Fast Equi-Partitioning of Rectangular Domains using Stripe Decomposition*

Wayne Martin †

February 5, 1996

Abstract

This paper presents a fast algorithm that provides optimal or near optimal solutions to the minimum perimeter problem on a rectangular grid. The minimum perimeter problem is to partition a grid of size $M \times N$ into P equal area regions while minimizing the total perimeter of the regions. The approach taken here is to divide the grid into stripes that can be filled completely with an integer number of regions. This striping method gives rise to a knapsack integer program that can be efficiently solved by existing codes. The solution of the knapsack problem is then used to generate the grid region assignments. An implementation of the algorithm partitioned a 1000×1000 grid into 1000 regions to a provably optimal solution in less than one second. With sufficient memory to hold the $M \times N$ grid array, extremely large minimum perimeter problems can be solved easily.

Introduction

The focus of the algorithm presented here is the Minimum Perimeter Equi-partition problem, MPE(M, N, P). In this problem one is to partition an $M \times N$ rectangular grid into P equal area regions while minimizing the total perimeter of the partition. The one restriction of this algorithm is that all regions must have the same area. The area of each region is defined by $A = MN/P$ so the restriction is equivalent to P evenly dividing MN.

The minimum perimeter problem has several applications in parallel computer systems. In solving partial differential equations numerically, a grid is partitioned among the available processors. Using a five point numerical method, each grid element must communicate with its North, East, South, and West neighbors [DT91]. In assigning processors to the regions of the grid, one wants to minimize the communication between the processors while equalizing the number of grid elements assigned to each processor. This assignment process is analogous to the minimum perimeter problem. Another area of application is in image processing and edge detection in computer vision systems implemented on parallel hardware [Sch89]. Here again the rectangular image needs to partitioned among the processors to minimize inter-processor communication.

In order to calculate a lower bound for the minimum perimeter problem, Yackel and Meyer [YM92] have shown that the minimum perimeter of a single region with area A is determined by $\Pi^*(A)$.

(1) $$\Pi^*(A) = 2\left\lceil 2\sqrt{A}\right\rceil$$

If the entire grid could be tiled with shapes of the optimal perimeter without overlapping then an optimal solution would be found. Because one cannot do any better than this optimal tiling, a lower bound for the objective function of MPE(M, N, P) is given by $\underline{z}$.

(2) $$\underline{z} = P\,\Pi^*(A)$$

The minimum perimeter problem is a special case of the graph partitioning problem which is NP-complete. MPE(M, N, P) can be formulated as a quadratic assignment problem with MNP binary variables and $MN+P$ constraints. Details of this formulation are given in Christou and Meyer

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

† Computer Sciences Department, University of Wisconsin, Madison, Wisconsin 53706.

[CM95b]. Unfortunately, the QAP approach quickly becomes unsolvable as the grid size becomes moderately large.

The algorithm developed here takes an approach that considers the geometry of the problem. The method breaks the total area into a series of completely filled stripes. For example, figure 1 shows optimal striped solutions to MPE(7,7,7) and MPE(32,31,32). The MPE(7,7,7) solution consists of three stripes: two of height 2 and one of height 3. The MPE(32,31,32) solution has stripes of height 5 and 6. The motivation for the striping approach is twofold. First, in observing the optimal solutions produced by Christou and Meyer's PERIX-GA method ([CM95a] and [CM95b]), most of the optimal solutions exhibit a striped form. Second, the proofs of lower bound convergence make use of a stripe filling argument [CM95a]. Thus a stripe filling algorithm should be an effective way to solve the MPE problem.

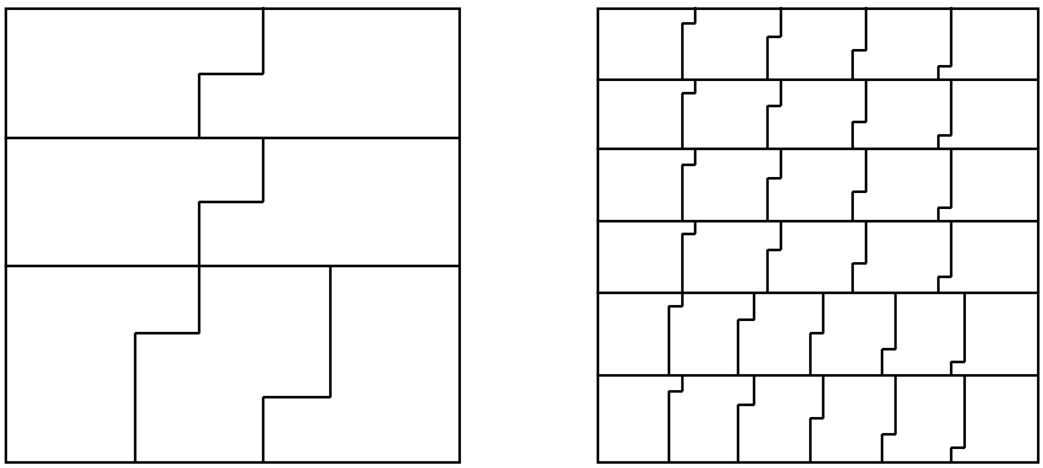


Figure 1 - Optimal Solutions of MPE(7,7,7) and MPE(32,31,32)

The algorithm consists of three phases. First, the possible completely filled stripe heights and corresponding perimeters are determined. The second phase is to solve a knapsack problem. The final phase takes the results of the knapsack problem and generates the region assignment grid. The following three sections describe in detail each of these phases.

