SIAM J. COMPUT.
Vol. 17, No. 6, December 1988

010



# THE DECOMPOSITION OF A RECTANGLE INTO RECTANGLES OF MINIMAL PERIMETER*

T. Y. KONG†, DAVID M. MOUNT‡, AND A. W. ROSCOE§

**Abstract.** We solve the problem of decomposing a rectangle $R$ into $p$ rectangles of equal area so that the maximum rectangle perimeter is as small as possible. This work has applications in areas such as flexible object packing and data allocation. Our solution requires only a constant number of arithmetic operations and integer square roots to characterize the decomposition, and linear time to print the decomposition. The discrete analogue of the problem in which the rectangle $R$ is replaced by a rectangular array of lattice points is also considered, and three heuristic methods of solution are given. All of the heuristic methods operate by finding a discrete approximation to our optimal decomposition of $R$, but with different tradeoffs between the accuracy of the approximation and running time.

**Key words.** rectangle decomposition, flexible packing, digitization

**AMS(MOS) subject classifications.** 52A45, 68Q25, 68U05

**1. Introduction.** A fundamental problem in geometrical and combinatorial computing is how to decompose a large object into smaller objects subject to various constraints. By a *decomposition* of a region $R$ we mean a finite set of closed regions whose union is $R$ and whose interiors are pairwise disjoint. The regions of the decomposition need not be connected. Decomposition problems generally fall into one of two classes. In the first class, the objects have fixed dimensions, as in the knapsack and bin-packing problems [8], [11], [12]. In the second class the objects satisfy certain properties, as in decompositions of simple polygons into polygons that are star-shaped [2], convex [5], [17], triangular [7], or involve rectangles and rectilinear polygons [6], [9]. We consider a middle ground between these two classes in which the objects are of some specified area (or volume), but their shapes are not fully specified. In other words, the objects are flexible. The objective is to produce a decomposition in which the objects are not severely stretched; that is, they are nearly circular or square. More specifically, we consider the following problem involving the decomposition of a rectangle into rectangles of equal area or measure.

RECTANGULAR DECOMPOSITION. Given a rectangle $R$ of height $A$ and width $B$, and given an integer $p$, decompose $R$ into $p$ rectangles of equal area in such a way that the maximum rectangle perimeter is minimized.

Intuitively, the problem is to make the $p$ rectangles in the decomposition of $R$ as close to squares as we can. A special case of this problem in which $R$ is a square was solved in [15]. The solution to the rectangular decomposition problem is a straightforward generalization of decomposition presented there, but the proof of optimality in the rectangular case is significantly more complex. A related problem with unconstrained areas was considered for the square in [1].

If we remove the restriction that $R$ be decomposed into *rectangles* then we obtain another interesting problem. Define the *projection-perimeter* of a measurable plane set to be twice the sum of the lengths (measures) of its projections on the coordinate axes.

* Received by the editors April 7, 1986; accepted for publication (in revised form) February 10, 1988.
† Department of Computer Science, Ohio University, Athens, Ohio 45701. This author gratefully acknowledges the support of the Air Force Office of Scientific Research under contract F-49620-85-K-0009.
‡ Department of Computer Science, University of Maryland, College Park, Maryland 20742.
§ Oxford University Computing Laboratory, Oxford, United Kingdom OX1 3QD.

GENERAL DECOMPOSITION. Given a rectangle $R$ on the Cartesian plane with sides parallel to the coordinate axes of height $A$ and width $B$, and given a positive integer $p$, decompose $R$ into $p$ regions of equal measure in such a way that the maximum projection-perimeter is minimized.

Here the terms *height* and *width* denote the length of the projection onto the $y$- and the $x$-axis, respectively.

In this paper we solve the rectangular decomposition problem. We also show that for certain values of $A$, $B$, and $p$, the optimal rectangular decomposition is, in fact, a solution to the general decomposition problem. For all values of $A$, $B$, and $p$, we prove that the optimal rectangular decomposition is a solution to the following problem, which can be thought of as a compromise between the rectangular and general decomposition problems. Define a *pseudorectangle* to be any set that is congruent to a Cartesian product $P \times Q$, where $P$ and $Q$ are measurable subsets of the real line. (In particular, every rectangle is also a pseudorectangle.)

PSEUDORECTANGULAR DECOMPOSITION. Given a rectangle $R$ on the Cartesian plane with sides parallel to the coordinate axes, of height $A$ and width $B$, and given a positive integer $p$, decompose $R$ into $p$ pseudorectangles of equal measure in such a way that the maximum projection-perimeter is minimized.

We consider a model of computation in which unit charge is assessed for $+, -, *, /, \leqq, \lfloor\ \rfloor$ and integer square root on rational numbers representable using $O(\log(p+A+B))$ bits of precision. In this model of computation our optimal rectangular decomposition can be characterized in constant time and output in $O(p)$ time.

These decomposition problems and their higher-dimensional counterparts have a number of applications:

- *Flexible object packaging.* There are $p$ sacks of fluid that are to be placed in a box of volume $A$, and each sack is to fit into a rectangular box of volume $A/p$. The dimensions of the partitions may vary, but to minimize the stress on each sack, it is desirable to make the boxes as nearly cubical as possible.
- *Circuit decomposition.* Given a large circuit, laid out on an $A \times B$ board (for example, a mesh of computer processors), decompose the circuit by slicing the board into $p$ rectangles of equal area. To minimize the interboard communications, the perimeter of each board should be as small as possible.
- *Flexible circuit layout.* In VLSI layout, a designer wants to place $p$ functionally identical circuits on a rectangular chip of area $A$. Each circuit can be deformed into an arbitrary rectangle, as long as the area is equal to $A/p$. However, highly eccentric rectangles lead to long wire lengths. It is desirable to reduce the length of the longest side of each rectangle, which implies that its perimeter is minimized.

An analogue of the general decomposition problem can be posed on a rectangular array of lattice points.

LATTICE DECOMPOSITION. Given positive integers $A$, $B$, and $p$, partition an $A \times B$ array of lattice points into $p$ subsets each containing at most $\lceil AB/p \rceil$ points, in such a way that the maximum projection-perimeter of a subset is minimized.

This arises in the following data-allocation problem for parallel computation of tables.

- *Data allocation for parallel computers.* We wish to compute all of the values of a binary function $f$ on the Cartesian product $S \times T$, where $|S| = A$ and $|T| = B$. The computation is to be performed in parallel on $p$ identical processing units. The function values are computed as follows. The $i$th processor computes the values of $f$ on some subset $W_i$ of $S \times T$. The sets $W_i$, $1 \leqq i \leqq p$, form a partition of $S \times T$. To minimize computation time, each processor is assigned at most $\lceil AB/p \rceil$ function values to compute. Each processor has a small amount of local memory used for storing its operands. The objective is to minimize this storage. The amount of storage used by the $i$th processor is equal to the number of operands needed to compute the values of $W_i$, which equals one half of the projection-perimeter of $W_i$.

The data allocation problem was in fact the initial motivation for this work [13], [15]. Although we will not present an exact solution to the lattice decomposition problem, we will give heuristic methods based on the idea of approximating our solution to the pseudorectangular decomposition problem on the discrete lattice. The quality of these approximate solutions will be good when $A$ and $B$ are large relative to $p$.

This paper is organized as follows. In § 2 we solve the rectangular decomposition problem and prove that the optimal rectangular decompositions are also optimal solutions to the pseudorectangular decomposition problem. This work is based on two interesting combinatorial lemmas that give lower bounds on the amount of stretching and compressing that must occur when $p$ rectangles of equal area are packed into an $A\times B$ rectangle. In § 3 we describe three procedures for approximating, or *digitizing*, the geometric decomposition described in § 2 on an integer lattice of height $A$ and width $B$. These digitization procedures provide different tradeoffs between desirable characteristics of the digitization and running time. Ideally, the digitization should respect the geometric solution and preserve areas as closely as possible. We say that a digitization is *equitable* if it has the property that the area of each digitized region is at most the ceiling of the area of the original region. We give an algorithm for computing an equitable digitization by reduction to the problem of finding a feasible flow in a graph. This algorithm runs in time polynomial in $A+B+p$. We give two efficient algorithms that produce approximations to an equitable digitization.

**2. Optimal decomposition of a rectangle.** Let $R$ be a rectangle in the Cartesian plane, with sides parallel to the coordinate axes, of height $A$ and width $B$. Let $p$ be any positive integer. In this section we solve the problem of how to decompose $R$ into $p$ rectangles $R_1, R_2, \cdots, R_p$ of equal area in such a way that the maximum of the perimeters of the $R_i$'s is as small as possible. Our solution actually minimizes the maximum projection-perimeter over all decompositions of $R$ into pseudorectangles of equal measure.

