Finding a Minimal Cover for Binary Images: an Optimal Parallel Algorithm *

Dipen Moitra

88-946

November 1988

Department of Computer Science
Cornell University
Ithaca, New York 14853

*Research partially supported by NSF grant DCI-86-02256, ONR contract N00014-83-K-0640, and IBM agreement 12060043

Finding a Minimal Cover for Binary Images: an Optimal Parallel Algorithm*

Dipen Moitra

Computer Science Department, Cornell University,
5134 Upson Hall, Ithaca, NY 14853.
email: dipen @ svax . cs . cornell . edu

Abstract

Given a black and white image, represented by an array of $\sqrt{n} \times \sqrt{n}$ binary valued pixels, we wish to cover the black pixels with a *minimal* set of (possibly overlapping) maximal squares. It was recently shown that obtaining a *minimum* cover with squares for a polygonal binary image having holes is NP-hard. We derive an optimal parallel algorithm for the *minimal square cover* problem, which for any desired computation time T in [log n, n] runs on an EREW-PRAM with (n / T) processors. The cornerstone of our algorithm is a novel data structure, the cover graph, which compactly represents the covering relationships between the maximal squares of the image. The size of the cover graph is linear in the number of pixels. This algorithm has applications to problems in VLSI mask generation, incremental update of raster displays, and image compression.

Key words: Parallel Algorithms, Computational Geometry, Image Compression, Minimal Square Covers, Orthogonal Polygons, Parallel Prefix Computations, Minimal Vertex Covers, Bipartite Graphs.

CR subject categories: **F.2.2** [Geometric Computations], **G.2.2** [Graph Algorithms], **I.4.2** [Image Compression, Exact Coding]

* Research supported in part by NSF grant DCI-86-02256, ONR contract N00014-83-K-0640, and IBM agreement 12060043.

A preliminary version of this paper was presented at the 26th Ann. Allerton Conference on Communications, Control and Computing, Monticello IL, Sept. 28-30 1988.

1 Introduction

In many applications it is important to encode digitized images compactly, in order to reduce storage and transmission costs, even at the expense of time and hardware for encoding and decoding the raw image. A compact encoding of digitized images can also provide the basis for efficient image analysis and manipulation algorithms, for applications such as remote sensing, VLSI mask generation, incremental update of raster displays, and object representation for sequential frames of a dynamic polygonal scene.

A popular technique for image representation is the *Quad-tree* [TaP-75], which provides a recursive decomposition of a binary image into quadrants, each quadrant being decomposed further only if it is neither all white nor all black. This yields a 4-ary tree with at most log n levels, for a digitized image consisting of $\sqrt{n} \times \sqrt{n}$ pixels. Although this supports efficient algorithms for image manipulation, the size of the resulting tree is extremely sensitive to the placement of the *origin* for the decomposition at the root. For example consider the decomposition of a single square of side $\sqrt{n}$, starting at (1, 1); it contains $\Omega(\sqrt{n})$ nodes, instead of the single node which suffices for a minimal cover. There have been many attempts to find an optimal placement for the *origin*, but all the techniques are computationally expensive, and do not yield guaranteed size reductions [Sam-84]. Scott and Iyengar [ScI-86] proposed that instead of finding an optimal origin for the Quad-tree, we could represent a binary image more compactly by means of a suitable small subset of the maximal subsquares of the image.

a. Given black and white image

b. Cover with Maximal Squares

Figure 1. Minimal Cover of Squares

Given a black and white image represented by an array of $\sqrt{n} \times \sqrt{n}$ binary valued pixels, we wish to cover the image with a minimal set of black subsquares of the image, and permit overlaps amongst the squares in the cover (see Figure 1). Clearly we lose nothing by insisting that the chosen squares be maximal; in fact the efficiency of the algorithm depends crucially upon the maximality of the squares. In the rest of this paper we will discuss parallel algorithms for obtaining a minimal cover consisting of maximal squares, for digitized black and white images. Our algorithm is optimal with respect to both its running time, as well as the number of

operations performed. It has a linear processor-time product, and for all desired computation times T in [log n, n] runs on an EREW-PRAM with (n / T) processors.

Definitions *(maximal square, minimal cover)* A *maximal square* is a black subsquare of the image, not contained in any larger black square. A *cover* for a given image is a set C of maximal squares, which can overlap with each other, and whose union equals the image. If no subset of C is a cover, C is called a *minimal cover*.

1.1 Comparison with Related Work

Scott and Iyengar [ScI-86] present an algorithm to find the maximal squares in a digitized image, and then divide the rectangular portions of the image to cover them minimally. Their algorithm does not yield a minimal cover over the entire image, is sequential, and requires O(n log n) time. It was recently shown by Aupperle et al. [ACK-88] that obtaining a minimum cover for a non-simply connected polygonal binary image (with holes), using (maximal) squares is NP-hard. In the same paper a sequential algorithm for obtaining a minimum cover for simply connected images (without holes) was also presented. Given an image I, they construct a cover graph G(I) with one vertex for every pixel of the image, and add an edge between every pair of pixels which belong to some common maximal square. We note that the maximal cliques in G(I) correspond to the maximal squares in I. They show that if the image has no holes, the resulting cover graph G(I) is *chordal*, and hence one can use the well known algorithm of Gavril [Gav-72] to find a *minimum clique cover* for G(I). The above construction yields a graph with $\Omega(n^2)$ edges in the worst case (consider an image consisting of a single square of side $\sqrt{n}$). They reduce the required number of operations to $O(n\sqrt{n})$ by avoiding the explicit construction of G(I), and rely instead on processing the maximal cliques of G(I) [Aup-88]. Although their algorithm produces a minimum cover for an image without holes, if the image has any holes, then no technique is presented for finding any cover (even irreducible). In fact, the problem of finding a *minimum cover* for general graphs is NP-hard, even if the graph is planar. Not only is our algorithm optimal and parallel, but it also works for images with holes. Of course our algorithm always yields a minimal cover, which is not necessarily the smallest cover.

2 Overview of the Minimal Cover Algorithm

In this section we will present an overview of our minimal cover algorithm. First let us see what any algorithm which obtains a minimal square cover must do. The obvious first step is to find all the maximal squares in the image. Let M be the set of all maximal squares in the given image I. The process of obtaining an irreducible cover for I consists of refining a tri-partition of M, of the form $\langle R, D, A \rangle$, where R is the set of *retained* squares, D is the set of *discarded* squares, and

$A=(M-(R \cup D))$ is the set of *available* squares for which we have not yet decided anything. Initially D is empty, and R contains only the *essential squares* of I (i.e. those maximal squares which contain at least one pixel not covered by any other maximal square). At the end of the algorithm, $M=(R \cup D)$.

During every *phase* the partition of M is refined closer to termination, by selecting a subset S of A such that all the squares in S can be discarded simultaneously, without violating the *safety* condition that $(A \cup R)$ covers I. Consequently, for every square $\mathbf{m}$ in S, we must be in a position to answer the question: "*does* $((A \cup R)-\{S\})$ *cover* $\mathbf{m}$?" For this purpose, we construct a *cover graph* G(I), which provides a compact representation of the covering relationships between the maximal squares in the image I. This will let us efficiently determine the relative redundancy of an arbitrary square $\mathbf{m}$ with respect to an arbitrary set of residual squares $((A \cup R)-\{S\})$. Clearly the number of phases of the algorithm (and hence its maximum parallelism) depends crucially on the number of squares which can be simultaneously discarded during each phase. The cover graph helps us to determine a large independent set of squares, which can be discarded simultaneously without violating the safety of the cover.

2.1 The Cover Graph, and Reducing its Size

As the first step of our algorithm we determine all the maximal squares in the image. Then we examine the geometric covering relationships between the maximal squares, and represent them compactly by means of a bipartite cover graph $G=\langle (M, P), E\rangle$; whose vertex sets M and P denote respectively the maximal squares, and the pixels in the image. An edge $\langle \mathbf{m}, \mathbf{p} \rangle$ in E denotes that square $\mathbf{m}$ contains pixel $\mathbf{p}$. After this step we restrict our attention to just the cover graph, and find a minimal subset M_c of M, such that every pixel in P is contained in at least one square in M_c.

Consider the images shown in Figure 2, which show respectively a horizontal and a diagonal sequence of identically sized maximal squares, spaced one pixel apart. One can easily verify that if we construct the cover graph (naively) defined above, it will have $\Omega(n \sqrt{n})$ edges in the worst case, and we will not be able to meet our overall linear resource bounds. Instead, we can apply a series of conceptual refinements to the naive cover graph, and obtain one whose size is linear in the number of pixels in the image. Of course, we do not construct the naive cover graph and then reduce it; instead we construct only the reduced cover graph as we go along.

(i) **Composite regions** : Replace all the pixels which are covered by the same set S of squares, by a single region R which equals the union of those pixels. Delete all the edges incident on the deleted pixels, and replace them with edges between R and each square in S.

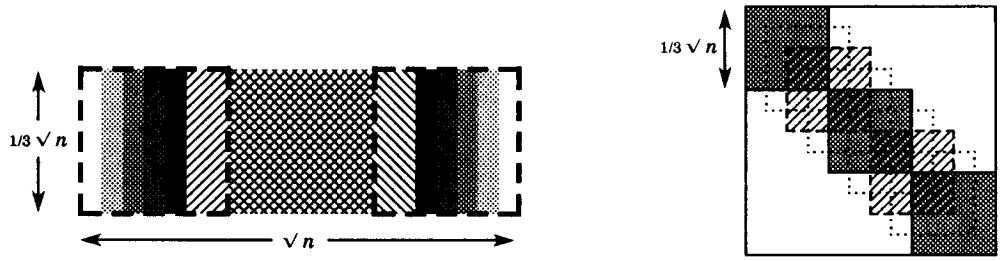


Figure 2. Example images for worst case size of unoptimized Cover Graph

(ii) **Prime implicants** : If pixels p and q of square m are covered respectively by two different sets of squares S_p and S_q such that S_p is a strict subset of S_q, then merge the regions containing p and q into a single region. The resulting region will clearly equal the intersection of all the squares in S_p, and be rectangular. Collapse the corresponding vertices and edges as before.

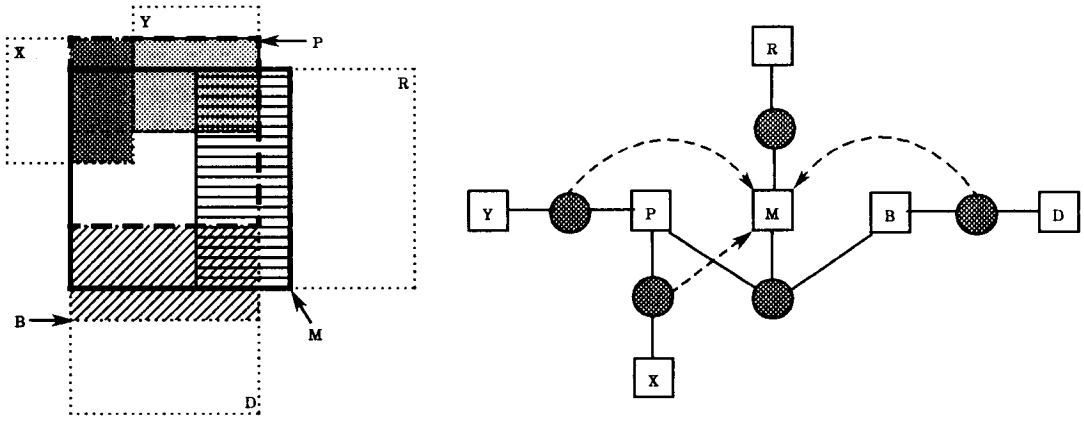
(iii) **Global implication** : If a pair of squares S and T are *pairwise essential* (i.e. some pixel in S∩T is not covered by any other square except S and T), then merge all the rectangles which are covered by strict subsets of S∩T into the rectangle S∩T. Again collapse the corresponding vertices and edges. This refinement is the one which makes the cover graph directed, in the sense that some rectangle r may be part of the cover of square m, although m may not contain r.

