Minimum-Perimeter Domain Assignment [*]

Jonathan Yackel[†] Robert R. Meyer[‡] Ioannis Christou[‡]

Abstract

For certain classes of problems defined over two-dimensional domains with grid structure, optimization problems involving the assignment of grid cells to processors present a nonlinear network model for the problem of partitioning tasks among processors so as to minimize interprocessor communication. Minimizing interprocessor communication in this context is shown to be equivalent to tiling the domain so as to minimize total tile perimeter, where each tile corresponds to the collection of tasks assigned to some processor. A tight lower bound on the perimeter of a tile as a function of its area is developed. We then show how to generate minimum-perimeter tiles. By using assignments corresponding to near-rectangular minimum-perimeter tiles, closed form solutions are developed for certain classes of domains. We conclude with computational results with parallel high-level genetic algorithms that have produced good (and sometimes provably optimal) solutions for very large perimeter minimization problems.

1 Introduction

Many computations performed by systems of parallel processors involve a collection of tasks which are related by a rectangular grid structure (*i.e.*, as in figure 1, each task has at most four "neighbor" tasks). Examples include the problem of determining the characteristics of fluid flow [8], solving obstacle problems using parallel successive overrelaxation [3], and edge detection in computer vision [16]. We assume initially that all grid cells are squares of uniform size as in figure 1, and that there is a task associated with each cell that uses only its own data and values from neighboring cells that share an edge. For cells on the boundary of the given region, boundary conditions may be used in the computations. If the grid cells are assigned to the processors (that is, the computation for each cell is done by a particular processor), then sharing data with neighboring cells may involve communicating with other processors.

[*]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 CCR-8907671 and CCR-9306807.

[†]Computer Science Department, UW Oshkosh, 800 Algoma Blvd., Oshkosh, WI 54901.

[‡]Center for Parallel Optimization, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin 53706.

1	1	1					8	8	8
1	1	1					8	8	8
2	2	2					7	7	7
2	2	2	4	4	5	5	7	7	7
3	3	3	4	4	5	5	6	6	6
3	3	3	4	4	5	5	6	6	6

Figure 1: Assigning the grid cells of a domain to processors

The term "tile" will refer to a connected group of cells assigned to the same processor. (In the combinatorics literature [11, 12], such a configuration is called a polyomino.) We say a set of cells is connected if for every pair of cells c_i, c_j there is a *path* of cells in the set from c_i to c_j such that adjacent cells on the path share an edge. (We will show that, in order to achieve the lower bounds on perimeter derived in §3, the cells assigned to each processor must be connected.) To measure interprocessor communication, we measure the length of the tile borders because only across the tile borders may data pass between different processors. In figure 1 we have placed processor identification numbers in the cells to indicate the assignment of cells to processors. For the case depicted in the figure there are eight processors, each assigned six cells for load balancing. Each processor's tile has a perimeter of ten, so the total length of the tile borders is 80 (the results in section 3 show this is the minimum possible total border length for any load-balanced assignment). In general the processors' tiles can be of different shapes even if their loads are equal.

This paper discusses the nonlinear network flow problem of minimizing total tile perimeter subject to workload balancing constraints; a formal statement of this problem is presented in §2. In §3, we develop lower bounds on individual tile perimeter and total tile perimeter (these bounds are used to establish the optimality of particular solutions for certain cases). §4 develops optimal tiles of cells for individual processors, and §5 provides combinations of these tiles producing optimal assignments that attain the lower bound on total perimeter. In §6 we investigate a database application in which domain boundaries are treated differently because a different style of communication is assumed; this results in a modification of the objective function. A genetic algorithm that proved very effective in this database application is discussed in §7, and extensions of this approach to the minimum perimeter and other problems are also considered. Our conclusions and some future research directions are contained in §8.

2 Problem Statement

Suppose that we wish to allocate the cells of a domain among N processors. Let $\mathcal{A}$ denote the number of cells (area) of the domain. Given a processor p, let A_p denote the number of cells (or area) assigned to p. (A_p may also be thought of as the workload assigned to processor p since there is an equal amount of computation associated with each cell.) Load balancing is achieved by specifying an appropriate A_p for each processor. (In typical applications, the specified processor loads are equal or differ by at most 1. It is assumed that $\sum_p A_p = A$.) We use $\mathcal{P}(T)$ to denote the perimeter of a configuration T of cells. The notation $\mathcal{P}(T_p)$ is used to denote the perimeter of the tile(s) held by processor p. (The cells assigned to processor p are not necessarily connected, and therefore may comprise several tiles rather than a polyomino.) The *objective function* $\mathcal{C}$ that we wish to minimize measures interprocessor communication and is defined as follows: $\mathcal{C} := \sum_p \mathcal{P}(T_p)$. (As we will see below, this objective can also be represented as a sum of quadratic terms, each of which corresponds to the assignment of adjacent cells to different processors.) This model is motivated by the assumption that total interprocessor communication may be expressed as the sum of the communication associated with the domain boundary (if there is any such communication) and the communication associated with "interior" borders between tiles (the total length of which is determined by the manner in which cells are assigned to processors). With respect to communication corresponding to the domain boundary, there are at least two possible simplifying assumptions that mesh with the models to be detailed below. One could assume that the computation associated with the boundary cells (i.e., those cells with edges on the boundary of the domain) requires no communication across boundary edges. Alternatively one could assume some fixed amount of communication proportional to boundary edges for each boundary cell. In either case the total amount of communication corresponding to the domain boundary is a constant. Thus, the total communication is given by $k_1\mathcal{B} + k_2\sigma$, where $\mathcal{B}$ is the total length of the domain boundary, σ is the total length of the border *between* tiles (note that in σ each piece of the "interior" border is counted twice, once for each tile), and k_1 and k_2 are scale factors relating boundary and border lengths to communication. Since $\mathcal{B}$ is constant, minimizing this expression is equivalent to minimizing $k_2\mathcal{B}+k_2\sigma$, which in turn is equivalent to minimizing $\mathcal{B}+\sigma$, the *total tile perimeter*, $\sum_p \mathcal{P}(T_p)$. This equivalent geometric problem, formally stated below, is a nonlinear semi-assignment problem:

Given: *N processors, a domain comprised of grid cells, and a load A_p for each processor.*

Problem: *Find an assignment that solves*

$$\begin{array}{ll} \textit{min} & \mathcal{C} = \sum_p \mathcal{P}(T_p) \\ \textit{s.t.} & \textit{every cell is assigned to a single processor,} \\ & \textit{and processor } p \textit{ is assigned } A_p \textit{ cells (for every } p = 1, 2, \ldots, N). \end{array}$$

