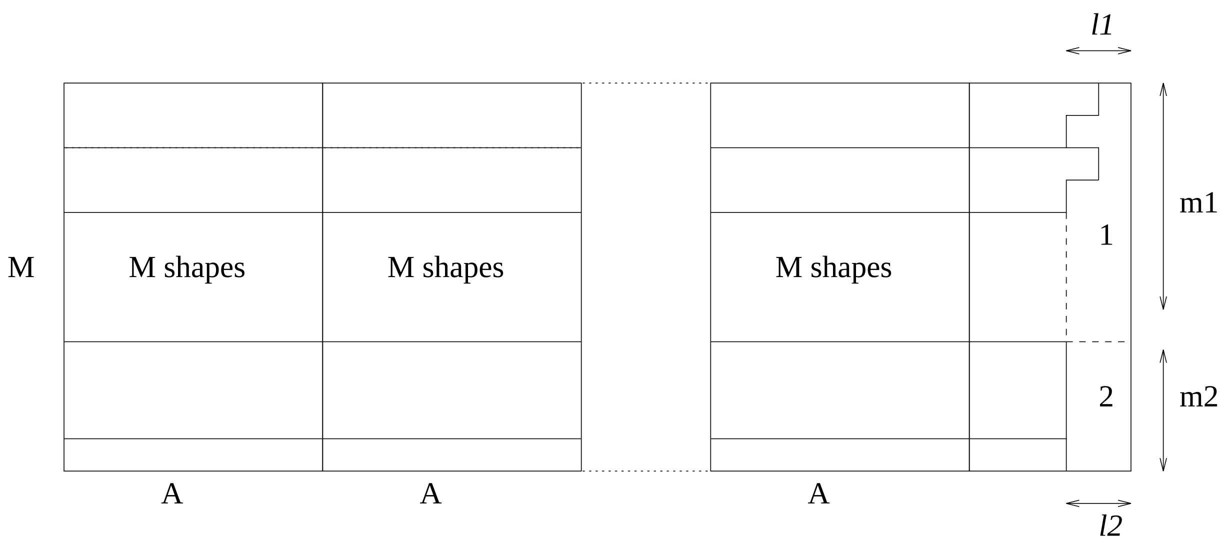

Figure 6: MPE(M, N, P), $P \geq max(M, N)$

The grid is shown in figure 6. Write $N = wA + d$ for some naturals $w \geq 1$ and $d < A$. Define $k = \left\lfloor \sqrt{A} \right\rfloor$. 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 problems MPE(M, A, M), use the techniques employed in the proof of the previous theorem to get a total absolute perimeter error $e < 2wM$. This striping technique (which partitions the rows of the grid into $r \leq M/k$ components $h_1, ..., h_r$) is continued over the last d columns in each stripe until no shape can be placed in this fashion.

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.

When this striping process ends, in area i , $i = 1, 2$, there remain at most $l_i \leq k+2$ columns that contain unassigned grid cells (see fig. 6), and the first such column, might have "slots" of empty cells in it. Use p^s shapes to fill these slots (for the last of these shapes we might have to place part of it in the rest of the free area). Each of these shapes will possess a perimeter no worse than $4A$. The number p^s will satisfy $(p^s - 1)A < M$.

To fill the remaining area we will use a "reverse-stripe-filling" algorithm: keep filling the cells of this rightmost area of the grid with the remaining components, one at a time row-wise: fill the n-th row before filling the (n+1)-st row.

Let p_i denote the number of shapes that are placed completely in area i, and m_i the number of rows in area i that are used by these shapes. In the worst case, there may exist one shape that has parts of it placed in both areas. Since we are interested in upper bounding the relative error of the solution we construct, we will assume the existence of such a shape (in any case, its perimeter may not be more than $4A$). Now, the number of shapes that were placed in the last d columns of the grid by continuing the stripe-filling process are $\frac{Md}{A} - p$ where $p = p_1 + p_2 + p^s + 1$. The total perimeter error incurred by these shapes is no more than $2(\frac{Md}{A} - p)$.