The resulting *reduced cover graph* $G \doteq \langle (M, R), E \rangle$ is directed and bipartite, with vertex sets M (maximal squares) and R (rectangles). The rectangles in R are derived by a suitable decomposition of the squares in M, as given in Section 6. The edge set E is a subset of $(M \times R) \cup (R \times M)$, and denotes containment between the squares and rectangles as follows. (i) The union of all the squares in M spans the entire image, (ii) a given square is contained in the union of all of its predecessor rectangles, and (iii) a given rectangle is contained in every one of its predecessor squares. We show in Figure 3 below a small collection of squares, and part of its cover graph, with the rectangles denoted by circles. For simplicity we show a pair of oppositely directed edges between a single pair of vertices, as a single undirected edge between them.

2.2 Getting a Minimal Cover from the Cover Graph

Given any cover graph G(I) which satisfies the three conditions listed above, a minimal cover for the image I can be found by finding a *one-sided minimal vertex cover* for G, which can be defined as follows. A *one-sided minimal vertex cover* for $G \doteq \langle (M, R), E \rangle$ is a subset M_C of the maximal squares M, such that (i) (**safety**): every rectangle r in R has at least one predecessor in

M_C, and (ii) (**minimality**): every square **m** in M_C has a successor **r** (called its *witness*), such that **r** has no predecessor in $(M_C - \{m\})$.



a. Given collection of squares b. Part of its Reduced Cover Graph

Figure 3. Reduced Cover Graph

We obtain a minimal cover for G by coloring G in *phases* as described in Section 7. During each phase we extract non-intersecting sets of maximal paths, consisting of *chains* of undiscarded vertices and edges of G. Every alternate vertex in each such chain is retained; and of the remaining vertices, the ones that are covered by the retained vertices are discarded. We show in Section 6 that G has a constant degree. Each such phase reduces the degree of undiscarded squares by at least two, hence a constant number of phases of coloring suffices to color all of G.

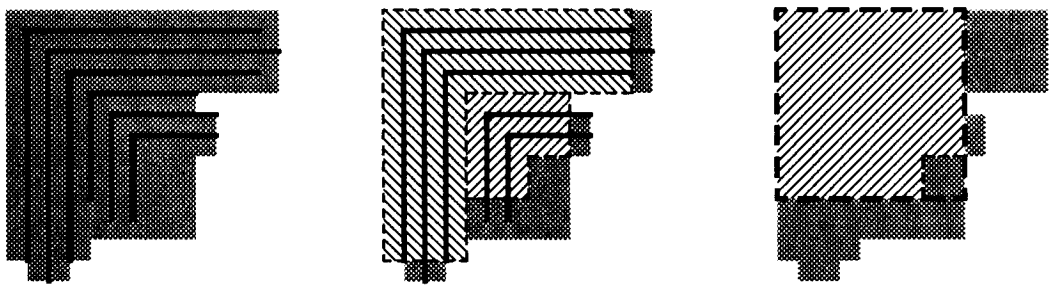
2.3 Algorithm Minimal_Cover

1. Determine the maximal squares, using the algorithm of Section 3.
2. Identify rectangular regions, each of which is covered by a contiguous horizontal or vertical sequence of identically sized maximal squares, and retain a minimum cover for each rectangle, using the algorithm of Section 4.
3. For every retained maximal square, determine if it is covered by the union of its neighboring maximal squares, as in Section 5.2. If no such cover exists, then mark it as being *essential.*
4. For every non-essential square **m**, derive its *extended cover* Ξ_m, which consists of all the retained squares which are "useful" for covering **m**, as given in Section 5.4.
5. Determine the reduced cover graph G(I) for the image I, as given in Section 6
6. Find a one-sided minimal vertex cover for G(I), using the coloring algorithm of Section 7.

3 Identifying Maximal Squares

The first step of our algorithm is to find the maximal squares in the image. As an initial step towards finding the maximal squares, we compute something called *largest black squares*, which are the lower right (diagonal) square suffixes of maximal squares. Let us define the *largest black square* ${}^{SE}L_{ij}$ to be the largest square of black pixels, with its top left corner (*origin*) at ⟨i, j⟩ and its diagonally opposite corner to the *SouthEast* of ⟨i, j⟩. We also define and compute ${}^{SW}L_{ij}$, ${}^{NW}L_{ij}$, and ${}^{NE}L_{ij}$ in a similar manner. This initial step not only helps us to find the maximal squares efficiently, but also generates information which will be useful during the elimination of redundant maximal squares. In the sequel, L_{ij} will mean ${}^{SE}L_{ij}$.

We describe below an optimal parallel algorithm, which uses *parallel prefix computations* [KRS-85, CoV-88a], to find ${}^{SE}L_{ij}$ for every pixel ⟨i, j⟩. Given a semigroup ⟨S, ⊗⟩ (where ⊗ is a binary associative operation on S), a *prefix computation* over a sequence $s_1 \ldots s_n$ of elements of S, consists of computing the n partial products $s_1 \otimes \ldots \otimes s_k$ for all $k \leq n$. A *prefix computation* over n elements can be performed optimally in parallel, in time $T \in [\log n, n]$, using an n/T processor EREW-PRAM, and in time $T \in [\log n / \log\log n, \log n)$, using an n/T processor CRCW-PRAM [CoV-88a].



a. Sequence of L-borders b. Two Arrowhead regions c. Composite Arrowhead

Figure 4. Algorithm for finding *Largest Black Squares*

We find the largest black squares in three steps. (i) For every black pixel, find R_i, the right end of the largest horizontal black strip containing it; and also L_i, the lower end of the largest vertical black strip containing it. (ii) Hence determine the largest L-shaped region which could potentially form the upper-left border of a black square to the right and below, as the minimum of the distances to the two ends. (iii) Finally, for every pixel, determine the largest square contained in the diagonal sequence of L-shaped black borders to the right and below, by combining the L-shaped regions. In subsequent paragraphs we elaborate on these three steps. (See Figure 4.)

Figure 5. Associative combination of *Hammerhead* regions

First we address the problem of finding the longest horizontal / vertical black strips. Define the *hammerhead* of a sequence of pixels $x_i \ldots x_{i+l_i-1}$ as $B_i \doteq \langle d_i, l_i \rangle$; where d_i (the depth of the hammerhead) is the maximum number of ones which prefix the sequence $x_i \ldots x_{i+l_i-1}$, and l_i is the length of the sequence. Clearly we can combine the hammerheads $\langle d_1, l_1 \rangle$, $\langle d_2, l_2 \rangle$ of two adjacent subsequences $x_i \ldots x_{i+l_1-1}$ and $x_{i+l_1} \ldots x_{i+l_1+l_2-1}$ into a single hammerhead $\langle d, (l_1+l_2) \rangle$; where the depth d of the composite hammerhead is given by $d \leftarrow$ **if** $d_1 < l_1$ **then** d_1 **else** $d_1 + d_2$ **fi**. A little reflection will show that this computation is associative, and the desired result R_i equals *depth*(B_i). (See Figure 5.)

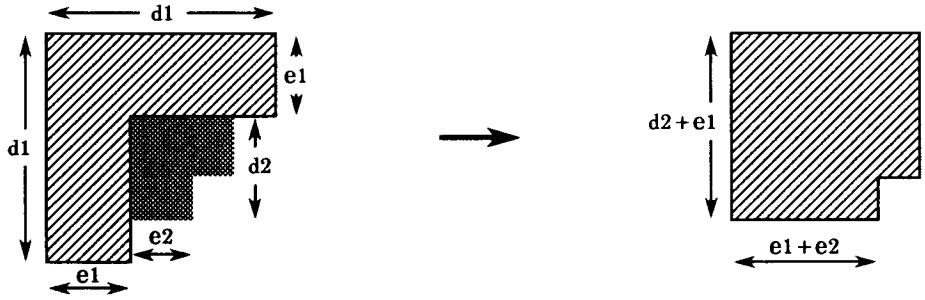


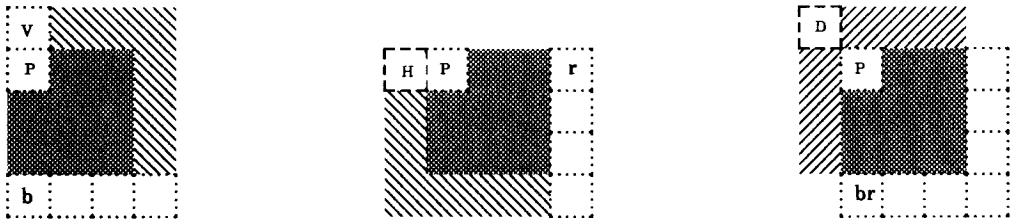
Figure 6. Associative combination of *Arrowhead* regions

Step (iii) can be similarly transformed into a prefix computation as follows. Define an *arrowhead* as an L-shaped region containing only black pixels, whose two arms have equal lengths and equal widths. An arrowhead with its *origin* (top left corner) at $\langle i, j \rangle$ is represented by the tuple $A_{ij} \doteq \langle d, e \rangle$, and denotes that pixel $\langle i, j \rangle$ can potentially be the origin of a black square of size atmost d, and that we have verified the top e rows and left e columns of such a square. We can combine two adjacent arrowheads $\langle d1, e1 \rangle$ and $\langle d2, e2 \rangle$, into a single arrowhead $\langle \min(d1, d2+e1), (e1+e2) \rangle$. This operation is also associative, and can be computed along the diagonals of the image, using a parallel prefix computation. When the thickness of an arrowhead equals its depth, we have found a largest black square. The desired result L_{ij} equals *depth*(A_{ij}). (See Figure 6.)

3.1 Largest Black Squares which are Maximal

As shown earlier, *largest black squares* are the lower right square suffixes of maximal squares, and are bounded by white pixels to the right and/or below. Clearly a largest black square L is maximal *iff* it is not contained in a larger black square M, and the *origin* (upper left corner) of such an M can only be above and/or to the left of the origin of L. Further, by the suffix property of largest black squares, the existence of a larger black square above (to the left of) the origin of L, implies the existence of one *immediately above* (*immediately to the left of*) the origin of L. We can now identify the maximal squares with only a constant number of computations per pixel.

Lemma 1 Let P be the largest black square with its *origin* at $\langle i, j \rangle$, and $l_{ij} \geq 1$ be its size. Then P is not *maximal* iff $l_{i-\delta i, j-\delta j} > l_{ij}$, for some $\langle \delta i, \delta j \rangle \in \{\langle 0, 1 \rangle, \langle 1, 0 \rangle, \langle 1, 1 \rangle\}$.



a. Bounded below b. Bounded to the right c. Bounded below and right

Figure 7. Largest Black Square which is not maximal

proof (Refer to Figure 7.)

Let B(r) be the predicate *"every pixel in the region* r *is black"*. Let V, H, D, X be the four squares of size $(l_{ij}+1)$, with their top-left corners respectively at $\langle i-1, j \rangle$, $\langle i, j-1 \rangle$, $\langle i-1, j-1 \rangle$, and $\langle i, j \rangle$. Clearly P is contained in each of V, H and D. Hence if any of V, H or D is all black then P is not *maximal*. Further H, V, D and X are the only 4 squares of size $= l_{ij}+1$ that contain P. By hypothesis B(X) is false. Hence P is not *maximal* implies B(V) $\vee$ B(H) $\vee$ B(D); which is equivalent to saying that $l_{i-\delta i, j-\delta j} > l_{ij}$, for some $\langle \delta i, \delta j \rangle \in \{\langle 0, 1 \rangle, \langle 1, 0 \rangle, \langle 1, 1 \rangle\}$. □

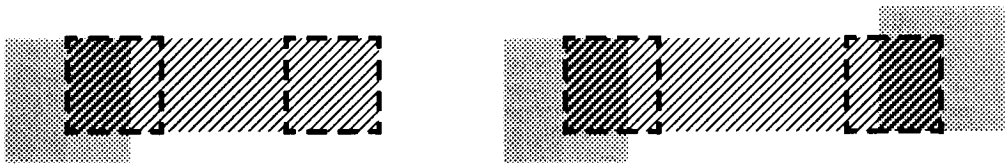
4 Dividing Uniform Strips

Having obtained the maximal squares, we identify rectangular regions spanned by sequences of identically sized maximal squares, and retain a minimum cover for each. Without this step, the cover graph can have $\Omega(n\sqrt{n})$ edges, and / or a maximum degree of $\Omega(\sqrt{n})$ in the worst case. In the sequel we will only be concerned with identifying all possible irreducible covers for non-*essential* squares, consisting only of *retained* squares (not discarded during the division of uniform strips).