Algebraically, letting c denote the total number of cells, this problem is most easily formulated as a *quadratic assignment problem*:

$$\text{(1)} \qquad \min \sum_{i,i'=1}^{c} \sum_{p,p'=1}^{N} k_{ii'} x_i^p x_{i'}^{p'}$$

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

$$\text{where } k_{ii'} = \begin{cases} 1 & \text{if i, i' are adjacent cells} \\ 0 & \text{otherwise} \end{cases}$$

It is easily seen that the number of assignments satisfying the balancing constraint is

$$\binom{\mathcal{A}}{A_1}\binom{\mathcal{A}-A_1}{A_2}\binom{\mathcal{A}-A_1-A_2}{A_3}\cdots\binom{\mathcal{A}-A_1-A_2-\cdots-A_{N-1}}{A_N} = \frac{\mathcal{A}!}{\prod_{p=1}^{N}(A_p!)}.$$

Complete enumeration of these assignments is not feasible even for relatively small problems. For example, given a domain consisting of 25 cells, 5 processors, and a load of 5 for each processor, there are more than 623×10^{12} possible assignments.

The processor assignment problem is related to the graph partition problem which is NP-complete (see Garey and Johnson, page 209 [4]). Yackel showed that the problem is NP-hard if the domains are allowed to be arbitrary rectilinear grids [17]. The complexity of the problem restricted to rectangular domains is unknown.

3 Lower Bounds

In this section we will develop a lower bound on the measure $\mathcal{C}$ and discuss conditions under which this lower bound is attained. To do this we introduce the concept of "semi-perimeter" for a configuration C, denoted by $\mathcal{S}(C)$, defined as the width plus height of the smallest rectangle enclosing the compact form of the configuration C, where all of the space between slices has been removed (see for example, figure 2, in which the cells of a configuration are labeled with the index p).

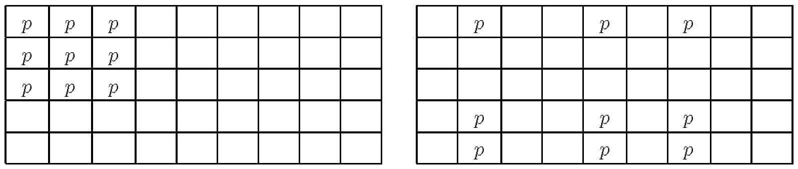
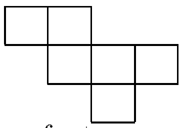


Figure 2: Compact and non-compact forms of a configuration

(As discussed below, semi-perimeter is also half of ordinary geometric perimeter for most configurations of interest, including minimum-perimeter tiles.)

For example, the semi-perimeter of the configuration  is $4+3=7$ and its perimeter is 14. The term "slice" is used to refer to a row or column of either the *domain* or of a *configuration*, depending on the context. $\mathcal{S}(C)$ is thus the number of slices intersecting C. A tight lower bound on $\mathcal{S}(C)$ as a function of the number of cells in C (equivalently, the area of C) will be developed, yielding a tight lower bound on $\mathcal{P}(C)$, hence a lower bound on $\mathcal{C}$.

We introduce the notion of "slice-convexity" of a configuration. (In the polyomino literature, such configurations are simply called "convex" even though they need not be convex in the usual mathematical programming sense. To emphasize their possible non-convexity, we will use the term "slice-convex".)

Definition 1 *A configuration is* slice-convex *if for any two cells c_1, c_2 of the configuration in the same slice, the smallest rectangle containing c_1 and c_2 lies entirely in the configuration.*

Lemma 2 *For any configuration C, $\mathcal{P}(C) \geq 2\mathcal{S}(C)$. Furthermore, $\mathcal{P}(C) = 2\mathcal{S}(C)$ if and only if C is a slice-convex configuration.*

Proof: There are at least two edges forming part of the configuration border in each slice of C. Therefore each slice of C contributes at least 2 to the perimeter of C, but exactly 1 to $\mathcal{S}(C)$. Since $\mathcal{S}(C)$ is the number of slices of C, $\mathcal{P}(C) \geq 2\mathcal{S}(C)$. For a slice-convex configuration, each slice of the configuration contains exactly 2 configuration borders in the dimension corresponding to the slice, so that for a slice-convex configuration $\mathcal{P}(C) = 2\mathcal{S}(C)$. For a configuration that is not slice-convex, there is a slice with more than 2 configuration borders, therefore $\mathcal{P}(C) > 2\mathcal{S}(C)$. ∎

When considering minimum-perimeter configurations in the following sections, lemma 2 allows us to restrict our attention to those which are slice-convex.

In order to develop this lower bound, we first consider how much area can be enclosed by a given perimeter. Let $A^*(\mathcal{S})$ be the function mapping $\mathcal{S}$ to the maximum area achievable with semi-perimeter $\mathcal{S}$.

Theorem 3 *Given a semi-perimeter $\mathcal{S}$, the maximum area tile with semi-perimeter $\mathcal{S}$ is an $\frac{\mathcal{S}}{2} \times \frac{\mathcal{S}}{2}$ square if $\mathcal{S}$ is even, and is an $\left(\frac{\mathcal{S}-1}{2}\right) \times \left(\frac{\mathcal{S}+1}{2}\right)$ rectangle if $\mathcal{S}$ is odd,* i.e.*,*

$$A^*(\mathcal{S}) = \begin{cases} \left(\frac{\mathcal{S}}{2}\right)^2 & \text{if } \mathcal{S} \text{ is even} \\ \\ \left(\frac{\mathcal{S}-1}{2}\right)\left(\frac{\mathcal{S}+1}{2}\right) & \text{if } \mathcal{S} \text{ is odd} \end{cases} .$$

Proof: For a configuration C let $\mathcal{S}_x(C)$ and $\mathcal{S}_y(C)$ denote the number of columns and number of rows in the configuration respectively. Given any configuration C with semi-perimeter $\mathcal{S}$ and area A, there is a rectangular configuration C' with dimensions $\mathcal{S}_x(C) \times \mathcal{S}_y(C)$, and area $\mathcal{S}_x(C)\mathcal{S}_y(C) \geq A$. Therefore we need only consider rectangles as candidates for maximum area configurations. To find the rectangle of maximum area with semi-perimeter $\mathcal{S}$, we maximize $\mathcal{S}_x\mathcal{S}_y$ subject to $\mathcal{S}_x + \mathcal{S}_y = \mathcal{S}$. Of all pairs of integers with a certain sum, the pair with the greatest product is the one with the numbers closest together. If $\mathcal{S}$ is even, this is achieved by setting $\mathcal{S}_x = \mathcal{S}_y = \frac{\mathcal{S}}{2}$, and if $\mathcal{S}$ is odd, it is achieved when $\mathcal{S}_x$ and $\mathcal{S}_y$ differ by 1. ∎