Phase I - Perimeter of the Regions within a Stripe of Height h_i

The first part of the process is to determine the heights of the stripes that can be filled with a whole number of regions. Such heights will be termed "valid." Given h_i as a possible stripe height, $1 \le h_i \le \min\{A, M\}$, the area of the entire stripe, a_i, is calculated.

(3) $$a_i = Nh_i$$

Next, the number of regions within the stripe, p_i, is determined.

(4) $$p_i = \frac{a_i}{A}$$

If the value of p_i is an integer then the stripe can be filled completely and h_i is declared valid. Otherwise, this stripe height is no longer considered. Equation (4) can be rewritten using equation (3) and the definition of A to get equation (5). This implies that if P/M (or equivalently N/A) is an integer, all stripe heights will be valid. The condition that N/A is an integer means that the number of columns is a multiple of the area. Considering this geometrically it becomes obvious that all stripe heights will fill completely when N/A is an integer. In any event, a height of $\min\{A, M\}$ will always be a valid (though generally undesirable) stripe height.

$$p_i = \frac{a_i}{A} = \frac{N}{A} h_i = \frac{P}{M} h_i \tag{5}$$

Figure 2 shows a completely filled stripe of height $h_i = 3$, area $A = 7$, and $N = 14$. Applying equations (3) and (4) gives $a_i = 42$ and $p_i = 6$.

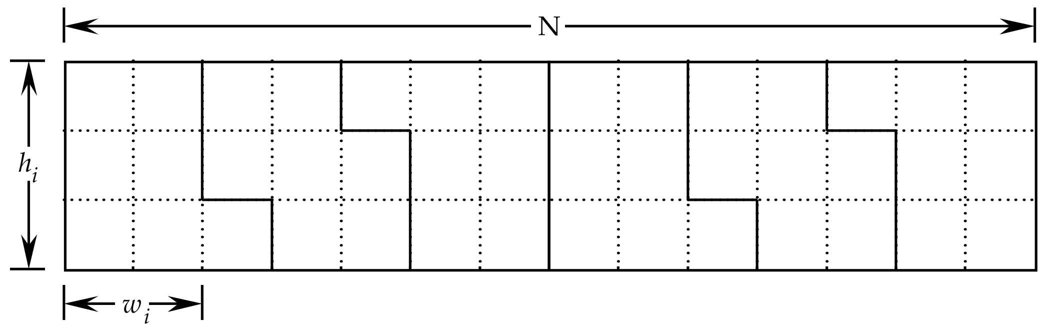


Figure 2

The width of the largest rectangle that will fit inside a region of area A and height h_i is determined by

$$w_i = \left\lfloor \frac{A}{h_i} \right\rfloor. \tag{6}$$

The cells of the region that are not in the largest rectangle are denoted as the fringe. The number of cells in the fringe is calculated as

$$f_i = A - h_i w_i\,. \tag{7}$$

For the example in figure 2, $w_i = 2$ and $f_i = 1$. Also seen in figure 2 is that the pattern of the cell shapes repeats itself every three regions. At the boundary between the repeating patterns the border is a vertical line and does not contain a step. To determine how often the pattern repeats, the following calculations are performed. If $f_i = 0$ then each region is rectangular and repeats every one region ($r_i = 1$). For $f_i > 0$ the repeat count, r_i, is determined as

$$r_i = \min\left\{ \frac{th_i}{f_i} \;\middle|\; \frac{th_i}{f_i} \text{ integer, } t = 1,2,\ldots \right\}. \tag{8}$$

The final step is to calculate the total perimeter of all the regions in the stripe. The perimeter of the outside of the stripe is simply $2(N + h_i)$. The number of boundaries between regions within the stripe is $(p_i - 1)$ of which $(p_i/r_i - 1)$ are vertical lines of length h_i and the remaining borders have a step in them giving a length of $(h_i + 1)$. Putting this all together gives the formula for the total perimeter of the regions within the stripe, c_i.

$$c_i = 2\left[N + h_i + \left(\frac{p_i}{r_i} - 1\right)h_i + \left(p_i - 1 - \left(\frac{p_i}{r_i} - 1\right)\right)(h_i + 1)\right] \tag{9}$$
$$= 2\left[N + p_i(h_i + 1) - \left(\frac{p_i}{r_i}\right)\right]$$

Phase II - Construction and Solution of the Knapsack Problem

At the end of phase I, the algorithm has generated n stripe heights and their corresponding perimeters (h_i and c_i, i=1,...,n). Phase II constructs a knapsack integer program to determine the combination of stripe heights that will completely fill the entire grid and produce the minimum total perimeter. The value of x_i represents how many stripes of height h_i are in the optimal striped solution. The knapsack problem is formulated as follows.