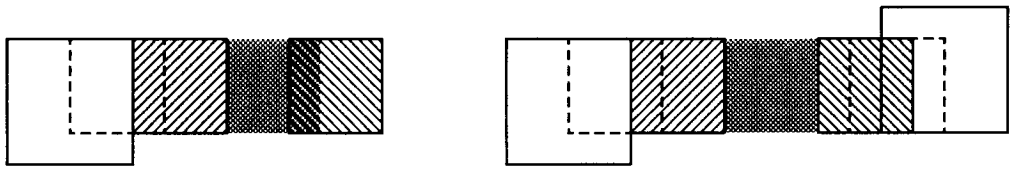
Definition (*uniform strip*) : A horizontal (vertical) *uniform strip* is a maximal sequence of identically sized maximal squares, whose *origins* are horizontally (vertically) contiguous.

Note that this definition of a uniform strip does not imply that all the pixels surrounding a uniform strip are white. We retain a minimum cover for a uniform strip, by dividing its longer side by its width in the following manner. For every maximal square m, determine the leftmost (uppermost) corner (*origin*) of the *uniform strip* U which contains it (as shown below). If the *origin* of the uniform strip is inside a maximal square h which is larger than the width of the strip, redefine the *origin* to be shifted just past the right (bottom) edge of h. Retain the leftmost (uppermost) maximal square in U, and all those maximal squares in U whose *origin* is displaced from the origin of U by an integral multiple of the width ω of the strip. In addition to the squares

retained by the previous condition, retain the rightmost (lowermost) square in U, unless its last $(\omega - 1)$ columns (rows) are covered by a larger maximal square. Discard all the others. (Figure 8.)



a. Uniform Strip, with black regions surrounding first and / or last squares



b. Squares retained after division

Figure 8. Dividing *Uniform Strips*

Let M_{ij} be the maximal square with its origin at $\langle i, j \rangle$. In order to find the left corner of (say) a horizontal uniform strip, we must compute the length of the uniform strip to the left of $\langle i, j \rangle$, as $H_{ij} \doteq \min \{k \mid 0 < k \leq j \text{ such that } |M_{i,j-k}| \neq |M_{ij}| \}$. This can be transformed to a *prefix computation* by a technique similar to that of Section 3. We define a *uniform_tail* U_{ij} to be the right end of a uniform strip which has its *origin* at $\langle i, j-k+1 \rangle$, and represent it as the triple $\langle m, d, l \rangle$; where **m** is the size of the rightmost maximal square in the tail, **d** (depth) is the length of the uniform strip at the end of the tail, and l is the length of the tail. We can associatively combine two adjacent *uniform_tails* into a single tail, as $\langle m_2, d_2, l_2 \rangle \otimes \langle m_1, d_1, l_1 \rangle \doteq \langle m_1, d, (l_1 + l_2) \rangle$; where the depth d of the composite strip is given by $d \leftarrow$ **if** $(m_2 \neq m_1)$ **or** $(d_1 < l_1)$ **then** d_1 **else** $d_1 + d_2$ **fi**. The required computation is defined recursively as $U_{ij} \leftarrow U_{i,j-1} \otimes \langle m_{ij}, m_{ij}, m_{ij} \rangle$. Observe that this computation proceeds from right to left, starting at the *origin* of each maximal square. H_{ij} equals *depth*(U_{ij}). (See Figure 9.)

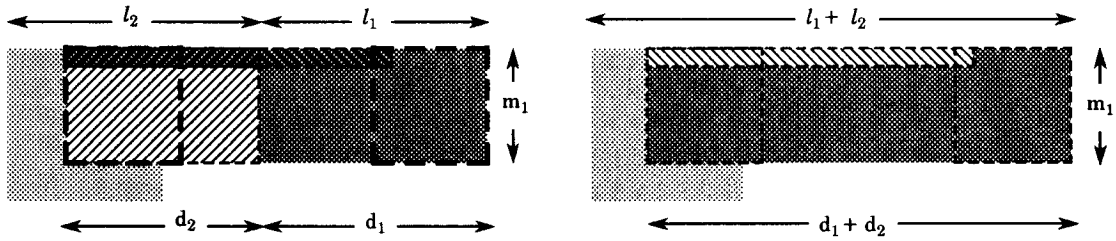


Figure 9. Combining adjacent substrips of *Uniform Strips*

5 Conditions for the Redundancy of Maximal Squares

In this section we will determine whether or not any given maximal square **m** is essential; and if it is not essential, then we will find all distinct sets of maximal squares which cover **m**. A naive approach might be to first determine all the squares which overlap with **m**. In the example

shown in Figure 10 below, $\Omega(\sqrt{n})$ maximal squares $T_1 \ldots T_k$ overlap with the given maximal square **m**. However, for every set S_i of maximal squares containing T_i such that S_i covers **m**, we observe that $(S_i - T_i)$ also covers **m**. Hence none of the squares $T_1 \ldots T_k$ are useful for determining the redundancy of **m** relative to any given set of squares. If we restrict our attention initially to those squares which overlap maximally with **m**, we can determine efficiently if **m** is essential, else identify a cover for it, as shown in the following sections. This motivates us to define the *basic cover* as given below. In Section 5.3 we define the *extended cover* of **m**, which characterizes all the squares which are "useful" for covering **m**. In Section 5.4 we show how to determine the extended cover of **m**, by tracing a sequence of basic cover relationships between the squares bordering **m**.

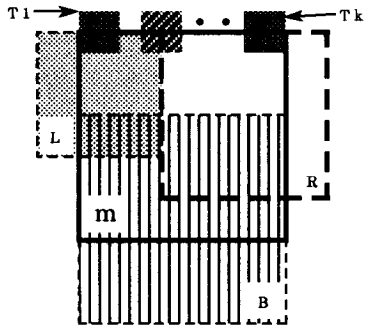


Figure 10. Non-triviality of identifying a cover for **m**

Definition (*basic cover*) : The *basic cover* β_m for an inessential square **m**, is a set of maximal squares excluding **m**, whose union covers **m**; such that for all $S \in \beta_m$, $S \cap \mathbf{m}$ is not strictly contained in $T \cap \mathbf{m}$ for any $T \notin \{\mathbf{m}, S\}$.

5.1 Basic Cover Relations

From the definition of a basic cover it follows that a square is not *essential* (i.e. potentially redundant) *iff* it has a basic cover. We should note that the definition of a basic cover implies that it is unique, though not necessarily irreducible. In the following lemma we prove that if a maximal square M is not essential, its *basic cover* has four or fewer maximal squares.

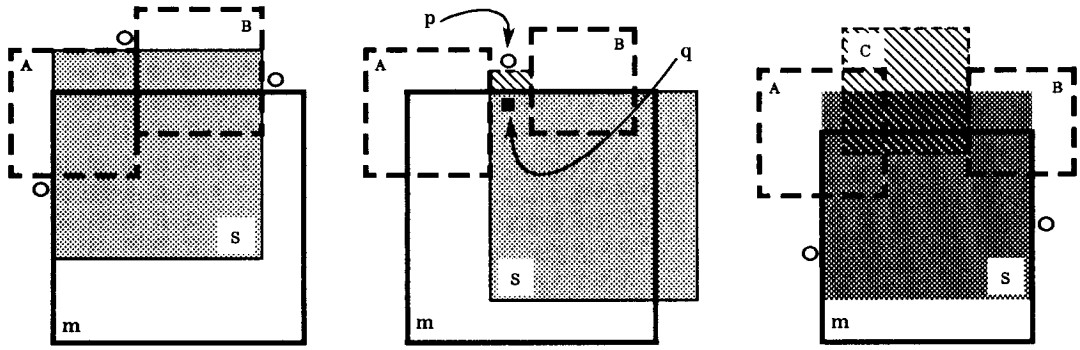
Theorem 1 The *basic cover* for a non-essential square **m** contains four or fewer maximal squares.

proof (Refer to the cases in figure 11.)

Let β_m be the basic cover for the given square **m**. If not more than one square in β_m extends past each edge of **m**, we are done. Else let **e** be some edge of **m** such that exactly two squares A and B in β_m extend past **e**. Let **e** be the top edge of **m** without loss of generality.

(a) If A and B abut or overlap along **e**, then there is another square S in β_m such that $S \cap \mathbf{m}$ strictly contains $(A \cup B) \cap \mathbf{m}$. Then neither of A and B belongs to β_m.

(b) Else A and B are separated by at least one white pixel **p**. Consider pixel **q** of **m**, which lies



a. Abutting / overlapping b. Separated squares c. 3 squares past same edge

Figure 11. Bounding the size of a Basic Cover

along e and is vertically below p. Let square S in β_m cover q. By assumption S does not extend past e, and hence it extends past the right or left edge of m. Then S∩m strictly contains either A∩m or B∩m. Which implies that either A or B does not belong to β_m.

(c) Else three or more squares A, B and C in β_m extend past some edge e of m. Let A, C, B ... etc. be the sequence of the squares from left to right, in ascending order of the X-coordinate of their left edges. We can apply the arguments in (a) and (b) above inductively to this sequence, and get a contradiction. □

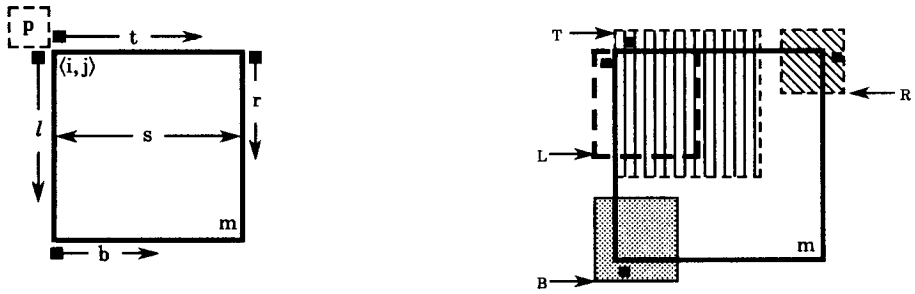
5.2 Finding the Basic Cover

We will now show that we can determine if a given maximal square is potentially redundant, and if so, find a candidate basic cover for it, all with only a constant number of operations per maximal square. Our algorithm for finding a basic cover relies on examining the length of maximal black strips which border the square m on each side, starting at each corner. This yields the sizes and locations of maximal squares which overlap maximally with m. A little case analysis helps to eliminate those squares which either do not have a cover completion, or whose overlap with m is subsumed by another square. At this point, a given square S which we have found to overlap maximally with m, may already have been discarded during the division of uniform strips. In that case we locate the predecessor / successor P of S in the uniform strip containing S and P. If P overlaps with m, we add P to the *normalized basic cover* of m.

Theorem 2 If a maximal square m is not essential, we can identify a candidate *basic cover* for it with only a constant number of operations.

proof (refer to Figure 12.)

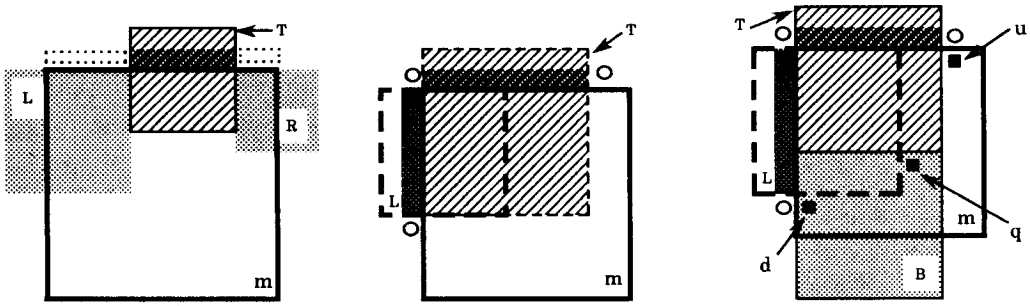
In what follows we will show how to identify those squares in β_m which cover the pixels along the *North* edge and *Northwest* corner of m. Similar arguments hold along the remaining edges of m. Let the *origin* of m be at $\langle i, j \rangle$, and *s* be its size. Let the image be given in array P. Let t and b be the lengths of the longest horizontal sequence of black pixels starting respectively at $\langle i-1, j \rangle$



a. Variables of interest b. Squares of interest

Figure 12. Identifying a candidate *basic cover*

and $\langle i+s, j\rangle$, and extending to the right. Let l and r be the lengths of the longest vertical sequence of black pixels starting respectively at $\langle i, j-1\rangle$ and $\langle i, j+s\rangle$ and extending downward. Recall that the lengths of all these black strips were determined when we were identifying the maximal squares in Section 3. Let $p = P_{i-1,j-1}$. Let T, L, R, B be the largest black squares which respectively contain the pixels at $\langle i-1, j\rangle$, $\langle i, j-1\rangle$, $\langle i, j+s\rangle$ and $\langle i+s, j\rangle$, and which extend maximally into m. We have the following cases, shown in figure 13 below.