Now, from the above definitions it is clear that

$$p_i A \leq (l_i - 1)m_i$$

and that $m_1 + m_2 \leq M$. In the worst case $l_i > 1$ (which means that in area i, there exists at least one completely free column; otherwise, this area is completely filled after the placement of the p^s shapes, and $p_i = 0$). The perimeter of each shape placed in area i, can be upper bounded as follows; it takes at most $\left\lceil \frac{A}{l_i - 1} \right\rceil$ rows to place it because $\left\lceil \frac{A}{l_i - 1} \right\rceil (l_i - 1) \geq A$. Its perimeter therefore, is $2(\left\lceil \frac{A}{l_i-1} \right\rceil + l_i - 1)$.

Thus, the maximum deviation from the perimeter bound is less than

$$2\left[wM + \frac{Md}{A} - (p_1 + p_2 + p^s + 1) + \sum_{i=1}^{2} p_i(\left\lceil \frac{A}{l_i - 1} \right\rceil + l_i - 1 - \left\lceil 2\sqrt{A} \right\rceil) + (p^s + 1)(2A - \left\lceil 2\sqrt{A} \right\rceil)\right]$$

And therefore, the total relative distance δ satisfies (since $\left\lceil 2\sqrt{A} \right\rceil \geq 3$ for all $A > 1$)

$$\delta < \frac{wM + \frac{Md}{A} + \sum_{i=1}^{2} p_i(\left\lceil \frac{A}{l_i-1} \right\rceil + l_i - 1 - 3) + 2A(p^s + 1)}{M\frac{wA+d}{A} \left\lceil 2\sqrt{A} \right\rceil}$$

and from this we get

$$\delta < \frac{M(wA + d) + A\sum_{i=1}^{2} p_i(\left\lceil \frac{A}{l_i-1} \right\rceil - 1 + l_i - 3) + 2A^2(p^s + 1)}{M(wA + d) \left\lceil 2\sqrt{A} \right\rceil}$$

Taking into account that

$$Ap_i(l_i - 3) \leq (l_i - 1)m_i(l_i - 3) \leq (k + 1)(k - 1)m_i \leq Am_i$$

and that

$$Ap_i\left(\left\lceil \frac{A}{l_i - 1} \right\rceil - 1\right) \leq A\frac{p_i A}{l_i - 1} \leq A\frac{(l_i-1)m_i}{l_i - 1} \leq m_i A$$

we conclude that

$$\delta < \frac{M(wA+d) + A\sum_{i=1}^{2}(m_i + m_i) + 2A^2(M/A+2)}{M(wA+d)\left\lceil 2\sqrt{A} \right\rceil}$$

and since $A \leq wA + d$ and $A \leq M$, we get

$$\delta < \frac{9}{\left\lceil 2\sqrt{A} \right\rceil}$$

∎

4 General Domains: the PERIX algorithm

Based on the previous observations , we have developed PERIX, an algorithm that accepts as input a (genetic) string of P optimal shapes and attempts to use them to tile the grid with minimum shape modification. The procedure that follows is similar to stripe-filling but more general.

To efficiently achieve this goal the PERIX algorithm maintains at each iteration a list of maximum free rectangles (this is the same data structure employed successfully in the related minimum-diversity problem [Yac93]). The elements of this list are maximal rectangles in the grid having the property that no part of them intersects an already placed shape. This list is sorted with primary key the y-coordinate of the upper left corner in ascending order, and secondary key the x-coordinate of the upper left corner in ascending order. At each iteration, the algorithm attempts to place the block part of the optimal shape it's working with in the upper-leftmost portion of the first free rectangle that fits in. If it can be placed within such a rectangle, it checks to see whether the left neighboring cells relative to the block are all occupied. If there exist cells that are not occupied then we put as much of the fringe as possible there. Furthermore, if even after the placement of the fringe to the immediate left of the block there still remain neighboring cells on that side that are not assigned, we alter the shape by removing cells from the rightmost column of its block and adding them to the remaining unoccupied left neigbours of the block. In this case the resulting shape is still optimal as its perimeter has not increased at all.