By "inverting" the function $A^*(\mathcal{S})$, we obtain a function $\mathcal{S}^*(A)$ which is defined as *the function mapping area A to the minimum semi-perimeter of all configurations of A cells.* We also define $\mathcal{P}^*(A) := 2\mathcal{S}^*(A)$ as the function mapping an area A to the minimum perimeter of all configurations of A cells. (We show below that $\mathcal{S}^*(A)$ is attained by a slice-convex tile, so that by slice-convexity (lemma 2) its perimeter is $2\mathcal{S}^*(A)$.) A result equivalent to proposition 4 appeared in the context of graph embedding in a paper by Rosenberg [15]. We restate the result here in terms of perimeter and provide a simpler and more intuitive proof.

Theorem 4

$$\mathcal{S}^*(A) = i\left\lceil A^{1/2}\right\rceil + (2-i)\left\lfloor A^{1/2}\right\rfloor$$

where i is the smallest positive integer such that

$$\left\lceil A^{1/2}\right\rceil^{i}\left\lfloor A^{1/2}\right\rfloor^{2-i} \geq A.$$

Proof: We may bound the semi-perimeter of any configuration of A cells from below by finding the smallest semi-perimeter $\mathcal{S}$ that satisfies $A^*(\mathcal{S}) \geq A$, since this implies $A > A^*(\mathcal{S}-1)$, which means that a semi-perimeter of $\mathcal{S}-1$ is not compatible with an area of A.

Consider the following sequence of rectangles:

$$Q_0 : 0\times 0 \;\; Q_1 : 1\times 0 \;\; Q_2 : 1\times 1 \;\; Q_3 : 2\times 1 \;\; Q_4 : 2\times 2 \;\; Q_5 : 3\times 2 \;\; Q_6 : 3\times 3 \;\; \ldots$$

We call these rectangles "quasi-squares" since the dimensions of each rectangle differ by at most 1. Note that the areas of the quasi-squares in the sequence are strictly increasing after the second, the area of the ith quasi-square Q_i for $i \geq 2$ is $A^*(i)$, and the semi-perimeter for the quasi-squares increases by 1 at each step. The areas of these quasi-squares are the points at which the lower bound on the semi-perimeter increases by 1.

For an arbitrary A, there is a unique smallest quasi-square Q_j whose area is at least A. Since the area of Q_j is at least A, by selecting A cells from Q_j a semi-perimeter of at most $\mathcal{S}(Q_j)$ is achievable for A. Since the area of Q_{j-1} is smaller than A, a semi-perimeter of $\mathcal{S}(Q_{j-1}) = \mathcal{S}(Q_j) - 1$ is not achievable for A. Therefore the smallest semi-perimeter achievable for any configuration of A cells is $\mathcal{S}(Q_j)$. It is easy to see that each dimension of Q_j is either $\lfloor A^{1/2}\rfloor$ or $\lceil A^{1/2}\rceil$ and that $\mathcal{S}(Q_j)$ is exactly the semi-perimeter bound in the statement of the theorem. ∎

The above argument implies a construction technique for "perimeter-optimal" configurations, *i.e.*, configurations of specified area with minimum perimeter. An optimal configuration for any A can be constructed by arranging A cells into a *partial square* as follows. Start with a complete square with sides of length $\lfloor A^{1/2} \rfloor$. Add cells to fill in new 1-dimensional faces (completing a face before starting on a new one) until the total number of cells is A. The resulting partial square will have sides of length $\lfloor A^{1/2} \rfloor$ and $\lceil A^{1/2} \rceil$, and will measure $\lceil A^{1/2} \rceil$ in as few dimensions as possible. By theorem 4, it will have minimum semi-perimeter. If the partial squares are constructed to be slice-convex then by lemma 2 they also have minimum perimeter. This construction technique is a special case of a technique to be described in §4.2 that may be used to construct all minimum-perimeter configurations of a given area.

There is a considerable literature (see for example [11, 12] and the references therein) dealing with the generating function approach for developing expressions for the exact number of "convex polyominoes" with various properties. It should be noted that these expressions are generally non-closed-form expressions that may involve infinite expansions, and that the algorithms that we develop are based on libraries (to be described below) comprised of particular small subsets of the full collection of minimum perimeter configurations for a given area, so that the full collection does not have to be counted or generated. In general, for large areas, the latter set contains so many elements that it would be impractical to generate all of them anyway.

For the two-dimensional case considered here, a result of Ghandeharizadeh *et al* [6] (lemma 1), and a result of Rosenberg [15] (lemma 4.3), both yield an alternate bound on the semi-perimeter $\mathcal{S}$:

$$\mathcal{S} \geq \left\lceil 2\, A^{1/2} \right\rceil .$$

This bound is equivalent to $\mathcal{S}^*(A)$ (see Yackel and Meyer [18]), and by the preceding construction is therefore attained by appropriate configurations for each value of A.

Lemma 5 $\quad \displaystyle\sum_p \mathcal{P}^*(A_p) \leq \mathcal{C}$.

Proof: Follows from $\mathcal{C} = \sum_p \mathcal{P}(T_p)$ and the area constraints for (T_p). ∎

Clearly, if the configuration for each processor has minimum perimeter (*i.e.*, $\mathcal{P}(A_p) = \mathcal{P}^*(A_p)$ for all p), then the corresponding set of cell assignments achieves the lower bound on the communication measure $\mathcal{C}$, and is therefore an optimal assignment. In §5 we give some classes of domains for which such assignments are possible.

4 Other Minimum Perimeter Configurations

In this section we discuss some additional characteristics of configurations that have minimum perimeter for their area. The previous section establishes that configurations that are square or nearly square have minimum perimeter. In this section we will see that configurations of other shapes may also have minimum perimeter.