1. $t = 0$ **2a.** $l<t<s$ & $p=0$ **2b.** $l=t<s$ & $p=0$

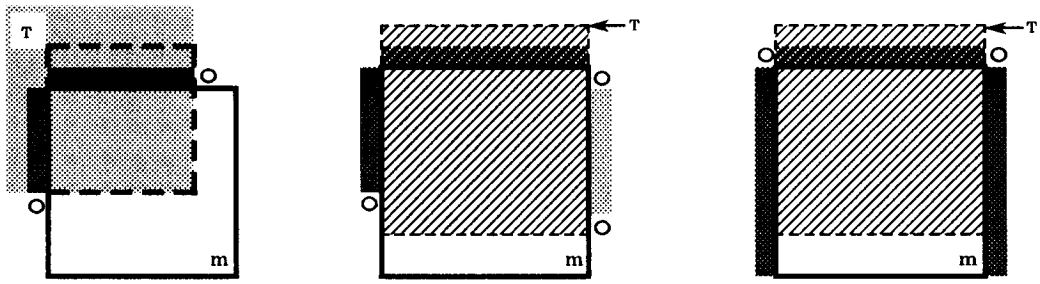
Figure 13. Cases for finding a candidate Basic Cover

(1) $(t=0)$. If there exists T in β_m such that T extends past the *North* edge of m, surely T must abut or overlap the right edge of L. Further, if $(l+r \geq s+2)$, then L abuts / overlaps with R, and we do not need look for T. Let h_1 be the length of the longest horizontal sequence of black pixels starting at $\langle i-1, j+l-1\rangle$ and extending to the right. We can see that the *Southeast* corner of T is at $\langle i+h_1-2, j+l+h_1-2\rangle$. Then one more lookup is enough to determine the *origin* of T. Further, if the right edge of T is to the left of the right edge of m, then another square R must abut / overlap the right edge of T, and R must cover the *Northeast* corner of m. Such an R will be added to the basic cover when we examine the *East* edge of m.

(2a) $(0<t<s) \wedge (p=0) \wedge (l<t)$. Clearly $l \leq t-1$. Also note that the origins L and T are respectively one column to the left of, and one row above the origin of m. Hence T covers t columns and $t-1$ rows along the top left corner of m; and L covers at most $t-1$ rows and $t-2$ columns along the top left corner of m. Whence $(T \cap m) \supsetneq (L \cap m)$, and $L \not\in \beta_m$. Clearly T is a

maximal square, and its origin is at $\langle i-1, j\rangle$. Add T to β_m. The case for $(p=0)$ and $(0<t<l<s)$ is similar.

(2b) $(0<t<s) \wedge (p=0) \wedge (l=t)$. Let d, q, u refer to the pixels respectively at $\langle i+l, j\rangle$, $\langle i+t-1, j+l-1\rangle$, and $\langle i, j+t\rangle$. Clearly d, q, u are not covered by either of L or T. Let $Q \in \beta_m$ such that Q covers q. We showed in the course of proving Theorem 1 that at most one square of β_m can extend past any given edge of m. Hence Q also covers one of d or u. In the former case $(b \geq l) \wedge (b+t \geq s+2)$, and $L \notin \beta_m$. In the latter case $(r \geq t)$ and $(l+r \geq s+2)$, whence $T \notin \beta_m$. If there are Q_1 and Q_2 in β_m such that Q_1 covers d and q, and Q_2 covers u and q, then $Q_1 \cup Q_2$ does not cover pixel $\langle i, j\rangle$ since m is maximal. Also both L and T are maximal, and one of them must be added to β_m in order to cover pixel $\langle i, j\rangle$.



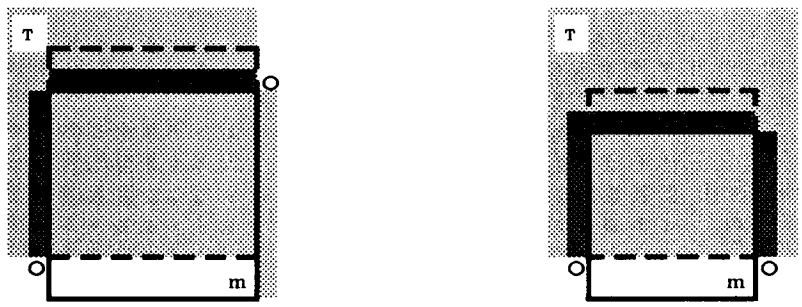
2c. $\max(l, t)<s \;\&\; p=1$ **3a.** $t=s \;\&\; \max(l, r)\leq s-2$ **3b.** $l=t=r=s$

Figure 13 (contd). Cases for finding a candidate Basic Cover

(2c) $(0<t<s) \wedge (p=1) \wedge \max(l, t)<s$. Let D be the maximal square containing the pixel at $\langle i-1, j-1\rangle$. Then clearly $(D \cap m) \supset (L \cup T) \cap m$, and $D \in \beta_m$. Clearly $\langle i+l-1, j+t-1\rangle$ is the *Southeast* corner of D, and one diagonal lookup of ${}^{NW}L_{i+l-1, j+t-1}$ is sufficient to locate the *origin* of D.

(3a) $(t=s) \wedge \max(l, r) \leq s-2$. Hence T is maximal, so add T to β_m.

(3b) $(t=s=l=r)$. Since m is maximal, both pixels at $\langle i-1, j-1\rangle$ and $\langle i-1, j+s-1\rangle$ are white. Hence T is maximal, so add T to β_m.



3c. $t=s \;\&\; r\geq s-1 \;\&\; p=1$ **3d.** $t>s \;\&\; p=1$

Figure 13 (contd). Cases for finding a candidate Basic Cover

(3c) $(t=s) \wedge (r \geq s-1) \wedge (p=1)$. Then since $\mathbf{m}$ is maximal, the pixel at $\langle i+s-1, j-1 \rangle$ is white, and $l \leq s-1$. Clearly $\langle i+s-1, j+s-1 \rangle$ is the *Southeast* corner of the square T in β_m which covers pixel $\langle i, j \rangle$. One lookup of ${}^{NW}L_{i+s-1, j+s-1}$ suffices to locate the *origin* of T. The mirror image (about the vertical axis) of this case is similar.

(3d) $(t>s) \wedge (p=1)$. Let $h=\min(l, r)$, $h_2=|\,{}^{NE}L_{i,j}\,|$, $v=|\,{}^{SE}L_{i-h_2+1,j}\,|$. Clearly $(h+h_2-1) \geq v$, and $\langle i+h-1, j+v-1 \rangle$ is the *Southeast* corner of T. One more lookup of ${}^{NW}L_{i+h-1, j+v-1}$ is enough to locate the *origin* of T. □

5.3 Determining if a Square is Essential

Once we have identified the set B of squares which overlap maximally with the given square $\mathbf{m}$, (and hence are candidates for inclusion in its basic cover) we have to verify if indeed the union of the squares in B covers $\mathbf{m}$. We show in Lemma 2 below that we can accomplish this task with only a constant number of operations.

Lemma 2 If B is a set of squares which overlap maximally with a given square $\mathbf{m}$, we can determine whether or not B covers $\mathbf{m}$, with only a constant number of operations. (See Figure 14.)

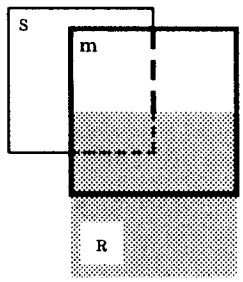


Figure 14. Verifying a cover

proof First we observe that by Theorem 1, B contains at most four maximal squares. Sort the squares in B cyclically about the perimeter of $\mathbf{m}$. Initialize the covered region R of $\mathbf{m}$ to the empty region. Next intersect each square S in B with $\mathbf{m}$, in the order obtained above. Add $S \cap \mathbf{m}$ to R, to get the augmented region of $\mathbf{m}$ covered by S. Note that we can keep track of R by keeping track of its perimeter, which consists of a constant number of line segments. Once we have added $S \cap \mathbf{m}$ to R, for each square S in B, it is easy to check if R equals $\mathbf{m}$. □

Combining Theorem 2 and Lemma 2, we get the following Theorem.

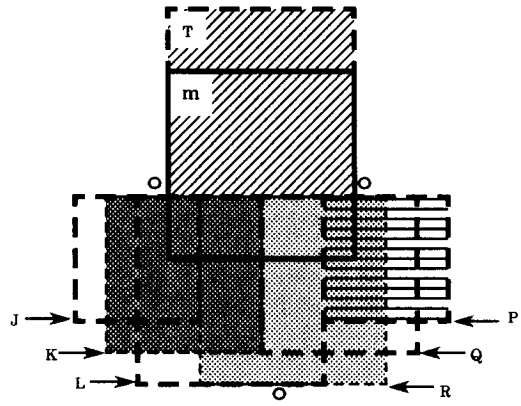
Theorem 3 We can determine whether a given maximal square $\mathbf{m}$ is *essential;* and if $\mathbf{m}$ is not essential we can identify its *basic cover* with only a constant number of operations. □

5.4 Extended Cover Relations

So far we have determined which of the maximal squares are essential; and for every square **m** which is not essential, we have determined a basic cover consisting of all those squares which overlap maximally with **m**. Clearly we must now determine all those squares which are "useful" for covering **m**, but are not in its basic cover. As we saw in Figure 10, not every square which overlaps with **m** is useful for covering it. If any square P is to be useful for covering **m**, there must be a set S of squares containing P, such that S covers **m**, but (S – P) does not. Which leads us to the following definition of an *irreducible cover*.

Definition (*irreducible cover*) : Let **m** be a maximal square which is not essential. An *irreducible cover* ξ of **m** is a set of maximal squares, such that ξ covers **m**, but no proper subset of ξ covers **m**.

We will now formalize the notion of the set of all squares which are useful for covering a given non-essential square **m**, by defining the *extended cover* of **m**. We know that every set S of squares which covers a given square **m** contains a subset T such that T is an irreducible cover of **m**. The squares in S – T cannot be useful for covering **m**, unless each of them independently belongs to some other irreducible cover of **m**.



$$m \subseteq (J \vee K \vee L)(K \vee L \vee R)(L \vee R \vee Q)(R \vee Q \vee P)T$$

Figure 15. Finding an un-necessarily fine cover for **m**

One more hurdle remains before we can define an extended cover for a non-essential square. Consider the collection of maximal squares shown in Figure 15 above. The square **m** has the following irreducible covers: {{T, J, R}, {T, K, R}, {T, K, Q}, {T, L, R}, {T, L, Q}, {T, L, P}}. In fact, we can extend this example so that $\Omega(\sqrt{n})$ squares overlap along the bottom left, and the bottom right corners of **m**, in the manner of J-L and P-R. Thus **m** has $\Omega(n)$ different irreducible covers, and any processing based upon the local neighborhood of each square, will not be possible within our stated resource bounds. Notice however, that the intersection of L and R contains at least one pixel which is not covered by any other square; hence one of L and R is always present in every

cover of the image. In which case, any square whose intersection with $\mathbf{m}$ is a strict subset of $L \cap R$, cannot really be useful for covering $\mathbf{m}$. In fact, as we shall see in Theorems 4 through 7, this last condition is both necessary and sufficient to enable us to construct a cover graph of constant degree, and to do so very efficiently.