$$\begin{aligned} &\underset{x}{\text{minimize}} \sum_{i=1}^{n} c_i x_i \\ &\text{subject to} \sum_{i=1}^{n} h_i x_i = M \\ &x_i \geq 0, \text{integer}, i = 1, \ldots, n \end{aligned} \tag{10}$$

Using the integrality of x and the fact that $h_i x_i \leq M$ must hold for each i, it is possible to define a bound (b_i) on the x_i variables. This bound helps in finding the solution of the knapsack problem.

$$x_i \leq b_i = \left\lfloor \frac{M}{h_i} \right\rfloor \tag{11}$$

Theorem 1 shows that the integer program in (10) always has a feasible solution and, since x is bounded, (10) also has an optimal solution. This implies that when MN/P is an integer, the MPE(M, N, P) problem has a feasible solution that is in striped form.

Theorem 1 *If* $\frac{MN}{P}$ *is an integer then the integer program in (10) has a feasible solution.*

Proof: Case 1) $M \leq A$. This case is trivial since a feasible solution to (10) is one stripe of height M which contains all P regions.

Case 2) $M > A$. For this case, one stripe of height M is invalid since the equations for calculating c_i are only for $h_i \leq A$. By equation (5), a stripe height of A is valid since $p_i = \frac{N}{A} A = N$ is an integer. A stripe of height A consists of N rectangular regions of width 1 and height A. To construct the feasible solution, the majority of the grid will be filled with k stripes of height A where $k = \left\lfloor \frac{M}{A} \right\rfloor$. This will leave $M' = M - kA$ rows remaining to be filled with $P' = P - kN$ regions. If $M' = P' = 0$ then a feasible solution has been found: k stripes of height A. Otherwise it must be shown that M' is a valid stripe height. Equation (5) is used again to show M' is valid since,

$p_i = \frac{P}{M} M' = \frac{PM - kPA}{M} = \frac{PM - kMN}{M} = P - kN$ is an integer. Thus a feasible solution consists of k stripes of height A and one stripe of height M'. ■

The optimal solution to problem (10) is not necessarily unique. An example of non-uniqueness can be found in the problem MPE(12,12,12). For this problem, solutions of three stripes of height four and four stripes of height three are both optimal.

Phase III - Grid Assignment

The final phase of the algorithm is to take the solution of the knapsack problem and generate the assignment grid. For each x_i of the solution vector not zero, x_i stripes of height h_i are added to the assignment grid. The striping procedure follows that given in [CM95a]. Below is the pseudo-code for the grid assignment phase.

```
inputs N - Number of columns in grid,
       A - Area of each region,
       h - Array of stripe heights,
       x - Solution of the knapsack problem,
       n - Number of elements in h and x.
output grid - Two dimensional array of the region assignments.
begin assign_grid
    toprow := 1
    proc := 1
    count := 0
    for i := 1 to n
        for j := 1 to x[i]
            bottomrow := toprow + h[i] - 1
            for col := 1 to N
                for row := toprow to bottomrow
                    grid[row,col] := proc
                    count := count + 1
                    if (count = A) then
                        proc := proc + 1
                        count := 0
                    end if
                end for
            end for
            toprow := bottomrow + 1
        end for
    end for
end assign_grid
```

Program Implementation

The implementation of this algorithm was coded in FORTRAN and was written as a callable subroutine. The inputs to the subroutine are M, N, and P and the declared dimensions of the grid array. The output is the minimum perimeter found and a two dimensional array of the grid assignments.

The MSP (Minimum Striped Perimeter) subroutine makes use of three other subroutines which correspond to the three phases described earlier. The first subroutine, GEN_STRIPES, generates the valid stripe heights and corresponding perimeters. Initially, stripe heights between h_{min} and h_{max}

are considered (see (12)). If no valid heights are found between h_{min} and h_{max}, the range is expanded to $\left[1, \min\{A, M\}\right]$.

$$h_{min} = \tfrac{1}{2}\sqrt{A} \quad \text{and} \quad h_{max} = 2\sqrt{A} \tag{12}$$

The second subroutine is KNAPSACK. It takes the stripe heights and perimeters generated in GEN_STRIPES and solves the knapsack problem using Martello and Toth's MTB2 routine [MT90]. The MTB2 routine requires that the problem be formulated as a bounded maximization problem as shown in (13).

$$\begin{aligned} &\underset{y}{\text{maximize}} \sum_{i=1}^{n} c_i y_i \\ &\text{subject to} \sum_{i=1}^{n} h_i y_i \le K \\ &\qquad 0 \le y_i \le b_i,\ y_i \ \text{integer}, i = 1,\dots,n \end{aligned} \tag{13}$$

The MTB2 subroutine also requires that c_i, h_i, and b_i all be positive integers. The following steps are taken to put (10) into the required form. First a variable substitution is made.