We may classify configuration shapes according to two independent characteristics. Configurations are either nearly square (dimensions differing by at most 1) or non-square. In addition, configurations are either regular (complete rectangles) or irregular. Figure 3 depicts examples of minimum-perimeter configurations in each of the four categories induced by these characteristics. Non-square regular

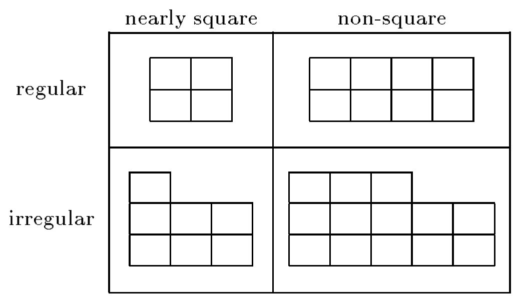


Figure 3: Categories of optimal configurations

configurations (rectangles) with minimum perimeter are discussed in §4.1. Irregular minimum-perimeter configurations and a technique for constructing all optimal configurations of a given area are taken up in §4.2. In the following lemmas we present two useful characterizations of configurations with minimum perimeter.

Lemma 6 *A configuration of A cells with semi-perimeter $\mathcal{S} > 2$ has minimum perimeter if and only if it is a slice-convex tile satisfying*

$$A^*(\mathcal{S}-1) < A \; . \tag{2}$$

Proof: From lemma 2 it follows that only slice-convex configurations may have minimum perimeter. To see that only tiles (*i.e.*, connected configurations) may have minimum perimeter, consider a configuration containing two disconnected sub-tiles denoted by T_1 and T_2. By translating sub-tile T_1 it is always possible to connect T_1 and T_2 (see figure 4), thereby decreasing the perimeter of the configuration by at least 2. If (2) holds, then A is greater than the largest area for which a smaller semi-perimeter is achievable, so the configuration has minimum semi-perimeter. This, along with slice-convexity imply the configuration has minimum perimeter. ∎

Lemma 7 *A configuration of A cells has minimum perimeter if and only if it is slice-convex and its minimum circumscribing rectangle has perimeter $\mathcal{P}^*(A)$.*

Proof: The lemma follows from the fact that a rectangle has the same perimeter as any slice-convex configuration of cells it minimally circumscribes. ∎

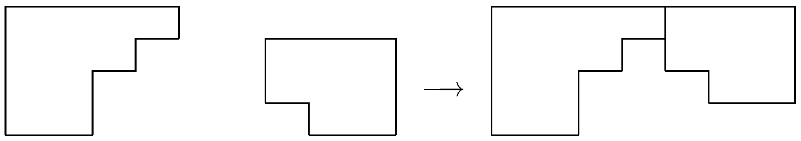


Figure 4: translating sub-tiles to decrease perimeter

4.1 Optimal Rectangles

Using previous results, we can characterize the rectangular blocks that have minimum perimeter. Below we use k to denote the difference between the height and width of a rectangular block.

Theorem 8 *An $x \times (x+k)$ or an $(x+k) \times x$ rectangular block is perimeter-optimal if and only if*

$$\begin{gathered} k \textit{ is even and } 1 + \tfrac{k}{2}(\tfrac{k}{2} - 1) \leq x \\ \textit{or} \\ k \textit{ is odd and } 1 + \left(\tfrac{k-1}{2}\right)^2 \leq x. \end{gathered}$$

Conversely, an $x \times (x+k)$ or an $(x+k) \times x$ rectangle is perimeter-optimal if and only if k is at most

$$\max\left\{2 \ \textit{round}(x^{1/2}), 2\left\lfloor x^{1/2} - 1\right\rfloor + 1\right\}$$

where round(x) rounds x to the nearest integer. (Since the fractional part of $x^{1/2}$ is never $1/2$ for integer x, we do not care if round$(1/2) = 0$ or 1.)

Proof: To prove the first part of the theorem, we simply apply the optimality test. By lemma 6, an $x \times (x+k)$ block is optimal if and only if

$$(3) \qquad \left\lceil \frac{2x+k-1}{2} \right\rceil^r \left\lfloor \frac{2x+k-1}{2} \right\rfloor^{2-r} < x^2 + kx$$

where $r \equiv 2x + k - 1 (\mod 2)$.

If k is even, (3) reduces to

$$\begin{array}{rcl} \left\lceil \frac{2x+k-1}{2} \right\rceil \left\lfloor \frac{2x+k-1}{2} \right\rfloor & < & x^2 + kx \\ \iff \quad (x + \frac{k}{2})(x + \frac{k}{2} - 1) & < & x^2 + kx \\ \iff \quad \frac{k}{2}(\frac{k}{2} - 1) & < & x. \end{array}$$

The integrality of both sides of the inequality allows us to derive the desired result.

If k is odd, (3) reduces to

$$\begin{array}{rcl} \left\lfloor \frac{2x+k-1}{2} \right\rfloor^2 & < & x^2 + kx \\ \iff \quad \left(x + \frac{k-1}{2}\right)^2 & < & x^2 + kx \\ \iff \quad \left(\frac{k-1}{2}\right)^2 & < & x. \end{array}$$

To prove the second part of the theorem, we show that $2\left\lfloor (x-1)^{1/2}\right\rfloor + 1$ and $2\ \mathrm{round}\left(x^{1/2}\right)$ are the largest odd and even integers respectively satisfying (3).

To prove the result for the odd numbers, we start with the expression for x in terms of odd k.

$$\begin{array}{rrcl} & \left(\frac{k-1}{2}\right)^2 + 1 & \leq & x \\ \iff & \frac{k-1}{2} & \leq & (x-1)^{1/2}. \end{array}$$

Since the LHS of the last inequality is integer, we may take the floor of the RHS.

$$\begin{array}{rrcl} \iff & \frac{k-1}{2} & \leq & \left\lfloor (x-1)^{1/2}\right\rfloor \\ \iff & k & \leq & 2\left\lfloor (x-1)^{1/2}\right\rfloor + 1. \end{array}$$

Since the RHS of the last inequality is an odd integer, $k = 2\left\lfloor (x-1)^{1/2}\right\rfloor + 1$ is the largest odd integer satisfying (3).

To prove the result for the even numbers we write $x^{1/2}$ in the form $x = r + f$ where r is the integer part and $f \in [0,1)$ is the fractional part. If $f < \frac{1}{2}$ then $2\ \mathrm{round}\left(x^{1/2}\right) = 2r$. If $f > \frac{1}{2}$ then $2\ \mathrm{round}\left(x^{1/2}\right) = 2r + 2$. (For integer x, f is never $\frac{1}{2}$ so $\mathrm{round}\left(x^{1/2}\right)$ is uniquely defined for integer x.)