If the rectangle $R$ is sufficiently thin relative to $p$, in particular if $p\leqq \max(A/B, B/A)$, then it is easy to see that the optimal decomposition results by simply partitioning the longer side of $R$ into $p$ equal parts. So we shall henceforth assume that $p>\max(A/B, B/A)$.

From now on, whenever we refer to a pseudorectangle we will assume that it is so oriented that its sides are parallel to the coordinate axes. Unless otherwise stated, the term *projection* will mean a projection on one of the coordinate axes. The two projections of a pseudorectangle are generally not connected sets. If a projection is a finite union of disjoint open and closed intervals then by the *length* of the projection we mean the sum of the lengths of those intervals. More generally the length of a projection is taken to mean its (Lebesgue) measure. Analogously, when we refer to the *area* of an arbitrary, measurable plane set, we mean its measure. When we speak of the *perimeter* of a pseudorectangle we mean its projection-perimeter, which is twice the sum of the lengths of its projections.

The sum of the lengths of the projections of a pseudorectangle of given area $q$ (in our case $q=AB/p$) is a strictly increasing function of the length of the longer of the two projections. (For if the longer projection has length $(\sqrt{q}+h)$ where $h\geqq 0$ then the sum of projections is $(\sqrt{q}+h)+q/(\sqrt{q}+h)$, which is easily shown to be a strictly increasing function of $h$ when $h \geqq 0$.) So if we define the *cost* of a pseudorectangle to be the length of its longer projection, and the *cost* of a decomposition of $R$ into $p$ pseudorectangles to be the cost of the most costly pseudorectangle in the decomposition, then our optimization problems are equivalent to the problems of finding minimal cost decompositions of $R$ into $p$ rectangles and into $p$ pseudorectangles of equal area.

Define a *row*(*column*) of rectangles to be a set of rectangles whose sides are parallel to the coordinate axes, all of which have the same projection onto the $y$-axis ($x$-axis). For any integer $n$ such that $1 \leqq n \leqq p$, define an *n-row decomposition* (*n-column decomposition*) of $R$ to be a decomposition consisting of just $n$ rows (columns) of rectangles, each of which contains either $\lfloor p/n \rfloor$ congruent rectangles or $\lceil p/n \rceil$ congruent rectangles. Figure 1 shows a 4-row decomposition in the case $p=23$, and a 5-column decomposition in the case $p=27$.

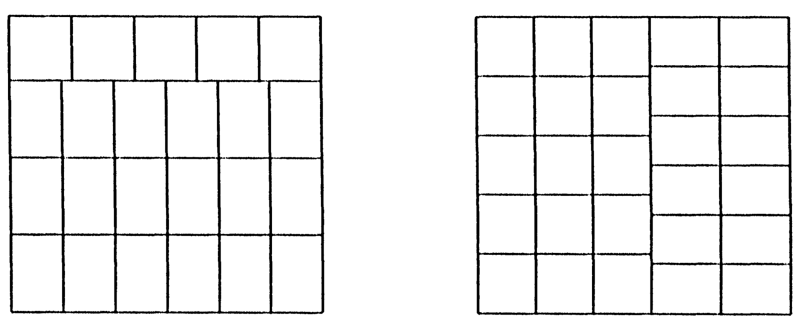

FIG. 1. *A* 4-*row decomposition and a* 5-*column decomposition.*

If $p/n$ is an integer then the $n$-row decomposition is an $n$-by-$(p/n)$ array of congruent rectangles with sides $A/n$ and $Bn/p$. If $p/n$ is not an integer then an $n$-row decomposition has $p-n\lfloor p/n \rfloor$ rows of rectangles, each of which contains exactly $\lceil p/n \rceil$ congruent rectangles, and $n\lceil p/n \rceil - p$ rows of rectangles, each of which contains exactly $\lfloor p/n \rfloor$ congruent rectangles. If we regard two $n$-row decompositions that are related by a permutation of rows to be the same, then for each $1 \leqq n \leqq p$ there is just one $n$-row decomposition of $R$. Thus, we may refer to "the" $n$-row decomposition of $R$. The cost of an $n$-row decomposition is the maximum of the side lengths: $B/\lfloor p/n \rfloor, A\lfloor p/n \rfloor/p, B/\lceil p/n \rceil, A\lceil p/n \rceil/p$. Clearly the maximum will be either the first or last of these values. Analogously the cost of an $n$-column decomposition is $\max(A/\lfloor p/n \rfloor, B\lceil p/n \rceil/p)$.

Intuitively, a decomposition into pseudorectangles of equal area has minimum cost when the pseudorectangles have horizontal and vertical projections that are nearly equal. Ideally, the rectangle would be divided exactly into squares with side lengths $\sqrt{AB/p}$, i.e., into $\sqrt{Ap/B}$ rows and $\sqrt{Bp/A}$ columns. This is possible only when these square roots are integers. However, we will show that one of four decompositions based on the floors and ceilings of these square roots must be a minimal-cost decomposition.

These four decompositions are the $h_1$-row, $h_2$-row, $k_1$-column, and $k_2$-column decompositions of $R$, where $h_1 = \lfloor \sqrt{Ap/B} \rfloor$, $h_2 = \lceil \sqrt{Ap/B} \rceil$, $k_1 = \lfloor \sqrt{Bp/A} \rfloor$, and $k_2 = \lceil \sqrt{Bp/A} \rceil$. (The four quantities $h_1, h_2, k_1, k_2$ all lie between 1 and $p$, because we are assuming that $p > \max(A/B, B/A)$.)

From now on we shall use the term *principal decomposition* to denote an $h_1$-row, $h_2$-row, $k_1$-column, or $k_2$-column decomposition of $R$. The main objective of this section is to prove the following theorem.

THEOREM 2.1. *At least one of the four principal decompositions of $R$ is an optimal decomposition of $R$ into pseudorectangles of equal area.*

This theorem provides us with a simple algorithm for finding an optimal decomposition. The algorithm performs $O(1)$ arithmetic operations. Once it is determined which decomposition is to be used, it is an easy matter to output the boundaries of the rectangles in $O(p)$ time.

*Outline of the Proof of Theorem* 2.1. The key results on which the proof is based are Lemmas 2.3 and 2.4. Each implies a good lower bound on the cost of any decomposition of $R$ into pseudorectangles. The lower bounds are explicitly stated in Lemma 2.5.

Plainly, any decomposition of $R$ that attains one of these two lower bounds must be optimal. This simple observation is used to derive a variety of sufficient conditions on $A$, $B$, and $p$ for one of the four principal decompositions to be an optimal decomposition (Lemmas 2.8, 2.9, and 2.10). We establish Theorem 2.1 by verifying that whatever the values of $A$, $B$, and $p$ are, at least one of these sufficient conditions is sure to be satisfied. □

We begin by dealing with a trivial special case.

LEMMA 2.1. *If $h_1 = h_2$ and $k_1 = k_2$ then the four principal decompositions are the same, and this decomposition is optimal.*

*Proof.* If $h_1 = h_2 = h$ and $k_1 = k_2 = k$ then $h = \sqrt{Ap/B}$ and $k = \sqrt{Bp/A}$, so $hk = p$, and $A/h = \sqrt{AB/p} = B/k$. Thus, all four of the decompositions yield an $h$-by-$k$ array of equal squares. It is clear from our definition of optimality that this decomposition is optimal. □

Our next goal is to derive the lower bounds on the cost of decompositions of $R$. We state these bounds in Lemma 2.5. One of our two bounds follows from a well-known result that is usually attributed to Chebyshev:

LEMMA 2.2 (Chebyshev). *If $x_1 \leqq \cdots \leqq x_n$ and $y_1 \geqq \cdots \geqq y_n$, then the arithmetic mean of the sequence $x_1 y_1, \cdots, x_n y_n$ does not exceed the product of the arithmetic mean of the $x_i$ and the arithmetic mean of the $y_i$.*

*Proof.* For all $i > j$ the product $(x_i - x_j)(y_i - y_j)$ is nonpositive, so the sum of all such products is nonpositive. But this sum is precisely $n^2(W - UV)$, where $U$ is the mean of the $x_i$, $V$ is the mean of the $y_i$, and $W$ is the mean of the $x_i y_i$. □

The next two lemmas imply the lower bounds we seek.

LEMMA 2.3. *Let $h$ and $k$ be positive integers such that $(h-1)(k-1) < p$. Let $\{R_i \mid 1 \leqq i \leqq p\}$ be a collection of pseudorectangles each of area $AB/p$ contained in $R$ whose interiors are pairwise disjoint. Then there is some pseudorectangle $R_i$ whose shortest projection has length at most* $\max(A/h, B/k)$.