$$y_i = b_i - x_i \tag{14}$$

Substituting (14) into (10) and (11) and writing as a maximization problem yields

$$\begin{aligned} &\underset{y}{\text{maximize}} \sum_{i=1}^{n} c_i y_i - \sum_{i=1}^{n} c_i b_i \\ &\text{subject to} \sum_{i=1}^{n} h_i y_i = \sum_{i=1}^{n} h_i b_i - M \\ &\qquad 0 \le y_i \le b_i,\ y_i \ \text{integer}, i = 1,\dots,n \end{aligned} \tag{15}$$

Dropping the constant term from the objective function and letting $K = \sum_{i=1}^{n} h_i b_i - M$, problem (15) is almost in the form required by MTB2. The only difference is the strict equality constraint in (15) versus the inequality in (13). Theorem 2 shows that for the data from the MPE problem, the optimal solution of (13) will always satisfy the inequality constraint as an equality.

Theorem 2 *An optimal solution to the integer program (13) must satisfy the inequality constraint as a strict equality when the c_i, h_i, and b_i are generated by the MSP algorithm.*

Proof by contradiction: Assume that y^* is optimal for (13) and that $\sum_{i=1}^{n} h_i y_i^* < K = \sum_{i=1}^{n} h_i b_i - M$.

Define $D = \left(\sum_{i=1}^{n} h_i b_i - M\right) - \sum_{i=1}^{n} h_i y_i^*$. Obviously $D \ge 1$. If $D \le A$ then it can be shown that D is a valid stripe height. By equation (5)

$$p_D = \frac{P}{M} D = \frac{P}{M}\left(\sum_{i=1}^{n} h_i b_i - \sum_{i=1}^{n} h_i y_i^*\right) - P = \sum_{i=1}^{n} \frac{P}{M} h_i \left(b_i - y_i^*\right) - P = \sum_{i=1}^{n} p_i \left(b_i - y_i^*\right) - P$$ which is an integer. Let $\bar{i}$ be the index such that $h_{\bar{i}} = D$.

If $D > A$ then in the proof of theorem 1 it was shown that a stripe height of A is valid so let $\bar{i}$ be the index such that $h_{\bar{i}} = A$.

Define $\bar{y}_i = y_i^*$ for $i \neq \bar{i}$ and $\bar{y}_{\bar{i}} = y_{\bar{i}}^* + 1$. This $\bar{y}$ is feasible because $\bar{i}$ was chosen based on the value of D. Since $c_i \geq 1$ for all i, it follows that $\sum_{i=1}^{n} c_i \bar{y}_i > \sum_{i=1}^{n} c_i y_i^*$, but this contradicts the assumption that y^* was optimal. Therefore the inequality constraint in (13) will always be satisfied as an equality for any optimal solution y^*. ■

The knapsack integer program (13) is passed to MTB2 to find the optimal solution. Once found, substitution (14) is reversed and the optimal value *z** of (10) is calculated from the optimal value *z*** returned by MTB2. The value *z** is the total perimeter for the solution of the MPE(*M, N, P*) problem.

(16) $$z^* = \sum_{i=1}^{n} c_i b_i - z^{**}$$

The third and final subroutine is GEN_GRID. This routine takes the solution of the KNAPSACK routine and fills in the assignment grid. A special option was added to the subroutine for extremely large problems. If the dimensions of the assignment grid array are passed in as zeroes then this routine is not called. This was done so that perimeters could be calculated for problems for which the grid assignment array would not fit into memory.

The main MSP subroutine also has extra code to check the transverse of the problem. If the original problem MPE(*M, N, P*) is not solved to optimality and $M \neq N$ then the routine also solves MPE(*N, M, P*). The better solution of the original and the transpose is passed to the GEN_GRID subroutine. The GEN_GRID subroutine makes sure that the output grid is in the correct orientation regardless of whether the original or the transpose was used.

Computational Results

This section presents the computational results of the presented algorithm. The program was tested on a Sun SPARCstation-20 workstation. First, table 1 compares the striping algorithm developed here (MSP) with the genetic algorithm PERIX-GA [CM95b] running on a cluster of 33 SPARCserver-20 computers. The times in the table all in seconds and the "Error" columns are the percent error from the lower bound. The first observation is that the running times for MSP are extremely fast. Second, the quality of the solutions from MSP are as good as or better than PERIX-GA in all cases except MPE(17,17,17). In this case PERIX-GA found an optimal solution where MSP did not. This is because the optimal solution is not in a striped form so MSP could not find it.

Problem			Lower	PERIX-GA		MSP	
M	*N*	*P*	Bound	Err (%)	Time	Err (%)	Time
7	7	7	84	0	196.1	0	0.01
13	13	13	208	0	227.8	0	0.01
17	17	17	306	0	268.6	0.65	0.01
32	31	256	2048	0	230.2	0	0.01
101	101	101	4242	0.05	219.1	0.05	0.04
200	200	200	11600	0	261.0	0	0.07
256	256	256	16384	0	105.1	0	0.09
512	512	512	47104	1.63	279.0	0.14	0.25
1000	1000	1000	128000	0.45	1660.5	0	0.67
2001	2001	2001	360180	-	-	0.08	2.18

Table 1 - Comparison of MSP and PERIX-GA

The next table compares MSP with PERIX-GA and two other popular graph-partitioning methods, the spectral bisection method and the geometric mesh partitioning method. The Chaco package version 2.0 was used for the spectral bisection method [HL95]. The geometric method was implemented in MATLAB as described in Gilbert, Miller, and Teng [GMT95]. Both the spectral bisection and the geometric mesh partitioning methods have the restriction that the number of regions be a power of two. The times and error values for these methods were taken from [CM95b].