Definitions (*pairwise essential squares, extended cover*) : A pair of maximal squares P and Q are *pairwise essential* if $P \cap Q$ contains a pixel $\mathbf{p}$ which does not belong to any other maximal square (other than P or Q). The *extended cover* Ξ_m of an inessential square $\mathbf{m}$ consists of all those retained squares P, such that (i) P belongs to some irreducible cover of $\mathbf{m}$, and (ii) $(P \cap \mathbf{m})$ is not a strict subset of $(X \cap Y)$ for any squares X and Y which are pairwise essential.

5.5 Finding the Extended Cover

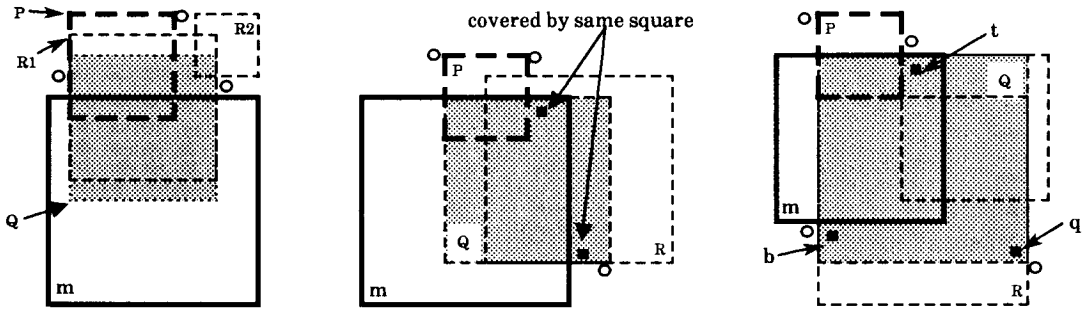
We can determine the extended cover Ξ_m of a square $\mathbf{m}$, starting with its *basic cover* β_m, and adding to it the basic cover β_S for every square S in β_m. Given two squares A, B in Ξ_m, we say that $A <_m B$ *iff* $A \cap \mathbf{m}$ is strictly contained in $B \cap \mathbf{m}$. This partial order separates the squares in Ξ_m into *chains*, each of which starts with a square that overlaps maximally with $\mathbf{m}$, i.e. is in its basic cover. We show in the following theorem that each such chain can be traced in decreasing order of overlap with $\mathbf{m}$, by a constant sized sequence of basic cover refinements.

Theorem 4 Let P be a maximal square in the extended cover Ξ_m of an inessential square $\mathbf{m}$, such that P does not belong to the normalized basic cover β_m of $\mathbf{m}$. There exists a square Q in β_m, such that P belongs to the basic cover β_Q of Q.

proof Let top(P), bot(P), lt(P), rt(P) respectively denote the four edges of a square P. Let the X-axis increase to the right, and Y-axis increase downwards, i.e. the image is in the fourth quadrant. Let *usd*(A, B) denote the division of the *uniform strip* containing squares A and B, and the associated discarding of squares. Let $P \in (\Xi_m - \beta_m)$. Clearly there exists Q in Ξ_m *st* $(Q \cap \mathbf{m}) \supseteq (P \cap \mathbf{m})$. Assume *wlog* that P extends above the top edge of $\mathbf{m}$. Consider the position of P *wrt* $\mathbf{m}$.

case 1 P covers no corners of $\mathbf{m}$.

Since $(Q \cap \mathbf{m}) \supseteq (P \cap \mathbf{m})$, and P covers no corners of $\mathbf{m}$: $lt(Q) \leq lt(P)$, $rt(Q) \geq rt(P)$, $top(Q) \leq top(\mathbf{m})$, and $|Q| \geq |P|$. Since P is unbounded below, it is bounded to its left and right by white pixels, and $top(Q) > top(P)$. Since $top(Q) > top(P)$ and $|Q| \geq |P|$, $bot(Q) > bot(P)$. If $|Q| = |P|$, then Q was discarded during *usd*(P, Q), and $P \in \beta_m$, a contradiction. Else $|Q| > |P|$. Let Q cover no corner of $\mathbf{m}$, as in fig16(a) below. If any square R other than Q covers the NE corner of Q: then R either forms a uniform strip with Q, or R does not abut / overlap the top edge of $\mathbf{m}$. In the latter case Q is essential, and $P \notin \Xi_m$. In the former case, the topmost square T in the uniform strip containing {Q, R} is essential, $(T \cap \mathbf{m}) \supseteq (P \cap \mathbf{m})$, and again $P \notin \Xi_m$.



a Q covers no corner of **m** **b** Q covers one corner of **m** **c** Q covers two corners of

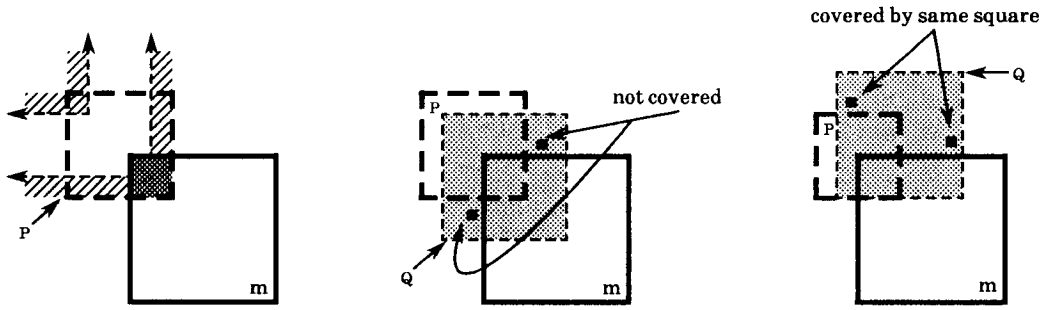
Figure 16. P in (Ext. Cov_m – Basic Cov_m) covers no corners of **m**

Else let Q cover one corner of **m**, and *wlog* let rt(Q) > rt(**m**), as in figure 16(b) above. Either the marked pixels of Q are covered by the same square R; or Q is pairwise essential with **m**, and $P \notin \Xi_m$. In the former case rt(R) > rt(Q), and $|R| \geq |Q|$. If $|R| > |Q|$, Q is part of a uniform strip all of which is covered by two larger squares R and **m**, and Q was discarded during *usd*(Q). Else Q and R form a uniform strip, and Q was discarded during *usd*(Q, R), a contradiction.

The last subcase is Q covers two corners of **m**, as in figure 16(c) above. Consider $R \in \Xi_Q$ *st* R covers the SE corner q of Q. Clearly R has its SE corner to the right of and / or below q. It is clear that if Q is not essential, and (Q, **m**) are not pairwise essential, then R covers pixel t or pixel b. In either case, R and Q are of the same size, and must be pairwise essential *wrt* the squares retained after the division of uniform strips. It follows that $P \notin \Xi_m$, a contradiction. In all cases, $P \in \beta_m$ or $P \notin \Xi_m$.

case 2 P covers only one corner of **m**.

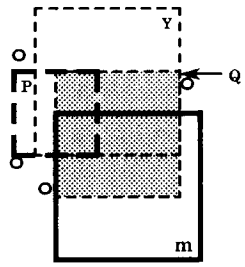
Since $(Q \cap \mathbf{m}) \supsetneq (P \cap \mathbf{m})$: we have (i) top(Q) ≤ top(**m**) and lt(Q) ≤ lt(**m**), (at least one inequality is strict), and (ii) P is maximal implies top(Q) > top(P) or lt(Q) > lt(P). Having constrained the origin of Q as above, we see that bot(Q) < bot(**m**) and rt(Q) < rt(**m**), since **m** is maximal. By examining the marked pixels in (B), one can see that if top(P) < top(Q) < top(**m**) and lt(P) < lt(Q) < lt(**m**), then Q is essential, hence $P \notin \Xi_m$. Hence (top(Q) ≤ top(P) or top(Q) = top(**m**)) or (lt(Q) ≤ lt(P) or lt(Q) = lt(**m**)).



a Allowed regions for origin of Q **b** Uncovered pixels of Q **c** Paired pixels of Q

Figure 17. P in (Ext. Cov_m – Basic Cov_m) covers one corner of **m**

Without loss of generality, let top(Q)≤top(P). One can easily see that the marked pixels in figure 17(c) are covered by the same square (say T) in Ξ_Q, unless Q is essential. In the latter case $P \notin \Xi_m$. In the former case, Q is maximal only if T and Q are of the same size; and T covers both marked pixels implies that it abuts / overlaps m: whence we get that Q must have been discarded during *usd*(T, Q), and $Q \notin \Xi_m$. Thus we get top(Q) equals top(P). In this case also, if lt(P)<lt(Q)<lt(m), then Q is essential as in (B) above. Putting all the above arguments together, we get (top(Q)=top(P) and lt(Q)=lt(m)) or (top(Q)=top(m) and lt(Q)=lt(P)).



d Configuration of P, Q, m

Figure 17 (contd). P covers one corner of m

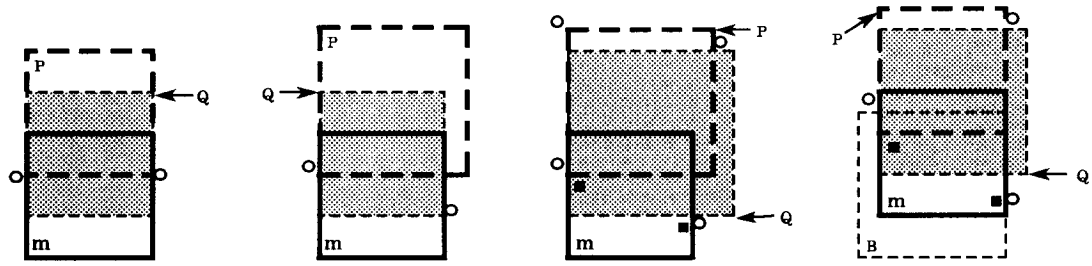
Let us fix the relative placements of P, Q and m as in Figure 17(d) above. If $Q \notin \beta_m$, ($\exists$ X$\in\beta_m$) *st* (X∩m) ⊋ (Q∩m). Since X covers all of Q∩m, it cannot extend past lt(m), else Q is not maximal. Since X covers the NW corner of m, it cannot extend past either rt(m) or bot(m), else m is not maximal. If X extends past top(m), then X and Q are of the same size, top(Q)<top(X)<top(m), and X must have been discarded during *usd*(Q, X): thus Q is in the normalized basic cover of m.

Since P$\in\Xi_m$ and (P∩m) ⊋ (Q∩m), we know that Q is not *essential*. If $P \notin \beta_Q$, ($\exists$ Y$\in\beta_Q$) *st* (Y∩Q) ⊋ (P∩Q). We note that since P and Q are maximal, P is bounded both above and below by white pixels, and Q is bounded to its left and right by white pixels. If lt(Y)<lt(m) & bot(Y)>bot(P), then Y is essential, and $P \notin \Xi_m$. There are only three possibilities for Y: either (i) top(Y)=top(P), bot(Y)=bot(P) and lt(P)<lt(Y)<lt(Q); or (ii) lt(Y)=lt(Q), rt(Y)=rt(Q), top(Y)<top(Q) and bot(P)≤bot(Y)<bot(Q); or (iii) rt(Y)=rt(Q), bot(Y)=bot(P) and lt(P)<lt(Y)<lt(Q). In case (i) Y is discarded during *usd*(P, Y); in case (ii) Q is discarded during *usd*(Y, Q); and in case (iii) Q is discarded during *usd*(Q). In all cases there is a contradiction, whence we have (P$\in\beta_Q$) and (Q$\in\beta_m$) as desired.

case 3 P covers two corners of m.

Since P covers two corners of m, |P|≥|m|. Without loss of generality, let top(P)<top(m). Then (Q∩m) ⊋ (P∩m) implies that Q covers the top row of m, and at least one row of m below bot(P). Either |P|=|Q|=|m|, and hence top(P)<top(Q)<top(m); in this case Q was discarded during *usd*(P, Q, R), and P$\in\beta_m$. Else |P|>|m|=|Q|, in which case (see figure 18(b)) Q was discarded during *usd*(Q, m), and again P$\in\beta_m$. (See Figure 18(a).)