*Proof.* Let $a_i$ be the length of the projection of $R_i$ on the $y$-axis, and let $b_i$ be the length of the projection of $R_i$ on the $x$-axis.

If some vertical line meets at least $h$ of the $R_i$ then there is $i$ such that $a_i \leqq A/h$. If some horizontal line meets at least $k$ of the $R_i$ then there is $i$ such that $b_i \leqq B/k$. In either case we are done.

Suppose towards a contradiction, that all horizontal lines meet at most $k-1$ of the $R_i$, and all vertical lines meet at most $h-1$ of the $R_i$. This implies that the sum of the $a_i$ is at most $A(k-1)$ and the sum of the $b_i$ is at most $B(h-1)$. Therefore the product of the mean of the $a_i$ and the mean of the $b_i$ is at most $AB(h-1)(k-1)/(p^2)$, which in turn is less than $AB/p$ by the hypotheses of the lemma.

Without loss of generality, assume that the $a_i$ are arranged in ascending order. Since for each $i$, $a_ib_i = AB/p$, the $b_i$ are in descending order. It follows, by Lemma 2.2, that the mean of the product $a_ib_i$ is at most the product of the means, which we showed to be less than $AB/p$. However, since $a_ib_i = AB/p$, the mean of the products is equal to $AB/p$, a contradiction. □

LEMMA 2.4. *Let $h$ and $k$ be positive integers such that $p<(h+1)(k+1)$. Let $\{R_i \mid 1 \leqq i \leqq p\}$ be a collection of (possibly disconnected) regions of area $AB/p$ whose union is $R$. Then there is some region $R_i$ whose longest projection has length at least* $\min(A/h, B/k)$.

*Proof.* Let $a_i$ be the length of the projection of $R_i$ on the $y$-axis, and let $b_i$ be the length of the projection of $R_i$ on the $x$-axis.

If some vertical line meets at most $h$ of the $R_i$ then there is $i$ such that $a_i \geqq A/h$. If some horizontal line meets at most $k$ of the $R_i$ then there is $i$ such that $b_i \geqq B/k$. In either case we are done. Thus, we may assume that each vertical line passing through $R$ meets at least $h+1$ of the $R_i$, and each horizontal line passing through $R$ meets at least $k+1$ of the $R_i$.

First, we show that there is some $i$ such that

$$a_i + b_i \geqq \min(A/h + Bh/p,\ B/k + Ak/p). \tag{1}$$

Our assumption implies that the sum of the $a_i$ is at least $A(k+1)$, and that the sum of the $b_i$ is at least $B(h+1)$. Thus there is some $i$ such that $a_i + b_i \geqq (A(k+1)+B(h+1))/p$.

Next, we show that $(A(k+1)+B(h+1))/p \geqq \min(A/h+Bh/p,\ B/k+Ak/p)$. Suppose not. Then we derive a contradiction as follows. Since $p<(h+1)(k+1)$ we have

$$p - h(k+1) \leqq k, \tag{2}$$

$$p - k(h+1) \leqq h. \tag{3}$$

It follows from $(A(k+1)+B(h+1))/p < A/h + Bh/p$ and (2) (by routine manipulations) that

$$Bh < A(p-h(k+1)) \leqq Ak.$$

Symmetrically, from $(A(k+1)+B(h+1))/p < B/k + Ak/p$ and (3) we have

$$Ak < B(p-k(h+1)) \leqq Bh,$$

giving the required contradiction. Thus (1) is proved. If $a_i+b_i \geqq A/h + Bh/p$, then since $a_ib_i = AB/p = (A/h)(Bh/p)$, it follows that $\max(a_i, b_i) \geqq \max(A/h, Bh/p) \geqq \min(A/h, B/k)$, as claimed. By symmetry, the same is true if $a_i + b_i \geqq B/k + Ak/p$. □

To state the lower bounds implied by the last two lemmas, define the following values where the variables $h$ and $k$ range over positive integers:

$$C = \min\{\max(A/h, B/k) \mid (h-1)(k-1) < p\},$$

$$S = \max\{\min(A/h, B/k) \mid p < (h+1)(k+1)\}.$$

The reason for the names $C$ and $S$ is that we found it helpful to visualize the "bad" pseudorectangles $R_i$ and $R_j$ in parts (i) and (ii) of the next lemma as a *compressed* and a *stretched* pseudosquare, respectively.

LEMMA 2.5. *In any decomposition of $R$ into pseudorectangles $R_1, \cdots, R_p$ of equal area*:

(i) *There is an $R_i$ whose shortest projection has length at most $C$; hence the cost of the decomposition is at least $AB/(pC)$.*

(ii) *There is an $R_j$ whose longest projection has length at least $S$; hence the cost of the decomposition is at least $S$.*

*Proof.* Assertion (i) follows from Lemma 2.3 and assertion (ii) from Lemma 2.4. □

By Lemma 2.5(ii) any decomposition of $R$ in which the longer projection of every pseudorectangle has length at most $S$ is optimal. Also, since the length of the shorter projection of a pseudorectangle of a given area determines the length of the longer projection, it follows from Lemma 2.5(i) that any decomposition of $R$ in which the shorter projection of every pseudorectangle has length at least $C$ is optimal. (Thus Lemma 2.5 generalizes Propositions 1 and 2 in [15].)

The next step in the proof is to establish a variety of sufficient conditions on $A$, $B$, and $p$ for one of the four principal decompositions to attain one of the lower bounds on cost stated in Lemma 2.5. As it turns out, these sufficient conditions are most conveniently stated in terms of the following four quantities:

$$h_0 = \lfloor p/k_2 \rfloor, \qquad k_0 = \lfloor p/h_2 \rfloor,$$
$$h_3 = \lceil p/k_1 \rceil, \qquad k_3 = \lceil p/h_1 \rceil.$$

Before deriving the sufficient conditions, we prove two useful technical lemmas.

LEMMA 2.6. (i) $h_0 \leqq h_1 \leqq h_2 \leqq h_3$;

(ii) $k_0 \leqq k_1 \leqq k_2 \leqq k_3$;

(iii) *Either* $k_2 = k_3$ *or* $h_0 = h_1$;

(iv) *Either* $h_2 = h_3$ *or* $k_0 = k_1$.

*Proof.* Plainly, $p/k_2 = p/\lceil \sqrt{Bp/A} \rceil \leqq p/\sqrt{Bp/A} = \sqrt{Ap/B}$. Hence $h_0 \leqq h_1$. Similarly, $h_3 \geqq h_2$. So (i) holds, and, by symmetry, so does (ii). Next, observe that if $h_1 k_2 \leqq p$ then $h_1$ is an integer such that $h_1 \leqq p/k_2$, so $h_1 \leqq \lfloor p/k_2 \rfloor = h_0$, whence (i) implies that $h_1 = h_0$. If on the other hand $h_1 k_2 \geqq p$ then by an analogous argument $k_2 = k_3$. So (iii) holds, and by symmetry so does (iv). □

LEMMA 2.7. (i) *If* $h_1 = h_2$ *and* $k_1 < k_2$ *then* $h_0 < h_1 = h_2 < h_3$.

(ii) *If* $k_1 = k_2$ *and* $h_1 < h_2$ *then* $k_0 < k_1 = k_2 < k_3$.

*Proof.* If $h_1 = h_2$ and $k_1 < k_2$ then $\sqrt{Ap/B}$ is an integer but $\sqrt{Bp/A}$ is not, so $h_0 \leqq p/\lceil \sqrt{Bp/A} \rceil < p/\sqrt{Bp/A} = \sqrt{Ap/B} = h_1$, and similarly $h_3 > h_2$. This proves (i), and (ii) follows by symmetry. □

If all the rectangles in a principal decomposition are congruent, then by Lemma 2.5 that decomposition is optimal if the longest (shortest) side of the rectangles has length at most $S$ (at least $C$). This simple observation yields the following sufficient conditions for one of the principal decompositions to be optimal.