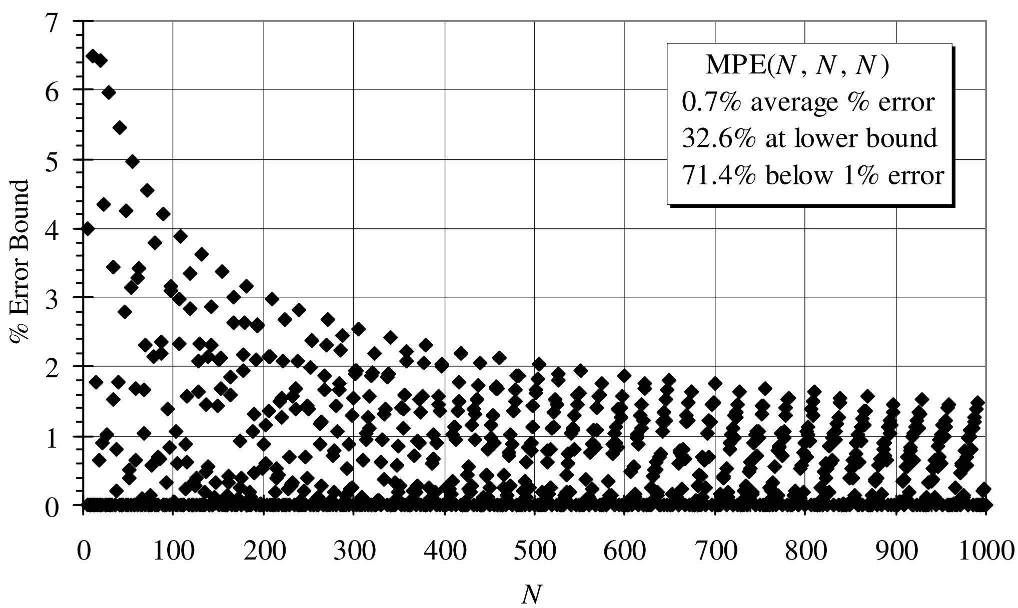
Problem			Lower	SPECTRAL		GEOMETRIC		PERIX-GA		MSP	
M	*N*	*P*	Bound	Err (%)	Time	Err (%)	Time	Err (%)	Time	Err (%)	Time
32	31	8	368	6.52	1.8	5.43	43.6	2.17	84.0	1.09	0.01
32	31	256	2048	6.73	4.3	-2.73*	152.3	0	80.4	0	0.01
32	30	64	1024	6.25	3.0	6.25	90.4	0	50.9	0	0.01
100	100	8	1136	9.33	9.0	7.36	111.0	2.64	81.9	5.63	0.04
128	128	128	5888	14.13	85.5	7.13	539.9	1.65	67.6	1.63	0.04
256	256	256	16384	13.25	227.8	4.15	3304.2	0	105.1	0	0.09
512	512	512	47104	-	-	-	-	1.63	279.0	0.14	0.25

Table 2 - Comparison of MSP to other methods

This table also shows that MSP is very fast compared to the other methods. It also produced solutions that were as good or better than the other methods with the exception of MPE(100,100,8) for which PERIX-GA found a better solution which did not have a striped form. The MPE(32,31,256) problem presented some difficulties because the area, A=32·31/256, is not an integer. A modification to the program was made to break the problem into two parts. Specifically MPE(32,28,224) with area 4 and MPE(32,3,32) with area 3. Each of these subproblems were solved separately then the assignment grids where appended together. The asterisk on the geometric partitioning result indicates that the solution found was unbalanced (i.e. some regions had area other than 3 or 4).