If $f < \frac{1}{2}$ and $2\ \mathrm{round}\left(x^{1/2}\right) = 2r$ then $k = 2r$ satisfies (3) because

$$\frac{k}{2}\left(\frac{k}{2} - 1\right) = r(r-1) = r^2 - r < r^2 + 2fr + f^2 = x$$

and $k = 2r + 2$ violates (3) because

$$\frac{k}{2}\left(\frac{k}{2} - 1\right) + 1 = (r+1)r + 1 = r^2 + r + 1 > r^2 + 2fr + f^2 = x.$$

If $f > \frac{1}{2}$ and $2\ \mathrm{round}\left(x^{1/2}\right) = 2r + 2$ then $k = 2r + 2$ satisfies (3) because

$$\frac{k}{2}\left(\frac{k}{2} - 1\right) = (r+1)r = r^2 + r < r^2 + 2fr + f^2 = x$$

and $k = 2r + 4$ violates (3) because

$$\frac{k}{2}\left(\frac{k}{2} - 1\right) = (r+2)(r+1) = r^2 + 3r + 2 > r^2 + 2fr + f^2 = x.$$

Therefore $k = 2\ \mathrm{round}\left(x^{1/2}\right)$ is the largest even integer satisfying (3). ∎

Note that the first part of the theorem shows that if a particular rectangle is optimal, then by increasing both dimensions by the same amount, the resulting larger rectangle is also optimal. Theorem 8 is addressed graphically in figure 5, which shows the dimensions of all rectangles with $x \leq 30$ that have minimum perimeter. The integral points on the diagonal line in the figure represent the squares, and the outer boxes represent the most-skewed rectangles with optimal perimeter. All integer points between (and including) the boxed points correspond to rectangles with minimum perimeter. Table 1 lists dimensions of the most skewed optimal rectangles corresponding to the boxed points above the diagonal. In the genetic algorithms to be developed below, we start with minimum-perimeter configurations that are "nearly rectangular". Theorem 8 can be used to show that for a given area A_p, the number of such "near rectangles" is limited by the skew-parameter k which is of order $A_p^{1/4}$.

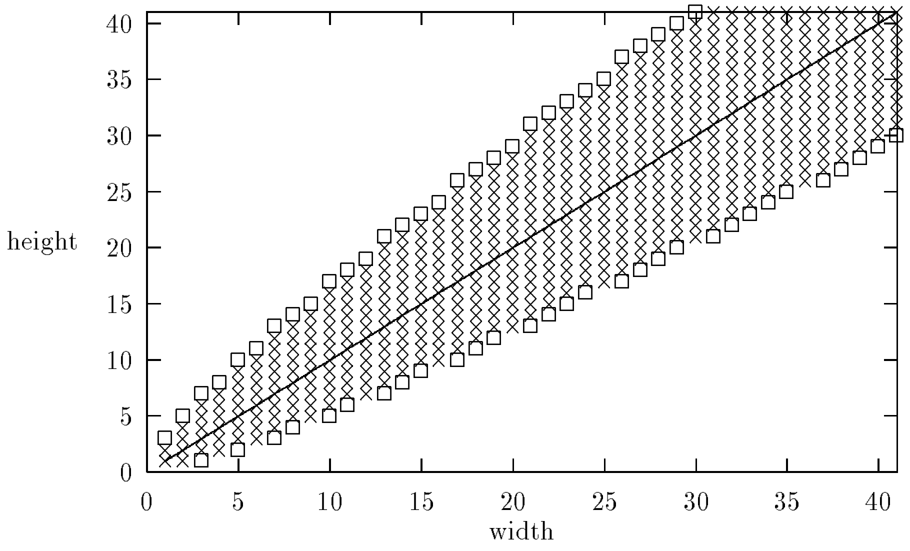


Figure 5: Dimensions of rectangles with optimal perimeter

k	Most-skewed perimeter-optimal rectangles $x \times (x+k)$				
2	1×3				
3	2×5				
4	3×7	4×8			
5	5×10	6×11			
6	7×13	8×14	9×15		
7	10×17	11×18	12×19		
8	13×21	14×22	15×23	16×24	
9	17×26	18×27	19×28	20×29	
10	21×31	22×32	23×33	24×34	25×35
11	26×37	27×38	28×39	29×40	30×41

Table 1: Some most-skewed perimeter-optimal rectangles

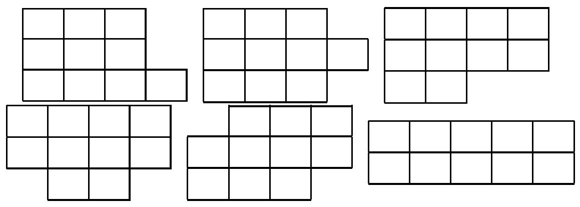


Figure 6: Optimal configurations of area 10

4.2 Optimal Irregular Tiles

Theorem 4 suggests a procedure for generating *all* the minimum-perimeter configurations for a given area A (although these are too numerous to actually compute for most large areas). In this discussion we consider two configurations to be *equivalent* if one can be transformed into the other by rotation and/or reflection. Using the theorem, all the possible circumscribing rectangles for perimeter-optimal configurations of area A can be specified. For each of these rectangles, any slice-convex subset of A cells forms a minimum-perimeter configuration. We say a cell is a *corner cell* of a configuration if it is not between two cells in any slice. By removing a corner cell from a slice-convex configuration, convexity is maintained. Therefore, starting from a minimum-perimeter rectangle, a minimum-perimeter configuration can be constructed by iteratively removing an appropriate number of corner cells. For example, given an area of 10, the minimum-perimeter configurations are generated as follows. $\mathcal{S}^*(10) = 7$, so the rectangles of semi-perimeter 7 are considered. The possibilities are 1×6, 2×5, and 3×4. A 1×6 rectangle can't enclose 10 cells, so all minimum-perimeter configurations are circumscribed by either 2×5 or 3×4 rectangles. Therefore all minimum-perimeter configurations of area 10 are represented by the configurations in figure 6.

The first five configurations in the figure represent all the possibilities for configurations enclosed by 3×4 rectangles: the first two are constructed by removing two corners from the same column, the next two by removing two corners from the same row, and the fifth by removing two corners from different rows and columns. There is only one possible configuration of area 10 contained in a 2×5 rectangle.