LEMMA 2.8. (i) *If* $k_0 = k_1$ *and* $h_2 = h_3$ *then the* $h_2$*-row and* $k_1$*-column decompositions are the same. If, in addition,* $A/h_2 \geqq C$ *or* $B/k_1 \leqq S$ *then this decomposition is optimal.*

(ii) *If* $h_0 = h_1$ *and* $k_2 = k_3$ *then the* $k_2$*-row and* $h_1$*-column decompositions are the same. If, in addition,* $B/k_2 \geqq C$ *or* $A/h_1 \leqq S$ *then this decomposition is optimal.*

*Proof.* Suppose $k_0 = k_1$ and $h_2 = h_3$. The first hypothesis implies $k_1 = \lfloor p/h_2 \rfloor \leqq p/h_2$ and the second implies $h_2 = \lceil p/k_1 \rceil \geqq p/k_1$. Hence $k_1 h_2 = p$, so the $h_2$-row and $k_1$-column decompositions are the same; the decomposition is an $h_2$-by-$k_1$ array of congruent rectangles with sides of length $A/h_2$ and $B/k_1$. A side of length $A/h_2$ is a shortest side, and a side of length $B/k_1$ is a longest side, so if $A/h_2 \geqq C$ or $B/k_1 \leqq S$ then the decomposition is optimal by Lemma 2.5. This proves (i); (ii) follows by a symmetrical argument. □

A principal decomposition usually contains exactly two different kinds of rectangles (see Fig. 1). If it is clear that one kind of rectangle is costlier than the other, and that the longest (shortest) side of the costlier rectangles has length at most $S$ (at least $C$), then by Lemma 2.5 the decomposition is optimal. The following lemma gives sufficient conditions for one of the principal decompositions to be optimal, based on this idea.

LEMMA 2.9. (i) *If* $k_0 < k_1$ *and* $B/k_0 \leqq S$ *then the* $h_2$*-row decomposition is optimal.*

(ii) *If* $k_3 > k_2$ *and* $B/k_3 \geqq C$ *then the* $h_1$*-row decomposition is optimal.*

(iii) *If* $h_0 < h_1$ *and* $A/h_0 \leqq S$ *then the* $k_2$*-column decomposition is optimal.*

(iv) *If* $h_3 > h_2$ *and* $A/h_3 \geqq C$ *then the* $k_1$*-column decomposition is optimal.*

*Proof.* Suppose $k_0 < k_1$ and $B/k_0 \leqq S$. Now each rectangle in an $h_2$-row decomposition either has a side of length $B/\lfloor p/h_2 \rfloor$ or has a side of length $B/\lceil p/h_2 \rceil$. But (since we are assuming $k_0 < k_1$) $\lceil p/h_2 \rceil \leqq k_0 + 1 \leqq k_1 \leqq \sqrt{Bp/A}$, so a side of length $B/\lceil p/h_2 \rceil$ is the longest side of a rectangle of area $AB/p$, and (a fortiori) the same is true of a side of length $B/\lfloor p/h_2 \rfloor$. As $B/k_0 \leqq S$, $B/\lfloor p/h_2 \rfloor \leqq S$, and (a fortiori) $B/\lceil p/h_2 \rceil \leqq S$. So the $h_2$-row decomposition is optimal by Lemma 2.5 (assertion (ii)). This proves part (i) of Lemma 2.9. Part (ii) is proved by an analogous argument by making substitutions that are order-inverting. That is, $k_i$ is replaced by $k_{3-i}$, $h_i$ by $h_{3-i}$, $\leqq$ by $\geqq$, $\lfloor\ \rfloor$ by $\lceil\ \rceil$, $S$ by $C$, and so on. Parts (iii) and (iv) are symmetrical with (i) and (ii). □

If the longest side of one kind of rectangle in a principal decomposition has length at most $S$, while the shortest side of the other kind of rectangle has length at least $C$, then by Lemma 2.5 the decomposition is optimal. Hence we have the following sufficient conditions for optimality.

LEMMA 2.10. (i) *If* $h_0 = h_1$, $A/h_1 \leqq S$, *and* $A/h_2 \geqq C$, *then the* $k_2$*-column decomposition is optimal.*

(ii) *If* $h_3 = h_2$, $A/h_1 \leqq S$, *and* $A/h_2 \geqq C$, *then the* $k_1$*-column decomposition is optimal.*

(iii) *If* $k_0 = k_1$, $B/k_1 \leqq S$, *and* $B/k_2 \geqq C$, *then the* $h_2$*-row decomposition is optimal.*

(iv) *If* $k_3 = k_2$, $B/k_1 \leqq S$, *and* $B/k_2 \geqq C$, *then the* $h_1$*-row decomposition is optimal.*

*Proof.* Suppose $h_0 = h_1$, $A/h_1 \leqq S$, and $A/h_2 \geqq C$. Then $\lfloor p/k_2 \rfloor = h_1$. There are two cases: either $\lceil p/k_2 \rceil = h_1$ or $\lceil p/k_2 \rceil = h_1 + 1$.

In the first case the $k_2$-column decomposition consists of $k_2$ columns, each of which contains $h_1$ congruent rectangles. Each of these rectangles has a side of length $A/h_1$, and (since by definition $h_1 \leqq \sqrt{Ap/B}$) this is the longest side of the rectangles. Hence $A/h_1 \leqq S$ implies that the decomposition is optimal (by assertion (ii) of Lemma 2.5).

In the second case we note that, by Lemma 2.7(i), $h_0 = h_1$ implies that either $h_1 < h_2$ or $k_1 = k_2$. So we may assume that $h_1 < h_2$, for if $k_1 = k_2$ and $h_1 = h_2$ then Lemma 2.10 is certainly true (by Lemma 2.1). By hypothesis $\lceil p/k_2 \rceil = h_1 + 1$, so $\lceil p/k_2 \rceil = h_2$. Now each rectangle in the $k_2$-column decomposition either has a side of length $A/\lfloor p/k_2 \rfloor (= A/h_1)$ or has a side of length $A/\lceil p/k_2 \rceil (= A/h_2)$. Since $h_1 \leqq \sqrt{Ap/B} \leqq h_2$, a side of length $A/h_1$ is a longest side, and a side of length $A/h_2$ is a shortest side. Recalling that $A/h_1 \leqq S$ and $A/h_2 \geqq C$, we see that Lemma 2.10 now follows from Lemma 2.5. This proves assertion (i); the other assertions follow by symmetrical arguments. □

Finally, we need to show that for all values of $A$, $B$, and $p$ at least one of the sufficient conditions for optimality holds. The proof is by case analysis, based on the following lemma.

LEMMA 2.11. (i) *If* $h_1 < h_2$ *then* $\min(A/h_1, B/k_0) \leqq S$ *and* $\max(A/h_2, B/k_3) \geqq C$.

(ii) *If* $k_1 < k_2$ *then* $\min(B/k_1, A/h_0) \leqq S$ *and* $\max(B/k_2, A/h_3) \geqq C$.

*Proof.* Suppose $h_1 < h_2$. Then $k_0 = \lfloor p/(h_1+1) \rfloor$ and $k_3 = \lceil p/(h_2 - 1) \rceil$. Hence, $(h_1 + 1)(k_0 + 1) > p$ implying that $\min(A/h_1, B/k_0) \leqq S$, by definition of $S$. Similarly, $(h_2 - 1) \times (k_3 - 1) < p$; thus $\max(A/h_2, B/k_3) \geqq C$. This proves (i); as usual, (ii) follows by a symmetrical argument. □

*Proof of Theorem* 2.1. If $h_1 = h_2$ and $k_1 = k_2$ then we are home by Lemma 2.1. Suppose $h_1 = h_2$ and $k_1 < k_2$. Then by Lemma 2.7 $h_0 < h_1 = h_2 < h_3$, and so, by Lemma 2.6(iii), $k_2 = k_3$. Hence, we deduce Theorem 2.1 by combining Lemma 2.11(ii) with Lemma 2.10(iv) and Lemma 2.9 ((iii) and (iv)). By symmetry, Theorem 2.1 holds if $h_1 < h_2$ and $k_1 = k_2$.

Now suppose $h_1 < h_2$ and $k_1 < k_2$. On applying Lemma 2.11(i) we see that there are four possibilities:

(a) $A/h_1 \leqq S$ and $B/k_3 \geqq C$.
(b) $B/k_0 \leqq S$ and $B/k_3 \geqq C$.
(c) $A/h_1 \leqq S$ and $A/h_2 \geqq C$.
(d) $B/k_0 \leqq S$ and $A/h_2 \geqq C$.

*Case* (a). Case (a) need not be considered separately as it is symmetrical with Case (d).

*Case* (b). If $k_0 < k_1$ or $k_3 > k_2$ then Theorem 2.1 follows from Lemma 2.9((i) and (ii)). Otherwise $k_0 = k_1$ and $k_3 = k_2$, and Theorem 2.1 follows from Lemma 2.10((iii) or (iv)).

*Case* (c). If $h_0 = h_1$ or $h_2 = h_3$ then Theorem 2.1 follows from Lemma 2.10((i) and (ii)). Otherwise $h_0 < h_1$ and $h_2 < h_3$, and, by Lemma 2.6(iii), $k_2 = k_3$. Now apply Lemma 2.11(ii) to get the following four subcases:

(c1) $B/k_1 \leqq S$ and $A/h_3 \geqq C$.
(c2) $A/h_0 \leqq S$ and $A/h_3 \geqq C$.
(c3) $B/k_1 \leqq S$ and $B/k_2 \geqq C$.
(c4) $A/h_0 \leqq S$ and $B/k_2 \geqq C$.