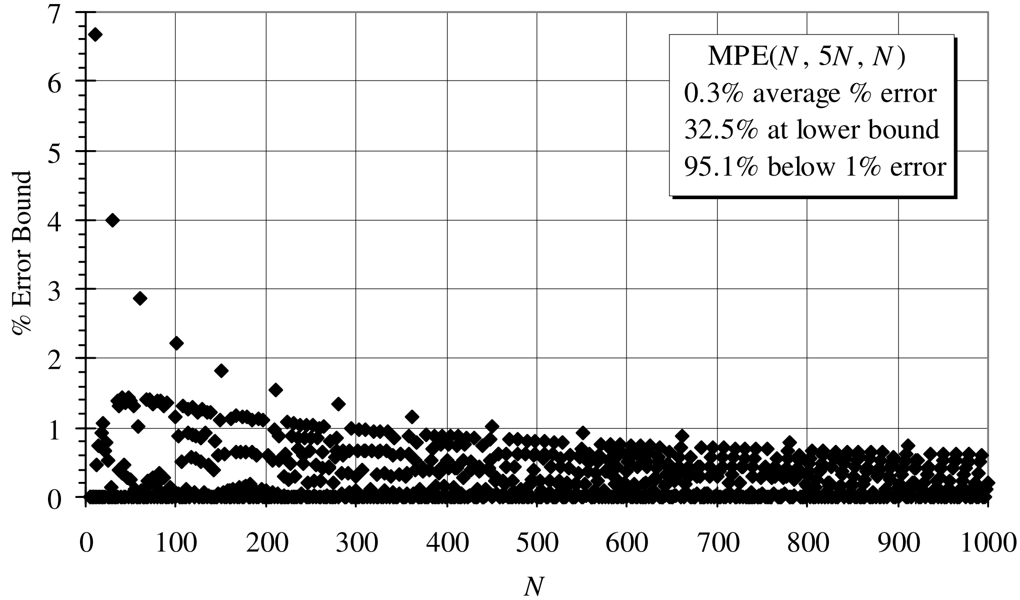
The MSP program is also capable of handling very large problems. One problem solved was MPE(10000,10000,1000). The grid size for this problem is 10^8 elements and the area assigned to each processor is 10^5. The large grid array size exceeded the computer memory available, thus phase III of the algorithm (explicit generation of assignments) could not be completed. Phases I and II were run in 0.22 seconds to calculate a perimeter which was within 0.042% of the lower bound. The solution consisted of 23 stripes of height 320 and 8 stripes of height 330. Another interesting large problem solved was MPE(20202,20202,20202). This problem was solved to within 0.006% of the lower bound in 3.59 seconds. The solution had a non-trivial striping configuration of: 3 stripes of height 138, 3 stripes of height 140, 4 stripes of height 143, and 127 stripes of height 148.

To see how the algorithm performs over a wider set of problems, four sequences of problems were run. The first sequence computed the percent from lower bound for MPE(N, N, N) for N from 5 to 1000. The graph of the bound error versus N is shown in graph 1. For this sequence the average error from lower bound was 0.7%. Of the 996 problems solved 32.6% were provably optimal and 71.4% had an error of less than 1%. As the theory in [CM95a] and [CM95b] predicts, the error bound seems to be approaching zero as N increases.



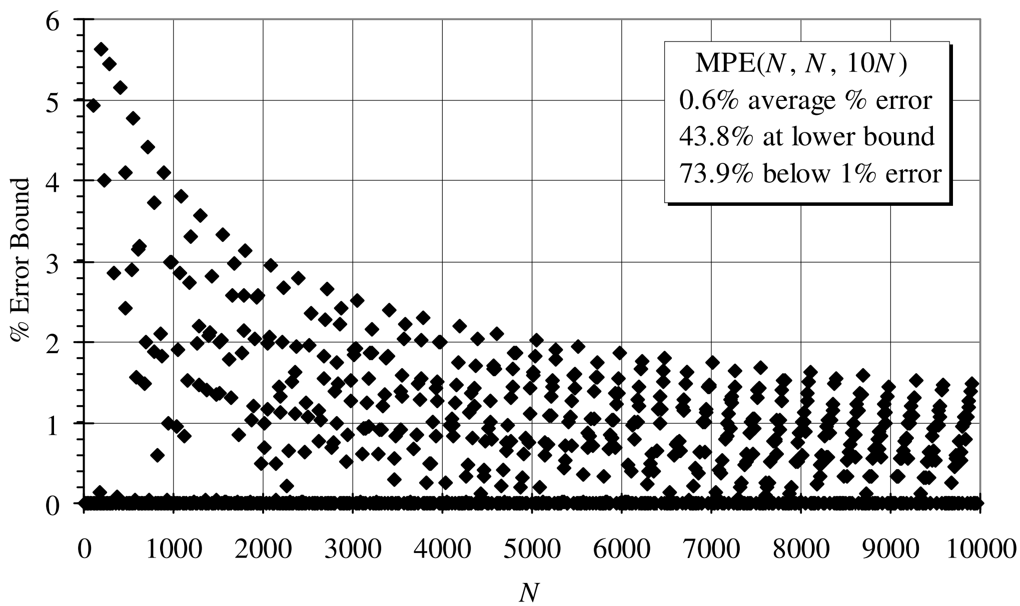
Graph 1 - Percent from lower bound versus N for MPE(N, N, N)

Graph 2 shows the sequence MPE(N,5N, N). This sequence tests how well the algorithm performs on narrow rectangular domains. Note that the same results would have been obtained for MPE(5N, N, N) because the program solves both the original problem and its transpose. The percentage of solutions that are at the optimal lower bound is about the same as that for the MPE(N, N, N) sequence. But the average percent error and the percent below 1% error are much better.