By examining the above technique, we are able to identify the cases in which there is a unique minimum-perimeter configuration of given area. Given an area A, if there is a unique rectangle with semi-perimeter $\mathcal{S}^*(A)$ and area $\geq A$, then all minimum-perimeter configurations of area A are circumscribed by that rectangle. Furthermore, if the area of that unique enclosing rectangle is A or $A+1$ then there is a unique optimal configuration of area A, because all removals of either zero or one corner result in equivalent configurations. We also show that all other cases lead to non-uniqueness.

Theorem 9 *There is a unique minimum-perimeter configuration of area A if and only if A can be expressed as $A = k^2$, $k(k+1)$, or $k(k+1) - 1$ for positive integer*

k.

Proof: If $A = k^2$ then $\mathcal{S}^*(A) = 2k$. The unique enclosing rectangle with semi-perimeter $2k$ and area at least A has dimensions $k \times k$ (any other rectangle with semi-perimeter $2k$ has area less than k^2). Similarly, if $A = k(k+1)$, or $k(k+1)-1$ and $k \geq 2$, then $\mathcal{S}^*(A) = 2k+1$, and the unique rectangle with semi-perimeter $2k+1$ and area at least A has dimensions $k \times (k+1)$.

To show that these are the only classes of areas which have unique optimal configurations, consider the fact that such an area A must have only one possible enclosing rectangle with perimeter $\mathcal{S}^*(A)$. Since $\mathcal{S}^*(A)$ corresponds to an enclosing square or quasi-square, it follows that the unique enclosing rectangle must be this square or quasi-square. This means that the area A is expressible as either $k^2 - j$ or $k(k+1) - j$ for positive integer k and non-negative integer j. For areas expressed as $A = k^2 - j$ with $j \geq 1$, a $(k+1) \times (k-1)$ rectangle is a second enclosing rectangle with semi-perimeter $2k$ because $(k+1)(k-1) \geq k^2 - j$. For areas expressed as $k(k+1) - j$ with $j > 2$ there can not be a unique optimal configuration because there are at least two ways of removing j corners from a $k \times (k+1)$ rectangle. ∎

The preceding argument proves that squares, quasi-squares, and quasi-squares minus one corner are the unique optimal configurations for their areas and that all other areas have alternate optimal configurations. Similar reasoning can be used to show that for areas of the form $k^2 - 1$ with $k \geq 2$, there are exactly two optimal configurations: a $k \times k$ square with one corner removed and a complete $(k-1) \times (k+1)$ rectangle.

Another interesting fact about the set of optimal configurations of a given area is that there can be at most one rectangular configuration in the set. To prove this we make use of the following lemma.

Lemma 10 *For (unordered) pairs of positive numbers, two distinct pairs with identical pair sums have different pair products,* i.e., *for positive numbers x_1, y_1, x_2, y_2, if $x_1 + y_1 = x_2 + y_2$ and $\{x_1, y_1\} \neq \{x_2, y_2\}$ then $x_1 y_1 \neq x_2 y_2$.*

Proof: Write x_2 as $x_1 + k$. Then $y_2 = y_1 - k$. Now assume $x_1 y_1 = x_2 y_2$. Substituting into this last equation, we get

$$x_1 y_1 = (x_1 + k)(y_1 - k).$$

Simplifying, we get

$$k(y_1 - x_1 - k) = 0,$$

in other words $k = 0$ or $k = y_1 - x_1$. In the first case $x_1 = x_2$ and $y_1 = y_2$, and in the second case $x_1 = y_2$ and $y_1 = x_2$, a contradiction. ∎

The lemma tells us that all rectangles with a given semi-perimeter have different areas and implies that if a rectangular configuration of area A is optimal then it is the only optimal rectangular configuration with area A (any other optimal configurations will be irregular).

5 Optimal Assignments

In order to achieve the lower bound for the objective $\mathcal{C}$, each configuration must have perimeter exactly $\mathcal{P}^*(A_p)$. Thus, ideally the cells assigned to each processor form perimeter-optimal configurations that fill the domain exactly. In this section we exhibit some classes of domains for which such tilings can be constructed. Some related results are also presented in §6. The genetic algorithms to be discussed in §7 deal with the general case in which such "perfect" tilings are not known to exist.

5.1 Optimal Tilings with Rectangles

One class of problems that have easily obtainable optimal solutions are instances in which the domain is a $M_1 \times M_2$ rectangular grid that can be tiled with perimeter-optimal rectangles. In particular, if N can be factored as $f_1 f_2$ where f_1 divides M_1, f_2 divides M_2 and $\frac{M_1}{f_1} \times \frac{M_2}{f_2}$ rectangles are perimeter-optimal, then such a tiling is possible. Below is an optimal assignment for such an instance: a 6×18 grid with 6 processors, each of which has a load of 18.

1	1	1	1	1	1	2	2	2	2	2	2	3	3	3	3	3	3
1	1	1	1	1	1	2	2	2	2	2	2	3	3	3	3	3	3
1	1	1	1	1	1	2	2	2	2	2	2	3	3	3	3	3	3
4	4	4	4	4	4	5	5	5	5	5	5	6	6	6	6	6	6
4	4	4	4	4	4	5	5	5	5	5	5	6	6	6	6	6	6
4	4	4	4	4	4	5	5	5	5	5	5	6	6	6	6	6	6

Figure 1 in §1 is an example of a non-rectangular domain optimally tiled with rectangles with different orientations.

5.2 Optimal Tiling with Irregular Tiles

Irregular tiles can fit together to tile many grids. The example below shows how irregular optimal tiles of area 10 can fit together in an optimal tiling.

1	1	1	1	4	4	4	4	5	5	6	6	6
1	1	1	1	4	4	4	4	5	5	6	6	6
1	1	2	2	3	3	4	4	5	5	5	6	6
2	2	2	2	3	3	3	3	5	5	5	6	6
2	2	2	2	3	3	3	3	8	8	8	7	7
								8	8	8	7	7
								8	8	7	7	7
								8	8	7	7	7

This example also demonstrates the technique of optimally tiling by decomposing the domain into subdomains which can each be optimally tiled by a proportional subset of the processors. The domain above can be split into four 5×4 rectangles, each of which can be tiled optimally with two processors. Such a divide and conquer approach to tiling is a method of constructing optimal tilings for large domains with complex shapes.