Else Q is larger than m. Clearly m is unbounded above, and is bounded along both its left and



a $|P|=|m|=|Q|$ **b** $|P|>|m|=|Q|$ **c** $|P|>|m|, |Q|>|m|$ **d** $|P|=|m|<|Q|$

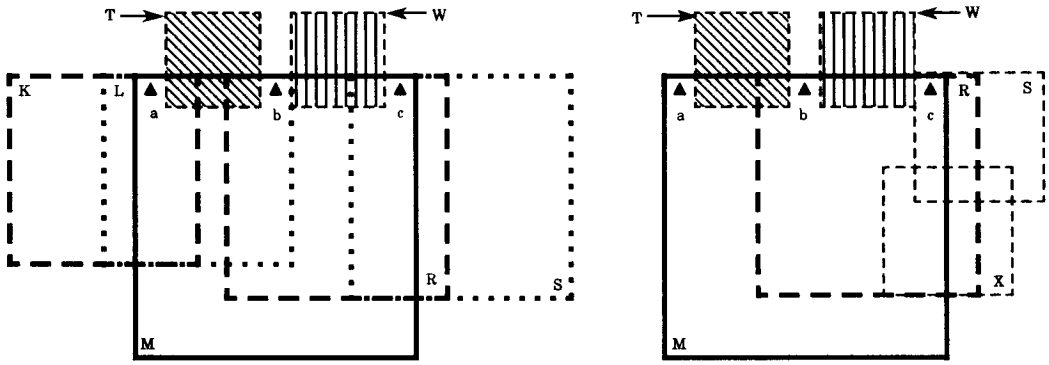
Figure 18. P in (Ext. Cov_m – Basic Cov_m) covers two corners of m

right edges by white pixels. Consider ξ_{mP}, an irreducible cover of m *st* $P \in \xi_{mP}$. By examining Figures 18(c) & (d) above, we can see that there is exactly one square, say B in ξ_{mP}, *st* $B \supset (m - P)$. Clearly $|B| \geq |m|$, and $bot(B) > bot(m)$. If $|B| = |m|$, m was discarded during *usd*(m, B), a contradiction. Else if $|B| > |m|$, m is part of a uniform strip: all of which is covered by two larger squares, whence m was discarded during *usd*(m), and again we have a contradiction. □

5.6 Bounded Size of the Extended Cover

It follows from Theorems 1 and 4 that the extended cover of any square contains only a constant number of squares. In the following theorem, we give an independent proof of this same fact.

Theorem 5 Let Ξ_m be the *extended cover* for an inessential square m. If two squares T and W in Ξ_m extend past the same edge e of m, then at least one of them extends past a corner of e.



a. Squares covering pixels a, b and c **b**. Finding a cover for R, $|S| < |R|$

Figure 19. Two squares in Ξ_m extending past same edge of m

proof (Refer to Figure 19.)

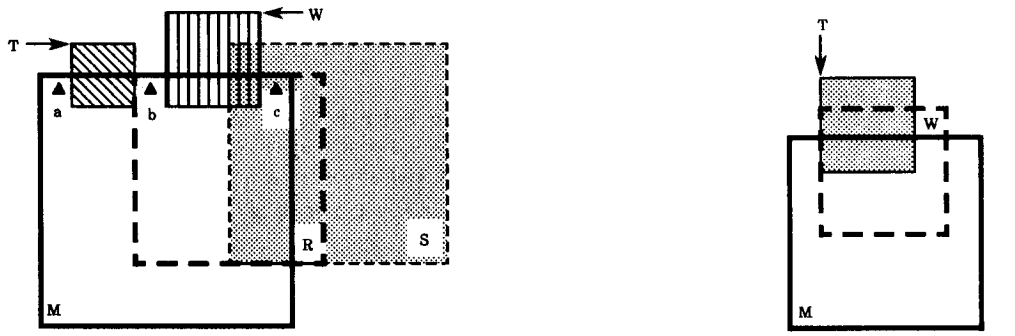
Assume to the contrary that neither of T and W covers any corner of e. Also *wlog* assume that e is the top edge of m, and T is to the left of W.

case 1: T and W do not abut / overlap along e. Consider pixels a, b, c $\in m$: all three are adjacent to e, and respectively to the left of T, between T and W, and to the right of W, as shown

above. Let $\xi_T \subseteq \Xi_m$ be an irreducible cover of m, *st* $T \in \xi_T$. Surely we must have K in ξ_T *st* K covers a but not b or c: else $\xi_T \not\supseteq m$, or $T \notin \xi_T$. Similarly, we have R in ξ_T *st* R covers b but not a. Neither of K and R can extend above the top edge of T (else T would not be maximal); hence K extends past the left edge of m, and R extends past the right edge of m. Thus R covers b and c but not a. Similarly let $\xi_W \subseteq \Xi_m$ be an irreducible cover of m, *st* $W \in \xi_W$. We have L in ξ_W *st* L covers a and b but not c, and S in ξ_W *st* S covers c but not a or b. This situation is depicted in Figure 19(a). Now R and S both overlap with m, and extend past its right edge: both are not bounded on the left, hence they are bounded above and below by white pixels. Also S extendss to the right of R, so it cannot extend past the bottom edge of R.

case 1a: Either R and S both have the same size (Figure 19(a)): in which case R would have been discarded during the division of the *uniform strip* containing R and S. Then T does not belong to any irreducible cover of m in M_U, and $T \notin \Xi_m$.

case 1b: The bottom edge of S is above the bottom edge of R (Figure 19(b)). Consider any square X in Ξ_m *st* X covers $R-(m \cup S)$. X must abut / overlap S along the bottom edge of S. The right edge of X is to the left of the right edge of S, else S is not maximal. Hence the left edge of X is to the right of the right edge of m, else X is not maximal. Hence X does not cover all of $R-(m \cup S)$. By induction on the number of squares needed to cover all of $R-(m \cup S)$, it follows that R is essential. Since $(R \cap m) \supsetneq (W \cap m)$, by the definition of extended covers $W \notin \Xi_m$.



c. Finding a cover for R, $|S| > |R|$ **d**. T and W overlapping

Figure 19 (contd.) Two squares in Ξ_m extending past same edge of m

case 1c: R and S have their bottom edges aligned, and S extends above T (Figure 19(c)). Then all of R is covered by m and S together: both of which are wider than R. In which case also, R is discarded during the division of the uniform strip containing R, and $T \notin \Xi_m$.

case 2: T and W abut / overlap along e. Then one of them (say T) extends farther above e than the other (Figure 19(d)). Because T and W are unbounded below, they are both bounded by white pixels along their left and right edges. By the same argument as applied *wrt* S and R above (cases 1a, 1b): either (i) T and W are of the same size, and W (the lower one) was discarded during the

division of the *uniform strip* containing T and W; or (ii) W (the lower and wider one) is essential, and $(W \cap \mathbf{m}) \supsetneq (T \cap \mathbf{m})$. In the latter case $W \notin \Xi_{\mathbf{m}}$. □

Corollary The *extended cover* of any maximal square contains only a constant number of squares.

6 Deriving the Cover Graph

During each phase of the cover refinement, we would like to select a large independent set of squares which can be discarded in parallel, while simultaneously guaranteeing that the entire image is covered at all times by the undiscarded squares. Consequently, for every square $\mathbf{m}$ in the image, and an arbitrary set S of maximal squares, we need to be able to efficiently determine if S covers $\mathbf{m}$. In order to do this, we examine the extended cover $\Xi_{\mathbf{m}}$ of every non-essential square $\mathbf{m}$ in the image, determine the geometric covering relationships amongst the squares in $\Xi_{\mathbf{m}}$, and represent these relationships compactly using a *cover graph* as introduced in section 2.

Clearly a square $\mathbf{m}$ is covered if each of its pixels is covered; and a chosen pixel $\mathbf{p}$ in $\mathbf{m}$ is covered by any one of the squares in $\Xi_{\mathbf{m}}$ which contain $\mathbf{p}$. Obviously we can aggregate all those pixels in $\mathbf{m}$ which are covered by the same subset of squares in $\Xi_{\mathbf{m}}$ into a single region. Further, if pixels $\mathbf{p}$ and $\mathbf{q}$ of $\mathbf{m}$ are covered respectively by two different subsets S_p and S_q of $\Xi_{\mathbf{m}}$, such that S_p is a strict subset of S_q, then we can merge the regions containing $\mathbf{p}$ and $\mathbf{q}$ into a single region. This is possible since whenever $\mathbf{p}$ is covered (by some square in S_p), so is $\mathbf{q}$. The resulting region will clearly equal the intersection of all the squares in S_p, and be rectangular. The result of all this is that we can replace $\mathbf{m}$ by a set $R_{\mathbf{m}}$ of its rectangular subregions, such that (i) the union of the rectangles in $R_{\mathbf{m}}$ equals $\mathbf{m}$; and (ii) in every irreducible cover ξ of $\mathbf{m}$, each rectangle in $R_{\mathbf{m}}$ is covered by a single square in ξ.

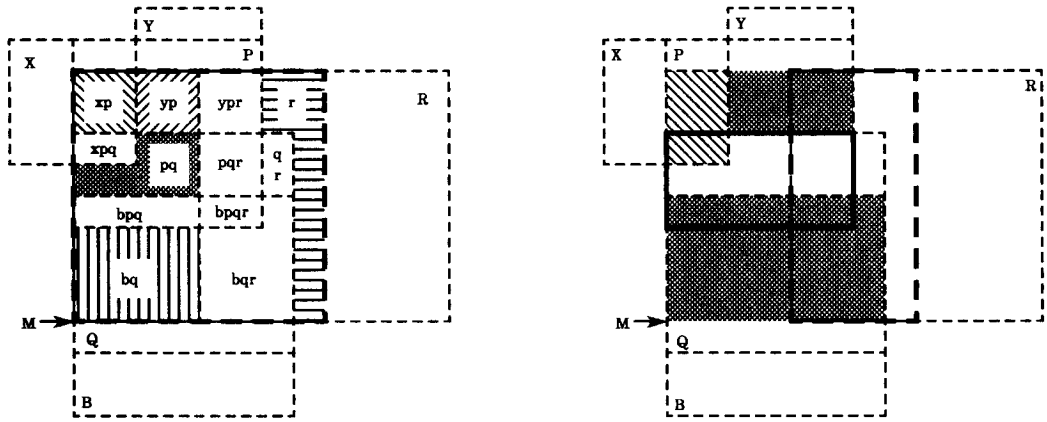
The resulting *reduced cover graph* $G \doteq \langle (M, R), E \rangle$ is directed and bipartite, with vertex sets M (maximal squares) and R (rectangles). The rectangles in R are derived by a suitable decomposition of the squares in M, as given in Section 6.1. The edge set E is a subset of $(M \times R) \cup (R \times M)$, and denotes containment between the squares and rectangles as follows. (i) The union of all the squares in M spans the entire image, (ii) a given square is contained in the union of all of its predecessor rectangles, (iii) a given rectangle is contained in every one of its predecessor squares, and (iv) a given maximal square $\mathbf{m}$ is covered by a subset S of $\Xi_{\mathbf{m}}$, *iff* every rectangle preceeding $\mathbf{m}$ has a predecessor $\mathbf{t}$ such that $\mathbf{t}$ belongs to S.

6.1 Unique Rectangular Decomposition of Squares

Given a maximal square $\mathbf{m}$, consider the set C_m of all irredundant convex covers of $\mathbf{m}$, given by $C_m = \{\mathcal{X} \mid \mathcal{X}$ is a set of convex subregions of $\mathbf{m}$ whose union covers $\mathbf{m}$; and for all distinct $\mathbf{p}$, $\mathbf{q}$ in $\mathcal{X}$, $\mathbf{p}$ does not contain $\mathbf{q}\}$. There is a naturally defined complete partial order $\propto$ on C_m, such that for all pairs $\mathcal{X}_1, \mathcal{X}_2$ in C_m, $\mathcal{X}_1 \propto \mathcal{X}_2$ *iff* $(\forall\, \mathbf{p} \in \mathcal{X}_1)\ (\exists\, \mathbf{q} \in \mathcal{X}_2)$ such that $\mathbf{p} \subseteq \mathbf{q}$. The minimum element $\perp$ of the lattice $\langle C_m, \propto \rangle$ is the pixel level partition of $\mathbf{m}$, and the maximum element $\top$ is $\{\mathbf{m}\}$. (Without the irredundancy condition in the definition of C_m, we can get two different covers which are less than or equal to each other.)