Since, in the present case, $h_0 < h_1$ and $h_2 < h_3$, Theorem 2.1 follows from Lemma 2.9 ((iii) and (iv)) in Cases (c1), (c2), and (c4). Since, in the present case, $k_2 = k_3$, Theorem 2.1 follows from Lemma 2.10(iv) in case (c3).

*Case* (d). We again apply Lemma 2.11(ii) to get the same four subcases (c1)–(c4) now renamed as Subcases (d1)–(d4).

*Subcase* (d1). Recall that $A/h_2 \geqq C$ and $B/k_0 \leqq S$ (from (d)). Now if $k_0 < k_1$ or $h_2 < h_3$ then Theorem 2.1 follows from Lemma 2.9((i) and (iv)); if, on the other hand, $k_0 = k_1$ and $h_2 = h_3$ then Theorem 2.1 follows from Lemma 2.8(i).

*Subcase* (d2). Symmetric with Case (b) above.

*Subcase* (d3). Symmetric with Case (c) above.

*Subcase* (d4). We have $B/k_2 \geqq C$ and (from (d)) $B/k_0 \leqq S$. Hence, if $k_0 < k_1$ then Theorem 2.1 follows from Lemma 2.9(i), while if $k_0 = k_1$ then Theorem 2.1 follows from Lemma 2.10(iii). □

We have proved that at least one of four principal decompositions is an optimal decomposition of $R$ into pseudorectangles. But we conjecture that an optimal decomposition of $R$ into pseudorectangles is in fact an optimal decomposition of $R$ into arbitrary sets of equal area. In other words, the conjecture is that Theorem 2.1 solves the general decomposition problem as well as the pseudorectangular decomposition problem. We end this section with a simple argument which shows that the conjecture is true if $S \geqq AB/(pC)$.

Observe, first of all, that in the statement of Lemma 2.4 the $R_i$'s need not be pseudorectangles. So we see that a decomposition of $R$ into arbitrary measurable sets of area $AB/p$ must contain a set whose $x$- or $y$-projection has length at least $S$. As was explained in our outline of the proof, we established Theorem 2.1 by showing that at least one of the four principal decompositions attains one of the two lower bounds stated in Lemma 2.5. In other words we showed that (at least) one of the principal decompositions either has cost $S$ or has cost $AB/(pC)$. Now suppose $S \geqq AB/(pC)$. Then by Lemma 2.5(ii) none of the principal decompositions of $R$ can have cost $AB/(pC)$. Therefore one of the principal decompositions has cost $S$, so that the sum of the height and width of any rectangle in that decomposition is at most $S + AB/(pS)$. The fact that a rectangular decomposition of $R$ has cost $S$ also implies that $S \geqq \sqrt{AB/p}$. But we have seen that a decomposition of $R$ into $p$ arbitrary sets of equal area must contain a set whose $x$- or $y$-projection has length at least $S$; the sum of the $x$- and $y$-projections of that set must be at least $S + AB/(pS)$.

**3. Digitizing a rectangular decomposition.** As noted in the Introduction, the lattice decomposition problem, is a discrete analogue to the region decomposition problem. Recall that the problem is to partition an $A \times B$ rectangular array of integer lattice points into $p$ subsets each of size at most $\lceil AB/p \rceil$ such that the maximum projection-perimeter is minimized. We do not know of an efficient solution to the lattice decomposition problem, but when $A$ and $B$ are large relative to $p$ the results of the previous section can be used to find an approximate solution. The problem reduces to approximating the decomposition of an $A \times B$ rectangle on the lattice, so that areas and projection-perimeters are very nearly preserved.

In this section we consider how to compute this approximation, which, to borrow a term from computer graphics and vision, we call *digitization*. We present three digitization algorithms that provide tradeoffs between running time and the quality of the digitization. The second digitization algorithm is quite general, and operates on any decomposition into convex polygons. The other digitizations are significantly more efficient, but operate on a special class of rectangular decompositions which we call *row-major* decompositions. Consider the $A \times B$ rectangle,

$$R = \{(x, y) \mid 0 \leqq x \leqq B, 0 \leqq y \leqq A\}.$$

A decomposition of $R$ into rectangles is called *row-major* if it is of the following form.

(1) $R$ is partitioned into $r$ rows by a set of horizontal line segments running from $x = 0$ to $x = B$. Let $0 = h_0 < h_1 < \cdots < h_r = A$ be the $y$-values of these segments.

(2) Each row is further decomposed by vertical segments into some number of columns. For the row bounded by $h_{i-1}$ and $h_i$ let $0 = v_{i,0} < v_{i,1} < \cdots < v_{i,s_i} = B$ be the $x$-values of these vertical segments.

Note that the decomposition produced by the algorithm of § 2 is either row-major, or can be made so by transposing rows and columns. In this section, we assume that $A$ and $B$ are positive integers. We assume that the line segments defining the decomposition are described using rational numbers representable using $O(\log(p + A + B))$ bits each.

Consider the $A \times B$ rectangular array of integer *lattice points* $L$ superimposed on the rectangle $R$. That is, $L = \{(i, j) \mid 0 \leqq i < A, 0 \leqq j < B\}$. For each $(i, j) \in L$, let $S_{i,j}$ denote the open unit square consisting of the points $(x, y)$ for $i < x < i+1$ and $j < y < j+1$. These are called the *lattice squares*. The rectangle $R$ consists of $A$ horizontal rows and $B$ vertical columns of squares. Our aim is to approximate a decomposition of $R$ into $p$ regions by a partition of the lattice squares into $p$ subsets. Although digitization is common in applications from computer vision and graphics, the goals of our digitization are rather special. We seek a partition of squares satisfying the following criteria: (1) the number of lattice squares assigned to a given region of the decomposition is nearly equal to the area of the region, and (2) the projection-perimeters of a region and its corresponding subset of lattice squares are nearly equal. We formalize these criteria by defining two properties of digitizations that we seek to produce through our algorithms.

DEFINITION. (1) A digitization is *overlapping* if each region is assigned only lattice squares $S_{i,j}$ that overlap the region.

(2) A digitization is *equitable* if the number of lattice squares assigned to a given region does not exceed the ceiling of the area of the region.

All three of the digitization algorithms presented here produce overlapping digitizations; however, the first and third algorithms do not necessarily produce equitable digitizations. Define the *absolute excess* of a digitization to be the maximum signed difference between the number of lattice squares assigned to a region and the true area of the region. (Note that the absolute excess may be negative.) An equitable digitization has an absolute excess less than 1. The *relative excess* is defined to be the maximum ratio of these two values. The first digitization algorithm produces a digitization with relative excess approaching unity as min $(A, B)$ approaches infinity. The second procedure produces an equitable digitization by reducing the digitization problem to the problem of finding a feasible flow in a graph. This algorithm is the least efficient of the three. The third algorithm, is a compromise between these two. Like the first algorithm, it is very efficient with respect to running time but produces a digitization that has an absolute excess less than 2 for all input parameters.

The amount of time required to compute the digitization can be measured as the amount of time required to describe the boundaries of the digitization as a sequence of line segments. Our first and third algorithms can print the boundaries in $O(p)$ time, and hence are optimal with respect to this criterion. Our second algorithm runs in polynomial time in $A+B+p$, although we do not attempt to find the most efficient implementation.

Our first algorithm is a simple naive digitization based on point containment. Stated simply, the set of lattice squares assigned to a region are those whose lower left corners are contained in the region. If a lattice point $(i, j)$ lies on the boundary between two or more regions, then the corresponding square is assigned arbitrarily to one of the regions that overlaps the square. This is easily seen to be an overlapping digitization. It is also easy to see that as $A$ and $B$ increase relative to $p$, then the relative excess of the digitization approaches 1.

Note that the digitization of a given rectangle can be computed in constant time. This digitization algorithm has the property of mapping rectangles to rectangular arrays of squares. The other two digitizations that we present do not have this property.

**3.1. Absolutely equitable digitization.** The second algorithm works by reducing the digitization problem to a graph flow problem. This algorithm can be applied to any decomposition of $R$ into convex polygonal regions. Let $R_1, R_2, \cdots, R_p$ denote a decomposition of $R$ into $p$ polygonal regions.

We define a bipartite flow graph with upper and lower vertex capacities, $G=(V, E, L, U)$, with vertex set $V$, edge set $E$, and lower and upper capacity functions $L$ and $U$. $V$ consists of the following elements:

- *region vertices* $r_1, r_2, \cdots, r_p$, one for each region of the decomposition, and
- *lattice vertices* $s_{i,j}$, one for each of the lattice squares, $S_{i,j}$ for $0 \leqq i < A$ and $0 \leqq j < B$.