6 Minimum Diversity Assignments

Before discussing genetic algorithms for the general minimum perimeter problem, we consider a closely related problem and a technique for generating optimal solutions that is relevant to our GA approach. The minimum diversity problem presented in [5] is a parallel database design problem with the same assignment constraints as the minimum perimeter problem, but has a different objective function. In this database application [5], we wish to assign grid cells of a rectangular domain to processors in order to minimize the total number of *distinct* processors that appear in the slices of the grid. A typical computation in the database system accesses all the data in a particular slice of the grid, and the processors assigned to cells in the slice must participate by communicating with a *coordinating* processor. (This contrasts with the minimum perimeter problem, in which communication occurs between processors that are assigned to adjacent cells.) We assume a communication overhead is associated with initiating and terminating a query on each of the processors associated with a slice. The goal is to minimize total overhead summed over all slices while balancing the workload between the processors (the constraints of this nonlinear semi-assignment problem are the same as those of the minimum perimeter problem). If we define the diversity of slice s, denoted by ν_s, to be the number of distinct processors in slice s of the grid, then our objective is to minimize $\theta_{\mathrm{total}} := \sum_s \nu_s$. We refer to this as the diversity minimization problem. (It is easy to formulate this objective function as a sum of fixed-charge functions corresponding to unit charges for the appearance of a processor (any number of times) in a slice; the minimum perimeter problem objective, on the other hand, is most directly modeled as a sum of quadratic terms corresponding to the assignment of adjacent cells to different processors.)

In Ghandeharizadeh *et al* [5] we considered a general D-dimensional assignment problem and showed that minimizing diversity is equivalent to minimizing the sum of the "D-perimeters" for the processors, where the D-perimeter for a processor is defined to be the number of $(D-1)$-dimensional slices in which a processor appears. In two dimensions, the D-perimeter of a configuration of cells is therefore the semi-perimeter of the configuration. Because semi-perimeter is not affected by a permutation of the rows or columns of the grid, the cells belonging to a processor do not have to be slice-convex or even connected in order to form a configuration with minimum D-perimeter (see figure 2). However, minimum-perimeter tiles are examples of configurations that have minimum D-perimeter. Cases such those considered in §5 in which tilings are comprised of minimum-perimeter configurations correspond to instances in which the minimum-perimeter and minimum diversity problems have common optimal solutions.

We now present another class of problems for which optimal solution to the diversity minimization problem may be constructed, and relate these problems to minimum perimeter problems on toroidally connected grids. Note that tiles can "wrap around" the top or side of the grid without incurring any extra diversity since they remain in the same slices. We can therefore think of the domain as lying on the surface of a torus, *i.e.*, the top row of the grid is adjacent to the bottom row, and the left-most column is adjacent to the right-most. So, using the concept of connectedness in the toroidal sense, we will develop another class of problems that have common optimal tilings for the diversity and perimeter minimization problems.

Theorem 11 *A toroidal domain can be tiled with partial square tiles of size A if A divides both the number of rows and columns evenly.*

Proof: See [17]. ∎

The constructive proof of this theorem is based on the use of diagonal strips of block-plus-fringe optimal shapes of the following type:

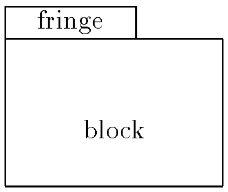


Figure 7 illustrates a minimum (toroidal) perimeter and minimum diversity assignment generated by this diagonal tiling process on a 7×7 grid with 7 processors.

1	6	6	6	7	7	7
1	1	1	2	7	7	7
1	1	1	2	2	2	3
3	3	4	2	2	2	3
3	3	4	4	4	5	3
5	6	4	4	4	5	5
5	6	6	6	7	5	5

Figure 7: Diagonal tiling of a toroidal domain

7 Parallel Genetic Algorithms

For the diversity minimization and minimum perimeter problems described above, the techniques of genetic algorithms (GA's) have been successful in computing optimal and near-optimal solutions for large-scale problems.

The GA approach to solving combinatorial optimization problems involves maintaining a "population" of points in the search space. Each iteration of the GA involves the following steps: the objective value of each individual in the population is evaluated by a "fitness function", the most fit individuals are "selected" with high probability to form a mating pool, and members of the mating pool "combine" assignments through a random "crossover" operation to form a new population. The hope is that good individuals will combine to form even better offspring.

Before describing the GA approach to cell assignment we present a heuristic that is used in the GA. This heuristic is based upon the availability of a "library" of optimal shapes consisting of minimum-perimeter configurations that are nearly rectangular. Specifically, each configuration in the library consists of a rectangular block plus (possibly) a fringe, analogous to the configurations discussed in the preceding section. Although not all minimum-perimeter configurations have this particular form (see figure 6), the heuristic includes procedures for altering shapes that allow arbitrary optimal shapes to be constructed during the solution process. Moreover, the subset of optimal shapes in the library is sufficiently large so as to yield optimal or near optimal solutions for many cases (see the results of the previous section and [2]). Finally, the number of configurations in the library is of order $A_p^{1/4}$, and, thus, even in the case $A_p = 1000$ (the largest area in the test problems below), the library contains only 9 elements. We have developed a heuristic to approximate optimal partitions by using this block-fringe viewpoint as an input. As input, our procedure is given a list of shapes from the optimal shape library, one shape (block-fringe) for each processor. The heuristic has two stages: in stage 1 it places a block of the specified shape for each processor in the grid, splitting the blocks if necessary to get them to fit. Stage 1 is completed by assigning the fringes in the remaining free grid cells so that they line up as much as possible with the corresponding blocks. The result is a *feasible* assignment in which each processor has a configuration of cells which is approximately the same as an optimal tile under some permutation of the rows and columns. In stage 2 the heuristic performs a local search of the problem space in a fashion similar to the Kernighan - Lin heuristic [9]. Using the cell-processor assignment from stage 1 as its starting point, it selects pairs of cells and evaluates the change in objective that would result from swapping the processors in the two cells. If the swap does not worsen the objective, then the cell assignments are interchanged. The number of swaps that are evaluated is provided as an input to the heuristic so that the user has control over the running time of stage 2.

The common feasible solution set of the diversity minimization and perimeter minimization problems is the set of all partitions of the given grid into N sets of the specified sizes (we restrict our attention to cases where the sizes are as similar as possible, *i.e.,* they differ by at most 1). The number of feasible solutions, even for small problems, is very large: for a 5×5 grid to be partitioned into 5 equal size sets, the number of feasible partitions is more than 6×10^{14}. Rather than developing a GA with an individual corresponding to each feasible solution, we take a high-level approach that incorporates knowledge about the form of perimeter-optimal configurations. The search space explored by the GA is not the space of all possible partitions of the domain among N processors, but rather the space of all possible choices of perimeter-optimal shapes for the N processors. With this encoding, an "individual" does not correspond to a particular feasible solution, but instead represents a potential assignment of perimeter-optimal configurations to processors.