Definitions (*compatible cover, induced cover*) : Let S be a subset of $(M - \{\mathbf{m}\})$ which covers a given maximal square $\mathbf{m}$, and I_S be the set of all subsets of S which constitute an irreducible cover of $\mathbf{m}$. Then a cover $\mathcal{X} \in C_m$ is *compatible* with S, if every region $\mathbf{r}$ in $\mathcal{X}$ can be covered by some single square in any $T \in I_S$. The *induced cover* $\eta_m(S)$ is the *least upper bound* of the covers $\mathcal{X}_m$ of $\mathbf{m}$ which are compatible with S.

Theorem 6 Let S be a set of maximal squares whose union contains a given maximal square $\mathbf{m}$, and $\mathbf{m} \notin S$. Then the *induced cover* $\eta_m(S)$ is *compatible* with S.



a. Partition of M (*residual regions* are shaded) **b.** Induced Cover for M

Figure 20. Derivation of an Induced Cover

proof (Refer to the construction shown in Figure 20.)

Partition $\mathbf{m}$ into maximal connected regions, such that all the pixels in a given region are contained in exactly the same subset of squares in S. (In practice, this can be done efficiently by performing a *plane sweep* of the collection of edges of the squares in S.) Label each region with the largest subset of squares in S which contains it. Mark those regions $\mathbf{r}$ as *residual*, for which there is no other region $\mathbf{t}$ in the partition of $\mathbf{m}$ such that $\mathbf{t}$ is labelled with a proper subset of the squares which label $\mathbf{r}$. (Observe that the residual regions correspond to the conjuncts in the minimum CNF representation of the cover function for $\mathbf{m}$ restricted to subsets of S.) For every non-residual region $\mathbf{t}$, add it to every residual region $\mathbf{r}$ which is labelled with a proper superset of the squares

which label t. Finally discard all the non-residual regions. Call the remaining set of (augmented residual) regions as η. It is obvious that η as constructed is compatible with S.

Now we will show that η is also the *least upper bound* of the reduced covers x_m of $\mathbf{m}$ which are compatible with S. Let α, β be a pair of overlapping regions in $\eta_m(S)$. Let $\gamma = \alpha \cap \beta$, and $\delta \subseteq \gamma$. Then $(\eta_m(S) - \{\alpha, \beta\}) \cup \{\alpha \cup \delta, \beta \cup (\gamma - \delta)\}$ is also a cover of $\mathbf{m}$ which is compatible with S, for all $\delta \subseteq \gamma$. If we similarly remove the overlap between all pairs of regions in $\eta_m(S)$, we can obtain all the reduced covers of $\mathbf{m}$ by apportioning the shared regions in varying amounts between the overlapping pairs, (and dividing non-convex regions into convex pieces), and the desired conclusion follows. □

We can obtain the cover graph for the entire image by unioning the cover graphs for each of the undiscarded squares. During this step, we also apply the graph transformation shown in Figure 21, (see the *Global implication* rule of Section2.1) to get the *reduced cover graph* shown in Figure 22 below.

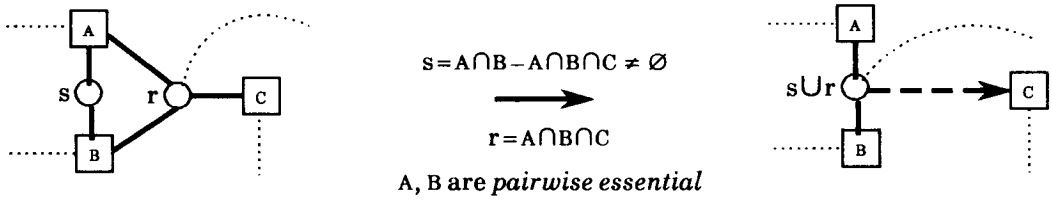
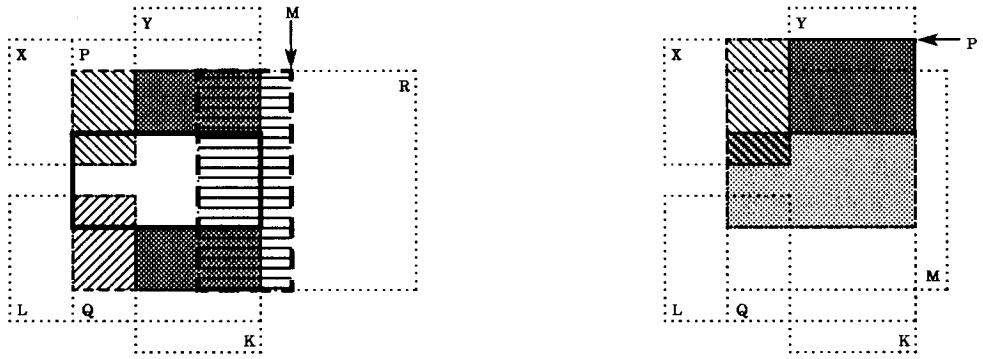


Figure 21. Global Implication Rule



$M \subseteq (X \vee P)(Y \vee P)(P \vee Q)(L \vee Q)(K \vee Q)\ R$ $\qquad$ $P \subseteq (Q \vee M) \wedge X \wedge Y$

Figure 22. Derivation of a Reduced Cover Graph

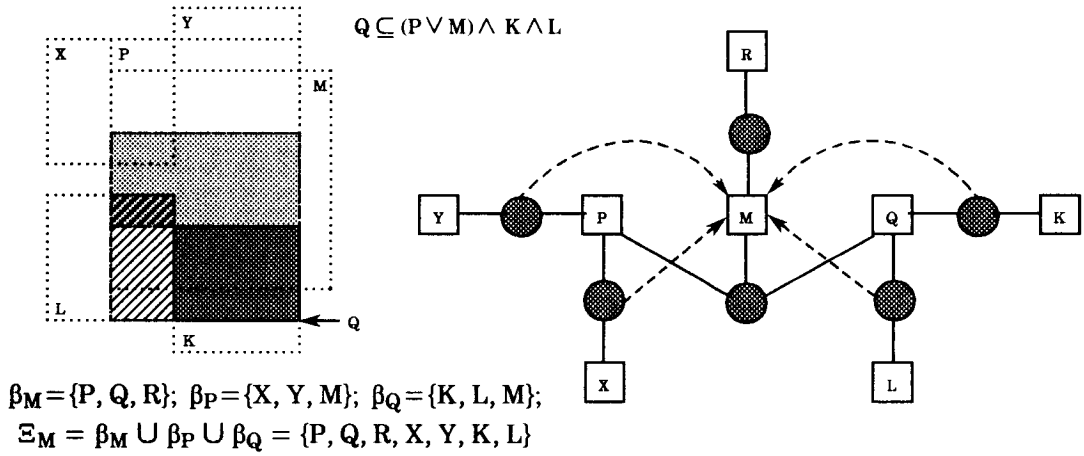


Figure 22 (contd.) Derivation of a Reduced Cover Graph

Theorem 7 The degree of the reduced cover graph is bounded by a (small) constant.

proof Follows from Theorem 5, and our construction of the cover graph. □

7 Finding a Minimal Square Cover

In this section we will find a minimal square cover for the image by determining a minimal vertex cover for its cover graph. We are given the cover graph $G \doteq \langle (M, R), E \rangle$, which is directed and bipartite. The two vertex sets of G are M (the maximal squares in the image) and R (the rectangles into which the squares in M are decomposed); and its edge set E is a subset of $(M \times R) \cup (R \times M)$. A minimal vertex cover for G is not an arbitrary subset of $(M \cup R)$, but rather it is a subset of M alone. We define below the concept of *minimal one-sided vertex cover* for a bipartite graph, in a manner analogous to its counterpart for a unipartite graph.

Definition (*minimal one-sided vertex cover*) : A subset $M_C \subseteq M$ is a *minimal one-sided vertex cover* for G, *iff* (i) (**safety**): every vertex r in R has at least one predecessor in M_C, and (ii) (**minimality**): every vertex m in M_C has a successor r (called its *witness*), such that r has no predecessor in $(M_C - \{m\})$.

For the sake of comparison, we also define the minimal vertex cover of a unipartite graph $G = \langle V, E \rangle$, in the same form as given above. A subset $V_C \subseteq V$ is a *minimal vertex cover* for G, *iff* (i) (safety): no vertex v in $(V - V_C)$ has a neighbor w also in $(V - V_C)$, and (ii) (minimality): every vertex v in V_C has a neighbor w in $(V - V_C)$.

7.1 Coloring the Cover Graph

In order to find a minimal cover for the image, we determine a 2-coloring of the *squares* in G, using *white* (discard) and *black* (retain). The *rectangles* are also colored in the manner implied by the containment relationships in G. We use three colors to represent the covering state of a rectangle: *black* (multiply covered), *gray* (uniquely covered) and *blue* (not covered). Before starting execution of the graph coloring algorithm we color all the squares *gray*, signifying that we are yet undecided whether to retain or discard them. We also color all the rectangles *white*, and let a *white rectangle* signify that we have not yet decided whether it will be uniquely or multiply covered. Since we never leave any pixel of the image without a cover during the intermediate steps of coloring, we will never need to color any rectangle *blue*.

The coloring is *consistent* (with respect to the covering relationships) if (i) a gray rectangle has only one black predecessor with all others being white, (ii) a black rectangle has two or more black predecessors, and (iii) a white rectangle has no more than one black predecessor. When the algorithm terminates, the coloring is *safe* (a complete cover) if every white square has only black or gray predecessors, and *minimal* if every black square has at least one gray successor. During the execution of the graph coloring algorithm we maintain *consistency*, and also the following two (weaker) invariants :

(i) **Safety** : Every rectangle has at least one black or gray predecessor.

(ii) **Minimality** : A black square has at least one gray or white successor.

The algorithm terminates when every square becomes black or white, and every rectangle becomes black or gray. The conjunction of the termination predicate with the invariants implies that a *minimal cover* has been found.

We initialize the coloring algorithm by coloring all the squares gray, and all the rectangles white. Next we delete all the unidirectional edges from the cover graph G (i.e. delete edge $\langle \mathbf{r}, \mathbf{m} \rangle$ iff $\langle \mathbf{m}, \mathbf{r} \rangle$ is not in G), to get an undirected graph G_u. Note that this has no effect on the minimal cover obtained, which can be seen as follows. Consider a one-sided minimal vertex cover M_c of G. If $\mathbf{m}$ belongs to M_c, it has a successor $\mathbf{t}$ (different from $\mathbf{r}$) as a witness. If $\mathbf{m}$ does not belong to M_c, then we require that $\mathbf{r}$ be covered by some square S (different from $\mathbf{m}$) in M_c, which is always true irrespective of whether $\mathbf{m}$ belongs to M_c. After constructing the undirected graph G_u, we color all the essential squares black (i.e. we retain them), and place a token on every successor of a black square.

We obtain a minimal cover for G by coloring its vertices in *phases* as described below. During each phase we examine the residual subgraph G_r of G induced by the gray squares and white

rectangles, and identify a set of non-intersecting maximal *chains* of vertices and edges in G_r, by suppressing a suitable set of edges. (A *chain* is any simple path in G_r.) We obtain a 2-coloring of the *squares* in each chain, such that alternating squares in a chain are black and gray. Then we stitch the chains back into G_r by reinstating the edges suppressed earlier. For each square which is colored black during the current phase, we place tokens on all of its successor rectangles, signifying that each such rectangle is covered by at least one of the retained squares. Next we examine the white predecessors of each gray square in G_r, and discard (color white) the gray squares which have no uncovered predecessor rectangles. Finally we recolor the rectangles in G_r to re-establish consistency of the coloring state. Recall from Theorem 7 (Section 6.1) that G has a constant degree. During each phase of coloring, the degree of gray squares decreases by at least two, hence a constant number of phases suffices to color all of G. (See Figure 23.)