The edge set, $E$, consists of the following pairs:

- an edge from each region vertex $r_k$ to each lattice vertex $s_{i,j}$ whenever the unit square $S_{i,j}$ overlaps the region $R_k$,

The vertex capacities are expressed as pairs, $(L(v), U(v))$, representing the lower and upper flow capacity for each vertex $v$. These capacities are:

- $(1, 1)$ for each lattice vertex, and
- $(\lfloor a \rfloor, \lceil a \rceil)$ for each region vertex $r_k$, where $a$ is the area of region $R_k$.

For example, Fig. 2 shows a decomposition of $R$ and the corresponding flow graph.

Let $\Gamma(v)$ denote the neighbors of a vertex $v$. A *feasible flow* is an assignment $f$ of nonnegative integers to each edge of $E$, such that, for each vertex $v \in V$

$$L(v) \leqq \sum_{w \in \Gamma(v)} f(v, w) \leqq U(v).$$

We say that a digitization is *absolutely equitable* if the number of lattice squares assigned to each region is either the floor or ceiling of the area of the region. An absolutely equitable digitization is, in some sense, the most equitable digitization that we can hope to achieve. The connection between the digitization problem and the graph $G$ is established in our next lemma, which follows immediately from our construction.

LEMMA 3.1. *There exists an overlapping, absolutely equitable digitization if and only if the graph $G$ has a feasible flow.*

Although the relationship between the feasible flow problem and the problem of absolutely equitable digitization gives a method of computing the digitization, it is not obvious that such a digitization need exist. Our next result applies a generalization of Hall's well-known theorem on complete matchings in bipartite graphs to show that such a digitization exists for all decompositions.

THEOREM 3.1. *Given any decomposition of $R$ into regions with disjoint interiors, an absolutely equitable digitization of the decomposition exists.*

*Proof.* We make a straightforward adaptation of an existing result from the study of feasible flows in graphs with lower- and upper-edge capacities [10, p. 81], [3, p. 88], which generalizes Hall's theorem on the existence of complete matchings in graphs [4]. This result states that a necessary and sufficient condition for the existence of nonnegative feasible flow in $G$ is that for all subsets $U$ of region vertices and all subsets, $T$, of lattice vertices we have

$$\sum_{v \in U} L(v) \leqq \sum_{w \in \Gamma(U)} U(w), \quad \sum_{w \in T} L(w) \leqq \sum_{v \in \Gamma(T)} U(v). \tag{4}$$

The first half of (4) follows by noting that the sum of areas of a set of disjoint regions $U$ is no greater than the number of unit squares $\Gamma(U)$ that cover the set. The second half follows by noting that the number of disjoint unit squares in a set $T$ is no greater than the sum of areas of the regions $\Gamma(T)$ covering $T$. $\square$

COROLLARY. *If the regions of the decomposition have integer areas, then there exists an overlapping digitization with an absolute excess of zero.*

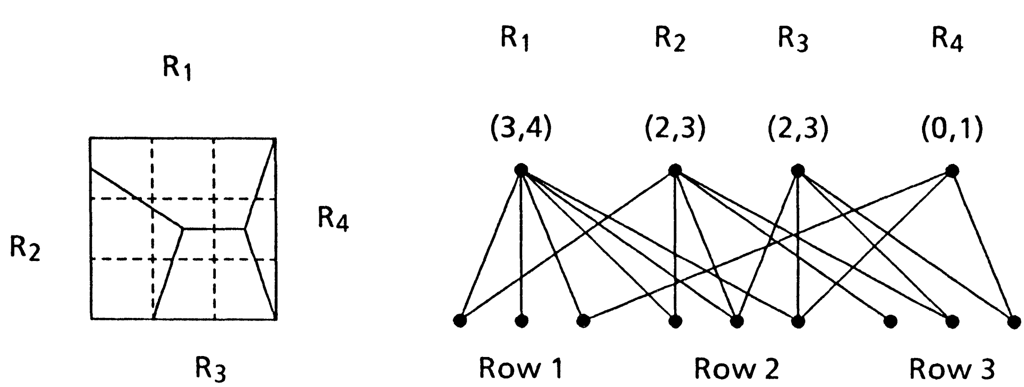


FIG. 2. *The flow graph of a decomposition.*

Once the graph $G$ has been constructed, the equitable digitization can be computed in $O(|V|^3)$ time by any known algorithm for finding feasible flows in graphs [3], [18]. The number of vertices $V$ is $p + AB$ and the magnitude of the capacities is at most $AB$.

In the case where $p$ is small relative to $A$ and $B$ we might hope to find a bound on the size of the vertex set of $G$ that is independent of $A$ and $B$. To do this we reduce the number of lattice vertices as follows. The lattice vertices can be partitioned into equivalence classes according to the set of regions that they overlap, that is according to $\Gamma$. The individual lattice vertices forming each equivalence class can then be replaced by a single aggregate vertex representing the entire class. The capacity of this aggregate vertex is the sum of the capacities of the individual vertices. It can be shown that if $R$ is decomposed into $p$ convex polygonal regions then the number of equivalence classes is $O(p^2)$ [14]. This reduction can be computed easily in $O(pAB)$ time, and the digitization can be computed in $O(p^6)$ time by a feasible flow algorithm.

**3.2. Nearly equitable digitization.** In the previous section we showed that absolutely equitable digitizations exist, although they are somewhat expensive to compute. Next we present an efficient digitization algorithm that does not provide us with an equitable digitization, but does achieve an absolute excess less than 2. Figure 3 shows the result of this digitization of a seven-region decomposition on a lattice of size $13 \times 13$. This digitization has the property that, given a lattice square, the rectangle to which it is assigned can be determined in $O(1)$ time. The boundary of a digitized region can be described by a collection of line segments in $O(1)$ time. This is possible because the algorithm works *locally*, digitizing each rectangle of the decomposition without knowledge of the digitization of any other rectangles, as opposed to the graph flow method which operates *globally*.

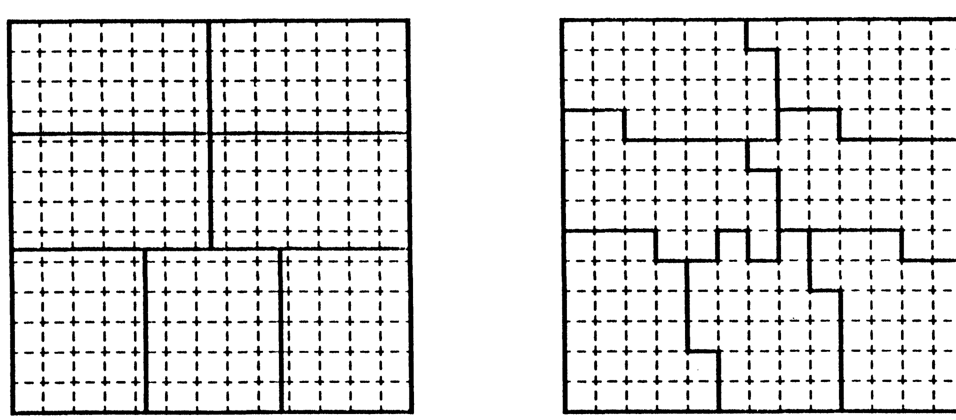

FIG. 3. *A 7-region digitization on a* $13 \times 13$ *lattice.*

This algorithm makes extensive use of the fact that the decomposition is a row-major decomposition. The algorithm operates in two phases. First, for each adjacent pair of horizontal lines $h_i$ and $h_{i-1}$ the digitized region lying between $h_i$ and $h_{i-1}$ is determined. In the second phase, within the digitized region between $h_i$ and $h_{i-1}$, we digitize the region lying between the vertical lines $v_j$ and $v_{j-1}$. In our discussion, we will make use of the following easily verified identity. For all numbers $s$ and $t$:

$$\lfloor s-t \rfloor \leqq \lceil s \rceil - \lceil t \rceil \leqq \lceil s-t \rceil. \tag{5}$$

For the first phase of the algorithm, we show how to digitize the region between the horizontal lines $y=h_i$ and $y=h_{i-1}$, $1 \leqq i \leqq r$. For each $i$ we digitize the region lying below $h_i$, denoted $H_i$, and then define the digitized region between $h_i$ and $h_{i-1}$ to be the set difference $H_i - H_{i-1}$. $H_i$ consists of all lattice squares lying strictly below $h_i$, that is, $S_{x,y}$ where $y < \lfloor h_i \rfloor$, and some subset of the horizontal row of squares $S_{x,y}$, where $y = \lfloor h_i \rfloor$. Consider the set of decomposition vertices lying in this row of squares. For each such vertex $(g, h)$, construct the vertical lines $x = \lfloor g \rfloor$ and $x = \lceil g \rceil$. Let $G_i$ denote the set of $x$-intercepts of these lines, and let $0 = g_{i,0} < g_{i,1} < \cdots < g_{i,m} = B$, the sorted elements of $G_i$. These lines partition this horizontal row of lattice squares into $m$ contiguous blocks. See Fig. 4.