On the other hand, if after filling the unassigned left neighbor cells of the current block, there still remains some part of the fringe that has not been placed yet, the algorithm attempts to place this remainder in the immediate right neighboring column of the block. In this case, the semi-perimeter of the current shape becomes suboptimal by 1. When a stripe-decomposition of the domain is possible, this represents the worst-case result for any shape. Otherwise, the placement of the remainder of the fringe is postponed until all other shapes have been placed in the grid.

In the case when attempts to insert the block part of a shape fail because there exists no rectangle in the list of free rectangles big enough so as to accommodate the current block, the block is split. The algorithm finds a free rectangle that can accommodate as much of the current block as possible, and places this piece. Then, it places the rest of the shape wherever possible while trying to limit the increase of the perimeter that such splitting incurs.

Finally, after all the block-parts of the input shapes have been placed, the algorithm places any remaining fringes from shapes whose fringe has not yet been placed. Then, a swap phase follows where for a specified number of times, two cells are picked from the grid and a test is performed to see whether swapping them would actually reduce the perimeter. If swapping the two cells does not increase the total perimeter, the swap is actually performed. In this phase, tabu-like methods (see [GL93]) are employed: when swapping of two cells reduces total perimeter, neither of the two cells is considered for swapping again for a certain number of iterations (the hope is that the two cells were "properly" assigned). In figure 7 an optimal assignment for the MPE$(17, 17, 17)$ problem is shown, with total perimeter $17\Pi^*(17)$, produced by PERIX. Note that this solution is different from one achieved by simply using a stripe-filling process.

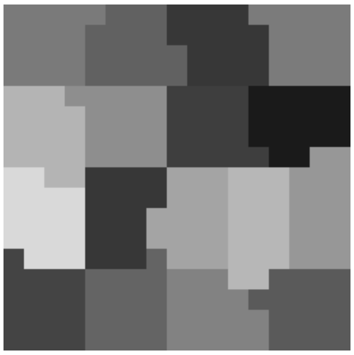

Figure 7: An optimal solution for the MPE(17,17,17)

5 A Genetic Algorithm for the General Case

So far we have discussed how our heuristic, given a selection of optimal shapes, seeks to minimize modification of those shapes. If for a certain problem MPE(M, N, P) the library of optimal shapes $\mathcal{L}(\frac{MN}{P})$ contains s shapes, then there are s^P different inputs for PERIX. To efficiently explore this huge space of solutions, we have employed the Genetic Algorithm (GA) paradigm.

Our Genetic Algorithm breeds a number of individuals that represent points in the space of possible inputs to PERIX. Each individual is therefore a string of P integers, each integer being an index to an optimal shape for the given problem. PERIX acts upon each individual of the current generation, and computes the perimeter that the individual produces. Then this perimeter is scaled to produce a fitness value. Depending upon this value, each individual may or may not mate with another individual to produce offspring. This is essentially the principle of natural selection (advocated originally by C. Darwin, and applied much more recently in many other contexts in [Hol92]).

A specialized directed *cross-over* operator is used; each gene in the individual has a tag associated with it, indicating whether the shape was placed in the beginning of a new row in the grid or not. When two individuals mate, they exchange their genetic material at points that are tagged as common beginnings of new rows for both of them. If such points in the genetic string do not exist, the common cross-over operator takes effect. The reason behind directed cross-over is the stripe form of the approximate solutions that we have discussed. We have experimented with one-point, two-point, and uniform cross-over, and have settled for one-point cross-over as this operator usually produces the best solutions faster in the evolution process.