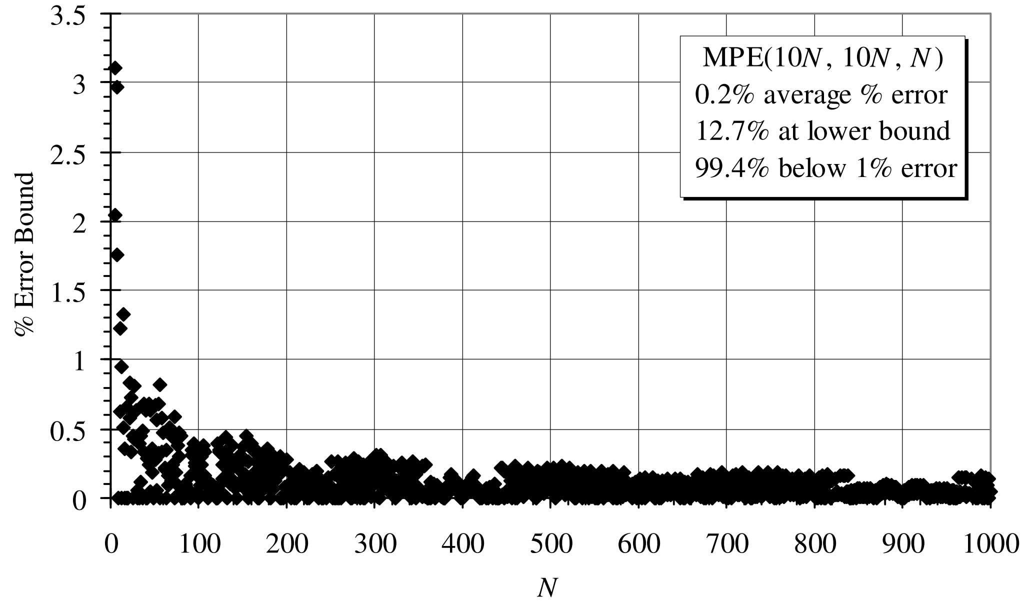
Graph 2 - Percent from lower bound versus *N* for MPE(*N*, 5*N*, *N*)

Graph 3 is for the sequence MPE(*N*, *N*,10*N*) for *N* from 50 to 10000 with an increment of 10. This sequence tests the cases when the number of regions is large compared to the grid dimensions. The percentage of solutions at the optimal lower bound has increased to almost 44%. But, as can be seen from the graph, the percent error values are larger than the previous two sequences.



Graph 3 - Percent from lower bound versus *N* for MPE(*N*, *N*, 10*N*)

Graph 4 is just the opposite of graph 3. This shows the sequence MPE(10*N*,10*N*, *N*) in which the number of regions is small compared to the grid size. Here only 12.7% of the solutions are at the optimal lower bound, but the average error is lower than all the other sequences.



Graph 4 - Percent from lower bound versus *N* for MPE(10*N*, 10*N*, *N*)

To determine the algorithm performance for image processing type applications, tables 3 and 4 were generated. These tables show the percent from lower bound that MSP produced for grid sizes and processor numbers that are all a power of two. Table 3 is for square grids and table 4 is for rectangular grids with proportions 2 to 1. For the square grids, processor numbers of 16, 64, and 256 were omitted because the solution is just the trivial solution of breaking the grid up into square regions. For the same reason, columns 8, 32, 128, and 512 were omitted from the rectangular grid results. Figure 3 shows a typical example of a power of two partitioning.

Grid Size		*P*			
M	*N*	8	32	128	512
32	32	2.17	0	0	0
64	64	1.09	2.17	0	0
128	128	1.65	0.54	1.63	0
256	256	1.37	0.82	0.14	1.63
512	512	1.52	0.41	0.41	0.14
1024	1024	1.59	0.48	0	0.41
2048	2048	1.62	0.52	0.07	0
4096	4096	1.64	0.53	0.10	0.02
8192	8192	1.65	0.54	0.12	0.03
16384	16384	1.64	0.55	0.13	0.08
32768	32768	1.65	1.09	0.41	0.20

Table 3 - Percent from lower bound for square power of 2 grid sizes

Grid Size		*P*			
M	*N*	16	64	256	1024
32	16	0	0	0	-
64	32	2.17	0	0	0
128	64	0.54	1.63	0	0
256	128	0.82	0.14	1.63	0
512	256	0.41	0.48	0.14	1.50
1024	512	0.48	0.10	0.41	0.03
2048	1024	0.52	0.19	0	0.38
4096	2048	0.53	0.23	0.03	0
8192	4096	0.54	0.25	0.05	0.03
16384	8192	0.55	0.27	0.06	0.05
32768	16384	0.54	0.27	0.10	0.03