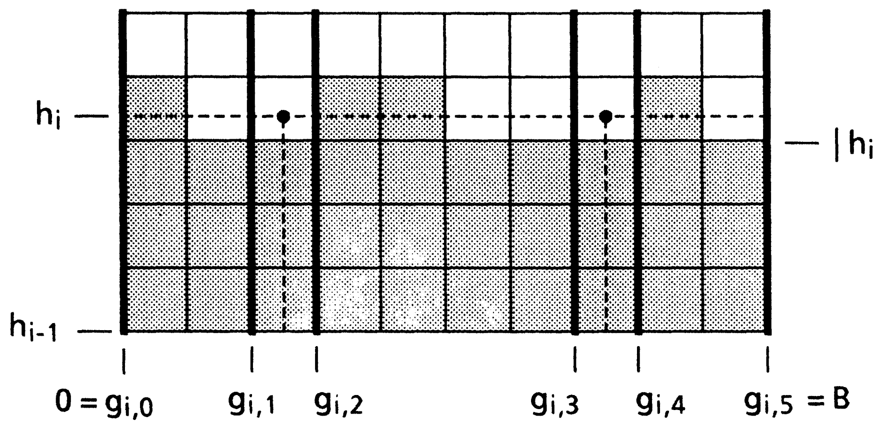


FIG. 4. *Allocation below a horizontal line.*

The portion of this row, bounded above by $h_i$ and lying to the left of $g_{i,j}$ has area $(h_i - \lfloor h_i \rfloor) g_{i,j}$. Let $C_{i,j}$ denote the ceiling of this value. $C_{i,j}$ is the desired total number of squares to allocate to the left of $g_{i,j}$. For the block bounded by $g_{i,j-1}$ and $g_{i,j}$, we allocate the leftmost $C_{i,j} - C_{i,j-1}$ lattice squares to $H_i$. For example, in Fig. 4, $h_i - \lfloor h_i \rfloor = \frac{1}{3}$; hence the number of squares allocated to $H_i$ lying between $g_{i,2} = 3$ and $g_{i,3} = 7$ is $\lceil 7/3 \rceil - \lceil 3/3 \rceil = 2$. The allocated squares are shaded in Fig. 4.

Using (5), we have

$$0 \leqq C_{i,j} - C_{i,j-1} \leqq \lceil (g_{i,j} - g_{i,j-1})(h_i - \lfloor h_i \rfloor) \rceil \leqq g_{i,j} - g_{i,j-1}.$$

Hence, there are enough lattice squares between $g_{i,j-1}$ and $g_{i,j}$ to satisfy this allocation scheme. It is an easy consequence of our definition that $H_{i-1} \subseteq H_i$, for $1 \leqq i \leqq r$. Thus, the digitization of the row between $h_i$ and $h_{i-1}$ can be defined to be $H_i - H_{i-1}$. It is clear by our construction that each square of $H_i - H_{i-1}$ overlaps the region between $h_i$ and $h_{i-1}$. The next result states that the boundary of $H_i$ can be computed in constant time within a fixed block of $G_i$.

CLAIM 3.1. *For a lattice square $S_{x,y}$, where $g_{i,j-1} \leqq x < g_{i,j}$, the membership of $S_{x,y}$ in $H_i$ can be tested in $O(1)$ time. The boundary of $H_i$ between $g_{i,j-1}$ and $g_{i,j}$ can be computed in $O(1)$ time.*

We now describe the second phase of the algorithm in which we complete the digitization of each rectangle by digitizing the vertical lines. Throughout, we will be considering the digitized region $H_i - H_{i-1}$ for any fixed $i$, $1 \leqq i \leqq r$. Let $v_1, v_2, \cdots, v_s$ denote the vertical segments between the horizontal lines $h_i$ and $h_{i-1}$. Similar to the first phase, we determine the digitized region lying to the left of $v_j$, for each $j$, and then define the final digitized region by set difference. The process is slightly more complex than the first phase because of the discontinuities in the boundary of $H_i$ and $H_{i-1}$.

For an integer $g, 0 \leqq g \leqq B$, let $L(g)$ denote the number of lattice squares in $H_i - H_{i-1}$ that lie strictly to the left of the vertical line $x = g$. Intuitively, $L(g)$ is a discrete approximation to the true area $g(h_i - h_{i-1})$. For example, in Fig. 4, the true area bounded by $h_i = 3\frac{1}{3}$, $h_{i-1} = 0$, and $g_{i,3} = 7$ is $23\frac{1}{3}$ and $L(g_{i,3}) = 24$. In the special case that $g$ is the floor or ceiling of a vertical line of the decomposition, then we can bound the value of $L(g)$ as we now show.

CLAIM 3.2. *Let $v_j$ be a vertical line of the decomposition between the horizontal lines $h_i$ and $h_{i-1}$. Then*

(i) $L(\lfloor v \rfloor) \leqq \lceil v(h_i - h_{i-1}) \rceil$, *and*

(ii) $L(\lceil v \rceil) \geqq \lfloor v(h_i - h_{i-1}) \rfloor$.

*Proof.* By definition, both $\lfloor v \rfloor$ and $\lceil v \rceil$ are in $G_i$ and $G_{i-1}$. From the definition of $H_i$ and $H_{i-1}$ it follows that $L(\lfloor v \rfloor) = \lceil \lfloor v \rfloor h_i \rceil - \lceil \lfloor v \rfloor h_{i-1} \rceil$. Part (i) follows by applying (5) and through simple manipulations. Part (ii) follows analogously. □

The digitized region to the left of $v_j$, denoted $V_j$, consists of all the squares $S_{x,y} \in H_i - H_{i-1}$ for which $x < \lfloor v_j \rfloor$ and a portion of the column of squares for which $x = \lfloor v_j \rfloor$. If $v_j$ is not an integer, then the set of squares $S_{x,y}$ with $x \leqq \lfloor v_j \rfloor$ is bounded on the right by the vertical line $x = \lceil v_j \rceil$. If $v_j$ is an integer, then the overlapping constraint implies that we cannot allocate any squares to the right of $v_j = \lceil v_j \rceil$. Thus in either case the maximum number of such squares that can be allocated is $L(\lceil v_j \rceil)$. The digitization seeks to approximate the area bounded horizontally between $h_i$ and $h_{i-1}$ and on the left by $v_j$, that is, $v_j(h_i - h_{i-1})$. Thus, we define the size of $V_j$, denoted $D_j$, by combining these two values:

$$D_j = \min\left(\lceil v_j(h_i - h_{i-1}) \rceil, L(\lceil v_j \rceil)\right).$$

There are $L(\lfloor v_j \rfloor)$ squares allocated strictly to the left of the column $\lfloor v_j \rfloor$, therefore, the number of squares $S_{x,y}$ to be allocated where $x = \lfloor v_j \rfloor$ is $D_j - L(\lfloor v_j \rfloor)$. We select the topmost squares of the column to be allocated. It follows from Claim 3.2(i) that $0 \leqq D_j - L(\lfloor v_j \rfloor)$. It follows from the definition of $D_j$ that there are enough squares in column $\lfloor v_j \rfloor$ to satisfy this allocation. Finally, the digitized region between $v_j$ and $v_{j-1}$ is defined to be $V_j - V_{j-1}$. By definition of $D_j$, this digitization is overlapping.

For example, in Fig. 5, the true area bounded by $h_i = 3\frac{1}{3}$, $h_{i-1} = 0$, and $v_j = 7\frac{1}{2}$ is 25 and $L(\lceil v_j \rceil) = 27$ and so $D_j = 25$. Since $L(\lfloor v_j \rfloor) = 24$, there is one square allocated in column $\lfloor v_j \rfloor$.