Furthermore, the standard GA operator of mutation -changing some genes of the genetic string with some small probability- was implemented, as was the inversion operator [Hol92], an operator that inverts part of the genetic string. This last operator helped the diversity of the fitness values of the solutions as generations evolved. The actual mating strategy we used is roulette-wheel-based, i. e. the probability that an individual is chosen for mating is proportional to its fitness value.

Finally, we use a keep-incumbent and no-worse survival policy, where each offspring is compared with its parents (in terms of fitness function values); if the best perimeter value of all the children in the current generation does not improve on the incumbent value, then the children of parents with the incumbent value do not survive and are replaced by their parents in the current generation. Also, if an offspring is worse than the worst of all the individuals in the previous generation, it is replaced by one of its parents. More details about these policies can be found in [Yac93].

Overall our GA performs remarkably well, solving extremely large instances of MPEs. In the next section we present our results as well as some comparisons with other codes based on exact approaches.

6 Computational Results

We now present the results of our code and we make comparisons with a QAP code that we used in order to solve our problems. We have tested our code extensively on a very wide set of problems ranging from small problems to extremely large values of M, N and P. The algorithm consistently produced very good approximate solutions (and very often provably optimal ones).

We implemented our algorithm in C on a Thinking Machines CM-5 multiprocessor [Thi91] with 2 partitions of 32 SPARC processors each. To co-ordinate the processors we used the CMMD v.3.1 message passing library provided by Thinking Machines Inc [Thi93] . The communication overhead of our Genetic Algorithm is very low as the program spends less than 3% of total time in communications between processors.

We have also tried to solve some of these problems using the GRASP code of Pardalos, Resende et al.[2] This GRASP code uses a Branch & Bound type algorithm incorporating a variance reduction based lower bound (see [PRRL94]) for solving the QAP. It has been implemented in FORTRAN 77 and we report on the results we got by running the code on one node of the CM-5. Other GRASP codes that do not use Branch & Bound methods have been developed for the partitioning of general graphs into two equal size sub-graphs (see [LFE94]).

The times shown in table 1 are all in seconds. The times shown for the PERIX-GA algorithm are averages of 5 runs on the CM-5. In all our experiments, we let the PERIX-GA run for 20 generations. The column Gens indicates the number of generations it took the GA to find the best solution; this number is influenced by the random number seed. Various choices of the seed sometimes force the GA to go through many generations before it finds the best solution. The results shown are produced by the best choice of the random number seed for each problem as found by empirical testing. An asterisk in table 1 indicates the fact that although the best solution found differs from the one predicted by the lower bound, it is nevertheless optimal.

To understand the performance of the GRASP code, it is important to realize that the QAP formulation of the MPE(M, N, P) is of QAP dimension MN and in terms of binary variables, one needs MNP $0-1$ variables to formulate this problem as a facility location problem. Table 2 shows the size of each problem using a QAP, linear MIP, or a GA formulation. In the GA formulation, the size of the problem is measured as the number of components that PERIX-GA must tile together.

Exact codes for QAPs have solved problems of dimension up to 30, but in general QAPs with dimension higher than 20 are considered large, difficult problems [PRW93]. It should come as no surprise therefore that such approaches which have no knowledge of the geometric nature of the problem (and the form of the optimal solutions) have difficulties with the larger problems in our test set.

7 Conclusions and Future Directions

We have presented PERIX-GA, an algorithm that solves the Minimum Perimeter Equi-partition problem MPE(M, N, P) on rectangular domains. This problem is a special case of the Graph

[2] We are grateful to P. Pardalos and M. Resende for providing us with the source code of their algorithm developed at Bell Labs.