This set of "shape" assignments is then input to our cell-assignment heuristic to obtain a feasible solution. Therefore the objective value produced by this heuristic is a natural fitness function for our GA.

We implemented our GA on a Thinking Machines Corporation Connection Machine CM-5 in the Computer Sciences Department at the UW-Madison. We used the "host-node" programming paradigm in which a host processor initiates and coordinates the node processors. This allows the implementation of a "fork-join" parallel program which fits the genetic algorithm's characteristics [17]. Each node on the parallel machine is responsible for two individuals. The nodes execute fitness function evaluation and crossover in parallel, sending the fitness values of the new generation back to the host. After receiving the fitness values, the host selects the pairs of individuals that will crossover and informs the nodes of the "names" of these so that the nodes can then exchange the corresponding genetic information.

Diversity minimization problems as large as 1000×1000 grids partitioned among 1000 processors (corresponding to 1 billion assignment variables in the original problem formulation) were processed with this GA. The GA actually produced provably optimal solutions to many of the medium-sized problems (involving a few thousand variables) as well as for one of the 1 million variable problems. The computing time varied over the test problems from 0.4 seconds to 6 seconds per GA iteration. Within 100 iterations, a solution within at most 4% of the optimal value was computed for each of the test problems studied. For complete details see [1].

The GA for the minimum perimeter problem was tested on a similar suite of problems; many of the medium-sized problems (that is, problems with fewer than $1,000,000$ binary variables in a Quadratic Assignment formulation) were solved to optimality within 10 minutes, and a $1,815,848$ variable problem was solved to optimality in less than 2.5 minutes. Processing some extremely large test problems took about 20 minutes for the 512×512 grid partitioned among 512 processors, and a little more than 2.5 hours for the 1000×1000 grid partitioned among 1000 processors. The algorithm was always able to compute solutions within 2.1% of the lower bound within 20 generations. Figure 8 shows an optimal solution generated by the GA for the 7×7 grid partitioned among 7 processors (compare with the toroidal 7×7 case in figure 7). A subset of these test problems (formulated as quadratic assignment problems) was also run on a GRASP algorithm (see [10]) for general quadratic assignment problems. GRASP, designed to handle more general problems, cannot readily make use of the special problem structure that leads to the optimal shape library that is a key feature of our GA approach, so it is not surprising that the largest problem that it was able to solve to optimality was the 7×7 grid referenced above. For problems larger than this, neither the solution time nor quality was comparable to that attained with the GA, and time limitations (we allowed problems to run for not more than 6 hours) prevented GRASP from running problems on grids larger than 13×13. (Details of this computational comparison are given in [2].)

We have also compared our GA against the popular spectral partitioning algorithm [14] and the geometric mesh partitioning method [13]. We obtained MATLAB versions of both algorithms from the ftp site referenced in [7]. Both methods were

1	1	1	2	2	3	3
1	1	2	2	2	3	3
1	1	2	2	3	3	3
4	4	4	4	5	5	5
4	4	4	5	5	5	5
6	6	6	6	7	7	7
6	6	6	7	7	7	7

Figure 8: Minimum Perimeter Solution for the 7×7 grid

run on a uniprocessor Sun SPARCstation 20. Our GA, written in C, using the CMMD libraries on a thirty-two node partition of the CM-5, significantly outperformed both methods in terms of the quality of the final solution in all cases (ranging from a 32×30 domain to be partitioned among 64 processors, to the large 512×512 domain to be partitioned among 512 processors) and it ran significantly faster than the spectral method in all cases. The serial geometric mesh partitioner was almost as fast as the GA on the smaller problems, but was slower (or did not run because of memory limitations) on the larger problems.

Finally, we observe that a similar high level approach may be worthwhile for other classes of fixed-charge assignment problems. For example, in multicommodity problems with fixed-charges it would be possible to develop for each commodity a collection of optimal and suboptimal fixed-charge assignments, define an "individual" as a specific such collection for each commodity, and then employ a heuristic within the GA that combines these "patterns" to produce a good overall solution and a corresponding "fitness" measure for each "individual".

8 Conclusions and Future Work

We have formalized as nonlinear network optimization models problems of partitioning tasks among processors to minimize interprocessor communication for parallel domain decomposition. Lower bounds on the objective functions have been developed and we have demonstrated how the bound is attained when the domain can be tiled with minimum-perimeter tiles. We have presented characteristics of minimum-perimeter tiles and demonstrated how a library consisting of a suitable subset of such tiles can be used to develop a genetic algorithm capable of generating good quality (and, in some cases, provably optimal) solutions to enormous problems that have up to one billion variables when formulated as quadratic semi-assignment problems. Continuing work in this area includes modifying the algorithms to generate optimal or near-optimal solutions for arbitrary domains and extending the results to three-dimensional domains, other data partitioning problems, and other types of fixed-charge networks.

References

[1] R.J. Chen, R.R. Meyer, and J. Yackel. A genetic algorithm for diversity minimization and its parallel implementation. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 163–170. Morgan Kaufman, 1993.

[2] I. T. Christou and R. R. Meyer. Optimal equi-partition of rectangular domains for parallel computation. *Journal of Global Optimization*, 8:15–34, 1996.

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

[4] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*, pages 60–62. W.H. Freeman and Company, New York, 1979.

[5] S. Ghandeharizadeh, G. L. Schultz, R. R. Meyer, and J. Yackel. Optimal processor assignment for parallel database design. In *Proceedings of the Fifth SIAM Conference on Parallel Processing for Scientific Computing*. SIAM, 1992.

[6] S. Ghandeharizadeh, G. L. Schultz, R. R. Meyer, and J. Yackel. Optimal balanced assignments and a parallel database application. *ORSA J. on Computing*, 5:151–167, 1993.

[7] 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.

[8] W. D. Gropp and D. E. Keyes. Domain decomposition methods in computational fluid dynamics. Technical Report 91-20, ICASE, February 1991.

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

[10] 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.

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

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

[13] 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.

[14] 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.

[15] A. L. Rosenberg. Encoding data structures in trees. *JACM*, 26(4):668–689, October 1979.

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

[17] J. Yackel. *Minimum-Perimeter Tiling for Optimization of Parallel Computation*. PhD thesis, University of Wisconsin - Madison, 1993. Computer Sciences technical report #1170.

[18] 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.