Table 4 - Percent from lower bound for rectangular power of 2 grid sizes

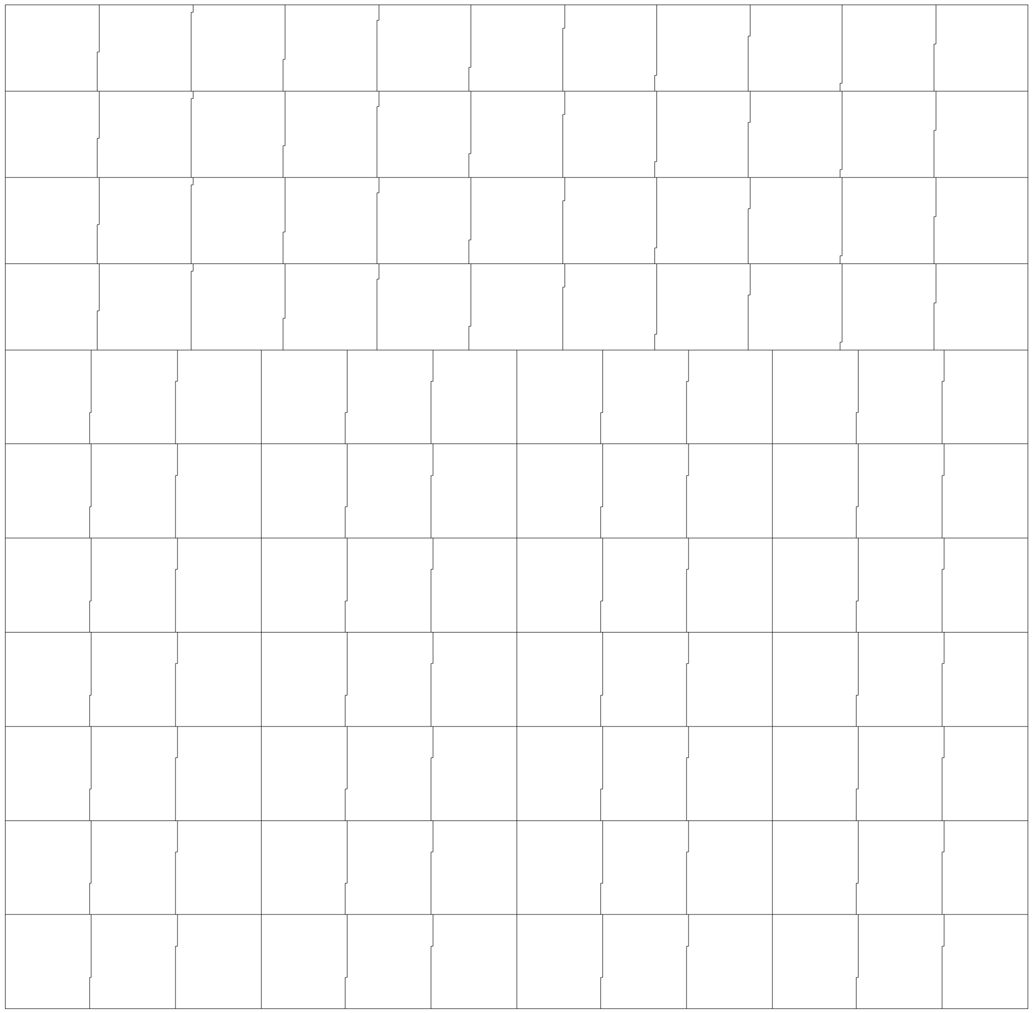


Figure 3 - MSP solution of MPE(512,512,128) within 0.41% of lower bound

Limitations

There are two main limitations to the MSP algorithm. The first is that it may not give results as good as PERIX-GA when the best solution is not in striped form (for example: MPE(17,17,17) and MPE(100,100,8)). The second limitation is that the algorithm cannot be directly extended to non-rectangular domains or to problems where the area of each region is non-uniform. In some cases though, non-uniform area problems can be done if the problem can be split into two subproblems. The two subproblems can be solved separately, then the results combined as in MPE(32,31,256) described earlier.

Conclusions

This paper has presented an algorithm based on optimal stripe decomposition that provides very good solutions to the Minimum Perimeter Equi-partition problem, MPE(M, N, P), on a rectangular grid. This algorithm has proven to be extremely fast compared to other graph partitioning methods and can handle very large problems which are intractable for other methods. The algorithm also has the property that as the problem size increases the deviation from the lower bound decreases.

Acknowledgments

I would like to thank I. Christou and R. Meyer for allowing me to use their test results for the spectral and geometric mesh partitioning methods and for the PERIX-GA method. I also thank R. Meyer for his guidance and direction on this project.

References

[CM95a] 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, WI, Feb. 1995. To appear in *Journal of Global Optimization*.

[CM95b] I. T. Christou and R. R. Meyer, Optimal and asymptotically optimal equi-partition of rectangular domains via stripe decomposition, Technical report MPTR 95-19., University of Wisconsin, Madison, WI, Nov. 1995.

[DT91] R. DeLeone and M. A. Tork-Roth, Massively parallel solution of quadratic programs via successive overrelaxation, Technical report 1041, University of Wisconsin, Madison, WI, 1991.

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

[HL95] B. Henderson and R Leland, *The Chaco User's Guide Version 2.0* (Sandia National Laboratories, 1995).

[MT90] S. Martello and P. Toth, *Knapsack Problems: Algorithms and Computer Implementations* (John Wiley & Sons, 1990).

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

[YM92] J. Yackel and R. R. Meyer, Minimum perimeter decomposition, Technical report 1078, University of Wisconsin, Madison, WI, 1992.