PROBLEM			Lower Bound	GRASP		PERIX-GA		
M	N	P		Err bnd(%)	Time	Err bnd(%)	Gens	Time
5	5	5	50	2.00*	18.2	2.0*	1	11.6
5	8	8	80	0.00	90.2	0.0	1	15.0
7	7	7	84	0.00	182.9	0.0	2	18.5
8	8	8	96	8.33	482.5	0.0	1	18.2
13	13	13	208	25.96	16357.6	0.0	11	385.0
17	17	17	306	-	-	0.0	9	578.5
32	31	8	368	-	-	2.17	2	358.4
32	31	32	768	-	-	0.52	4	228.1
101	101	101	4242	-	-	0.04	2	37.8
128	101	128	5376	-	-	0.14	4	45.3
122	122	122	5612	-	-	0.00	2	133.3
200	200	200	11600	-	-	0.06	12	158.9
512	512	512	47104	-	-	0.24	9	1339.1
1000	1000	1000	128000	-	-	0.32	7	8330.0

Table 1: Computational Results

PROBLEM			QAP		MIP		GA
M	N	P	VARS	CONSTR	VARS	CONSTR	VARS
5	5	5	125	30	165	830	5
5	8	8	320	48	387	3800	8
7	7	7	343	56	427	3584	7
8	8	8	512	72	624	6344	8
13	13	13	2197	182	2509	48854	13
17	17	17	4913	306	5457	148274	17
32	31	8	7936	1000	9857	108576	8
32	31	32	31744	1024	33665	1906656	32
101	101	101	1030301	10302	1050501	204030302	101
128	101	128	1654784	13056	1680411	416605568	128
122	122	122	1815848	15006	1845372	435848294	122
200	200	200	8000000	40200	8079600	3.168E+09	200
512	512	512	1.342E+08	262656	1.347E+08	1.369E+11	512
1000	1000	1000	1.0E+09	1001000	1.001E+09	1.996E+12	1000

Table 2: Problem Sizes under Various Formulations

Partitioning Problem which is NP-complete. We develop lower bounds using a theory of optimal tiles which states that for every instance of the problem there is a set of optimal shapes associated with it, having the property that if any P of them can be tiled together so as to completely cover the grid with no overlap, then the resulting partition is optimal. This allows us to include knowledge of the geometric aspect of the problem into our algorithm. We have proved that for a large class of problems there exist approximate solutions that yield objective values that differ from the theoretical lower bound only slightly as the problem size gets larger, and that this error in fact tends to zero. We have provided a technique for finding such solutions in polynomial time.

To efficiently explore the space of permutations of the input shapes, we have incorporated a Genetic Algorithm which breeds generations of solutions in the hope of finding an optimal selection. Genetic Algorithms are almost ideal candidates for parallel implementations and so we implemented our algorithm on a CM-5. In all our test problems the GA successfully found solutions having a distance from the lower bound no more than 2.1%.

Future work involves implementing the algorithm on a local Cluster of Workstations (COW) and extending the algorithm to partition arbitrary non-rectangular domains. We also plan to experiment with the possibility of replacing the swap phase of the PERIX algorithm by a more sophisticated assignment phase. Finally, we plan to investigate the possibility of extending our theoretical results on error bounds to larger classes of Minimum Perimeter Problems by modifying our construction technique or by examining new ones.

References

[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.

[GL93] F. Glover and M. Laguna. Tabu search. In C. R. Reeves, editor, *Modern Heuristic Techniques for Combinatorial Problems*, pages 70–150. Blackwell Scientific Publications, 1993.

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

[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.

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

[NW85] G. Nemhauser and L. Wolsey. *Integer and Combinatorial Optimization.* John Wiley & Sons, 1985.

[PRRL94] P. M. Pardalos, K. G. Ramakrishnan, M. G. C. Resende, and Y. Li. Implementation of a reduction based lower bound in a branch and bound algorithm for the quadratic assignment problem, September 1994. Submitted to the SIAM J. on Opt.

[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.

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

[Thi91] Thinking Machines Corporation. *The Connection Machine CM-5 Technical Summary*, October 1991.

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

[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. P. Pardalos, editor, *Advances in Optimization and Parallel Computing*, pages 293–309. North - Holland, 1992.