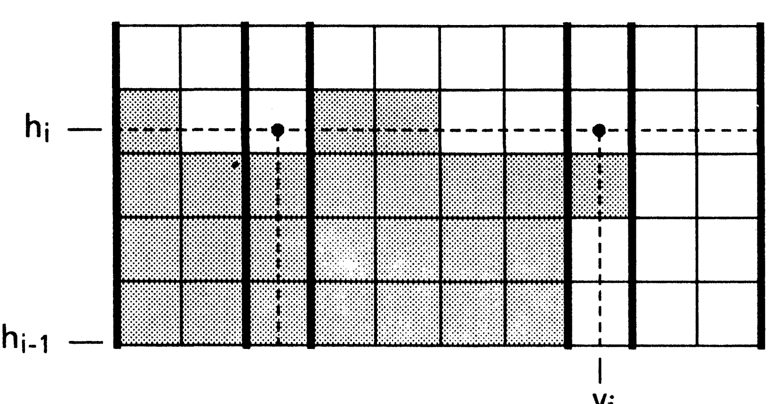


FIG. 5. *Allocation to the left of a vertical line.*

CLAIM 3.3. *The right-side boundary of $V_j$ can be computed in $O(1)$ time.*

*Proof.* The right-hand-side boundary is defined by the squares of column $\lfloor v_j \rfloor$ that are in $V_j$. These squares can be computed in constant time once we know the topmost square of column $\lfloor v_j \rfloor$ in $H_i$. However, since $\lfloor v_j \rfloor \in G_i$, the membership of this square in $H_i$ can be determined in $O(1)$ time by Claim 3.1. □

In summary, to digitize the rectangle bounded by horizontal lines $h_i$ and $h_{i-1}$ and vertical lines $v_j$ and $v_{j-1}$ we apply the first phase of the algorithm to digitize the horizontal region, and then apply the second phase to complete the digitization. We claim that this algorithm defines a digitization that is overlapping and has an absolute excess less than 2.

THEOREM 3.2. *Consider the digitization of each rectangle in a row-major decomposition of R described above.*

(i) *The digitization defines a partition of the set of lattice squares in R.*

(ii) *The digitization is overlapping.*

(iii) *The digitization has an absolute excess less than* 2.

*Proof.* Parts (i) and (ii) follow from the preceding discussion. Details for both cases appear in [14]. To prove (iii), consider a digitized region bounded by vertical segments $v_j$ and $v_{j-1}$ between rows $h_i$ and $h_{i-1}$. The size of the allocation is

$$\begin{aligned} D_j - D_{j-1} &= \min\left(\lceil v_j(h_i - h_{i-1})\rceil, L(\lceil v_j \rceil)\right) - \min\left(\lceil v_{j-1}(h_i - h_{i-1})\rceil, L(\lceil v_{j-1}\rceil)\right) \\ &\leqq \lceil v_j(h_i - h_{i-1})\rceil - \lfloor v_{j-1}(h_i - h_{i-1})\rfloor \quad \text{(by Claim 3.2(ii))} \\ &< \lceil (v_j - v_{j-1})(h_i - h_{i-1})\rceil + 1 \quad \text{(by Equation (5))} \\ &< (v_j - v_{j-1})(h_i - h_{i-1}) + 2. \end{aligned}$$

Thus, the digitization has absolute excess less than 2. □

The fact that the absolute excess may exceed 1 results from the min appearing in the definition of $D_j$. This seems to be an inherent consequence of the constraint that the digitization be overlapping and the locality exploited by the algorithm.

The running time of the algorithm follows from Claims 3.1 and 3.3 together with a few additional observations. By Claim 3.1, we can find the digitization of a rectangle, provided that the number of points in $G_i$ is not too large. We can ignore all the vertices of the decomposition, except for those that appear within the set of unit squares that cover the rectangle, since the remaining vertices cannot affect the digitization here. The vertices to be considered will consist of the four corners of the rectangle, plus any other vertices along the horizontal edges of the rectangle. If the decomposition is generated by the algorithm presented in § 2, the widths of adjacent rectangles differ by at most a factor of $\frac{1}{2}$, from which it follows that the number of such vertices is never greater than 2. From this observation we have

CLAIM 3.4. *When the algorithm is applied to the n-row and n-column decompositions generated by the algorithm of* § 2 *and if* $p \leqq AB$, *then we have the following*:

(i) *The region containing a given lattice square can be determined in $O(1)$ time.*

(ii) *The boundary of a digitized region can be computed in $O(1)$ time.*

**4. Further remarks.** We have given a simple algorithm for decomposing a rectangle into rectangles of equal area whose maximum perimeter is minimized. We have shown that this algorithm is optimal over the more general category of decompositions into pseudorectangles. We have also given an approximate solution to the discrete problem of partitioning grid squares into sets of equal size so that the maximum pseudoperimeter (sum of projections) is minimized.

There are a number of open questions remaining. In § 2, we showed that our decomposition is optimal over all decompositions into pseudorectangles. Is it true that the decomposition is optimal even over all measurable, possibly disconnected, regions? The remark following Lemma 2.5 states that the algorithm is optimal (with respect to pseudoperimeter) for decomposition into arbitrary measurable sets, for certain input values. Also, the question of solving the rectangle decomposition problem in higher dimensions is open.

There is a wide class of similar partitioning problems that are related to the problems considered here. For example, given an $A \times B$ rectangle $R$ and a set of positive real numbers $a_1, a_2, \cdots, a_p$, where $\sum_i a_i = AB$, decompose the rectangles into rectangles of area $a_i$, so that the maximum eccentricity (ratio of a rectangle's longer to shorter side) is minimized. It is also natural to consider alternate cost criteria, such as the sum of perimeters, rather than the maximum perimeter.

**Acknowledgments.** The authors thank Azriel Rosenfeld and Simon Kasif for drawing our attention to the data allocation problem, which was the motivation for this work. Stimulating conversations with Michael Werman are gratefully acknowledged. We would also like to thank the referees for their comments, which significantly improved the presentation.

## REFERENCES

[1] N. ALON AND D. J. KLEITMAN, *Covering a square by small perimeter rectangles*, Discrete Comput. Geom., 1 (1986), pp. 1-7.

[2] D. AVIS AND G. T. TOUSSAINT, *An efficient algorithm for decomposing a polygon into star-shaped polygons*, Pattern Recognition, 13 (1981), pp. 395-398.

[3] C. BERGE, *Graphs and Hypergraphs*, (translated by Edward Minieka), American Elsevier, New York, 1973.

[4] B. BOLLOBÁS, *Graph Theory: An Introductory Course*, Springer-Verlag, Berlin, 1979.

[5] B. CHAZELLE AND D. DOBKIN, *Decomposing a polygon into its convex parts*, in Proc. 11th Annual ACM Symposium on Theory of Computing, Atlanta, GA, 1979, pp. 38-48.

[6] D. S. FRANZBLAU AND D. J. KLEITMAN, *An algorithm for constructing regions with rectangles: independence and minimum generating sets for collections of intervals*, in Proc. 16th Annual ACM Symposium on Theory of Computing, Washington, D.C., 1984, pp. 167-174.

[7] M. R. GAREY, D. S. JOHNSON, F. P. PREPARATA, AND R. E. TARJAN, *Triangulating a simple polygon*, Inform. Process Lett., 7 (1978), pp. 175-179.

[8] M. R. GAREY AND D. S. JOHNSON, *Computers and Intractability: A Guide to the Theory of* NP-*Completeness*, W. H. Freeman, New York, 1979.

[9] T. GONZALEZ AND S-Q. ZHENG, *Bounds for partitioning rectilinear polygons*, in Proc. Symposium on Computational Geometry, 1985, pp. 281-287.

[10] C. BERGE, *The Theory of Graphs and its Applications*, Translated by A. Doig, John Wiley, New York, 1962. (In French.)

[11] M. HOFRI, *Two-dimensional packing: expected performance of simple level algorithms*, Inform. Control, 45 (1980), pp. 1-17.

[12] R. KARP, M. LUBY, AND A. MARCHETTI-SPACCAMELA, *Probabilistic analysis of multidimensional bin packing problems*, in Proc. 16th Annual ACM Symposium on Theory of Computing, 1984, pp. 289-298.

[13] S. KASIF AND R. KLETTE, *A data allocation problem for* SIMD *computers*, Tech. Report CAR-TR-11 Center for Automation Research, University of Maryland, College Park, MD, 1983.

[14] T. Y. KONG, D. M. MOUNT, AND A. W. ROSCOE, *The decomposition of a rectangle into rectangles of minimal perimeter*, Tech. Report CAR-TR-169, Center for Automation Research, College Park, MD, 1986.

[15] T. Y. KONG, D. M. MOUNT, AND M. WERMAN, *The decomposition of a square into rectangles of minimal perimeter*, Discrete Appl. Math., 16 (1987), pp. 239-243.

[16] S. R. LAY, *Convex Sets and their Applications*, John Wiley, New York, 1982.

[17] J. O'ROURKE, *The complexity of computing minimum convex convers for polygons*, in Proc. 20th Annual Allerton Conference on Communications, Control and Computing, Urbana, IL, 1982, pp. 75-84.

[18] R. E. TARJAN, *Data Structures and Network Algorithms*, CBMS-NSF Regional Conference Series in Applied Mathematics 44, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1983.