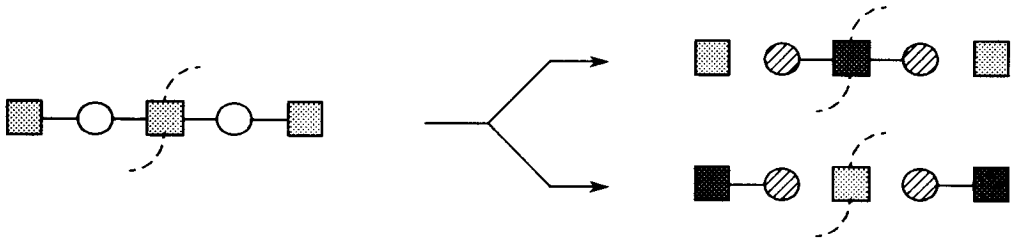


Figure 23. The two cases for 2-coloring extracted *chains*

Now let us see how we can extract sets of non-intersecting maximal chains from the residual graph G_u during each phase of coloring. Starting at each gray square $\mathbf{m}$, examine all the paths of length two which end at another gray square. Notice that each such path from $\mathbf{m}$ leads to only some square S_i in the extended cover Ξ_m of $\mathbf{m}$, and hence by Theorem 5 there are only a bounded number of such paths. We can consider each such path $\langle \mathbf{m}, \mathbf{r}, S_i \rangle$ to be really a path from the origin of $\mathbf{m}$ to the origin of S_i. Now consider the cyclic order (say clockwise) in which the origins of each S_i (reachable from $\mathbf{m}$ via two consecutive edges in G_u) appear with respect to the origin of $\mathbf{m}$. We can select the one which appears closest to the North-going edge from the origin of $\mathbf{m}$. Thus we obtain a directed spanning forest for G_u. If a square S has multiple paths directed into it, we can select the one which originates at a square whose origin is closest to the South-going edge from the origin of S (in anticlockwise order about the origin of S). By thus insisting that the extracted chains be oriented monotonically along both axes, we can guarantee that we choose only simple acyclic maximal paths. Of course, we can prefer paths running Northeast and Northwest during alternate phases of coloring.

Finally, let us examine the time bounds for coloring in the manner described above. We need only O(1) computational steps per maximal square during each phase, for obtaining the residual graph, extracting the set of non-intersecting chains, and for re-establishing the coloring invariants after 2-coloring each chain. We can 2-color all the chains in $O(\log n)$ time, on an EREW–PRAM with $(n / \log n)$ processors, using the *optimal parallel list ranking* technique of Cole

and Vishkin [CoV-88b]. Each phase of coloring reduces the residual degree (in G_u) of the gray squares which participate in chains by at least two. Note that a gray square which does not participate in any chain during a given phase, will always be discarded at the end of that phase, which can be seen as follows. If a gray square S does not participate in any chain, all of its neighboring rectangles have a predecessor which participates in some chain; hence each of those rectangles acquires a cover after the chains have been 2-colored, and S is discarded at the end of that phase. Thus we see that a constant number of phases of coloring suffices to color all of G, and we can meet our overall resource bounds for coloring. A complete example is shown in figures 24 through 27.

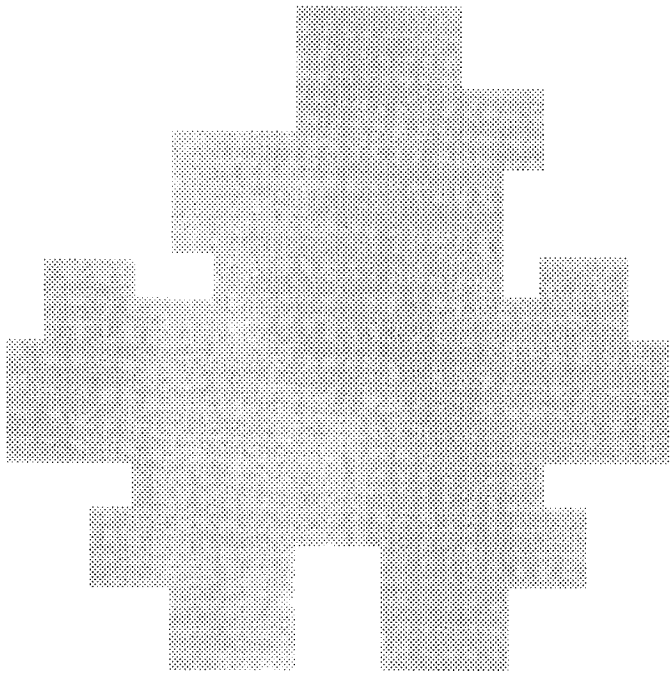


Figure 24(a). Example image for the coloring algorithm

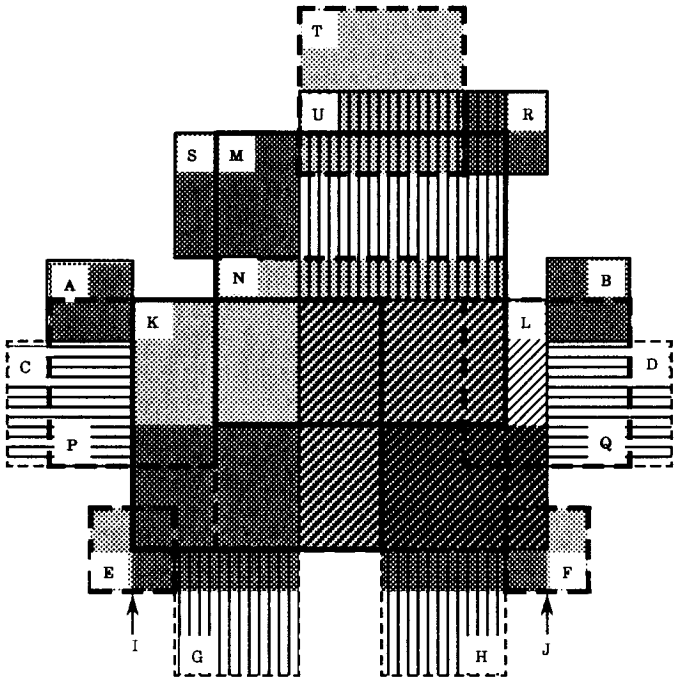


Figure 24 (b). Squares retained after division of *uniform strips*

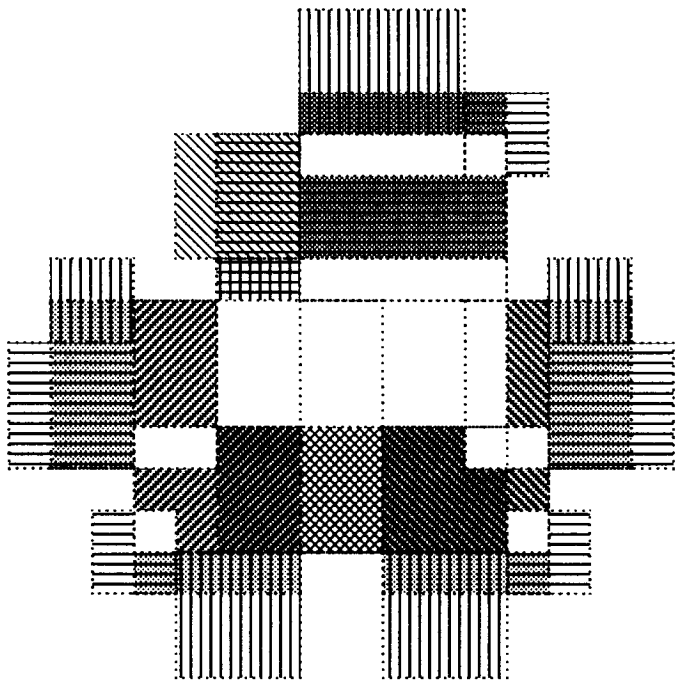


Figure 24 (c). *Residual regions* which determine the cover graph

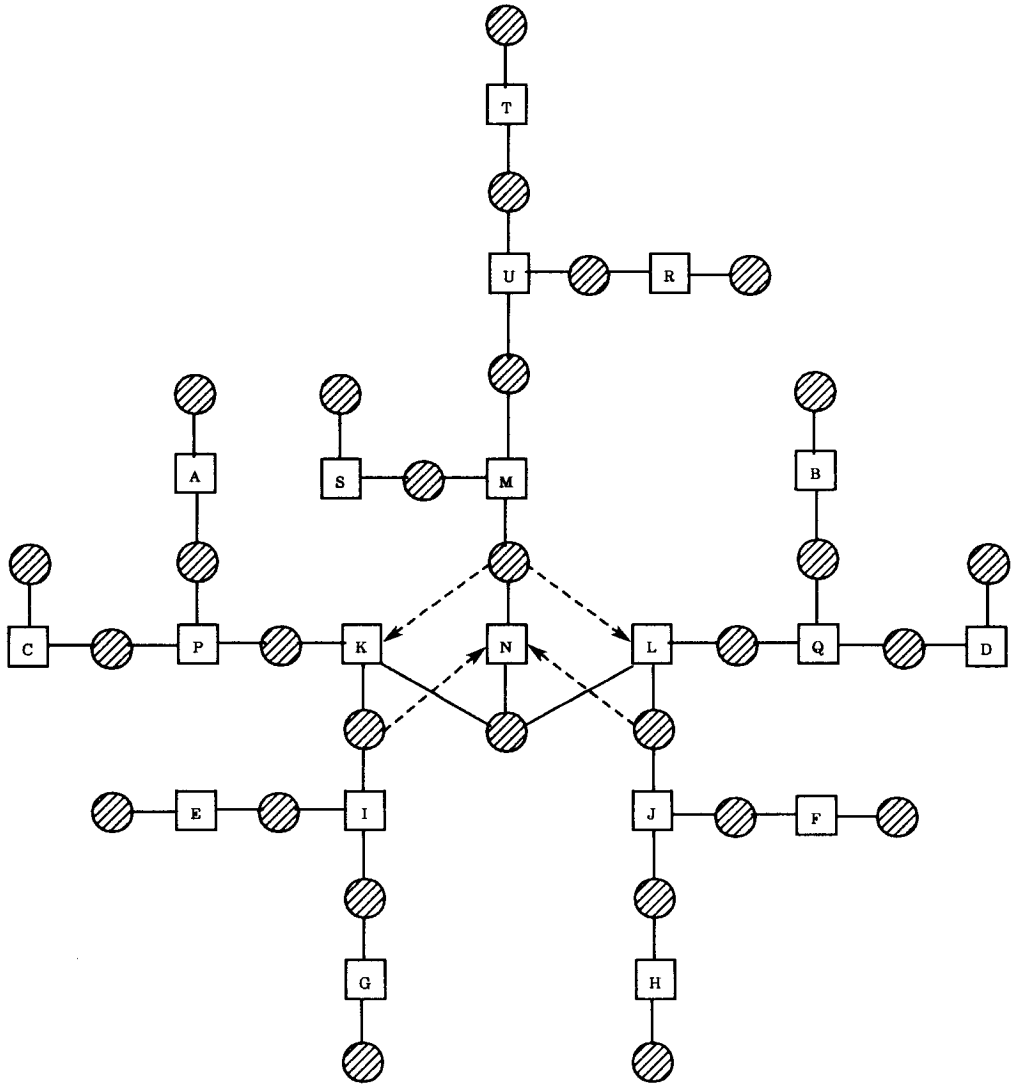


Figure 25. Reduced cover graph for the image in figure 24

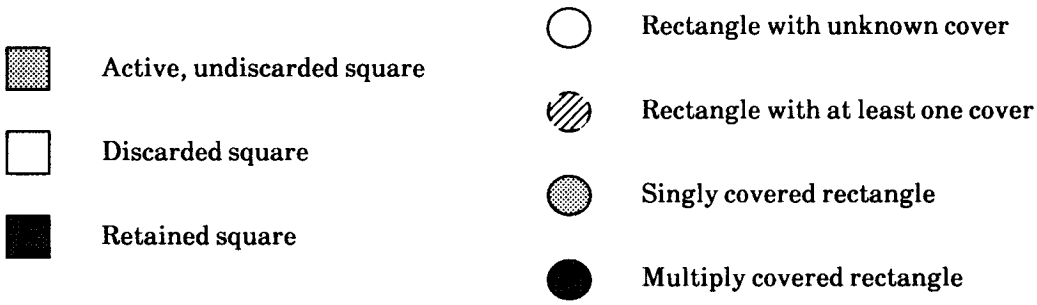
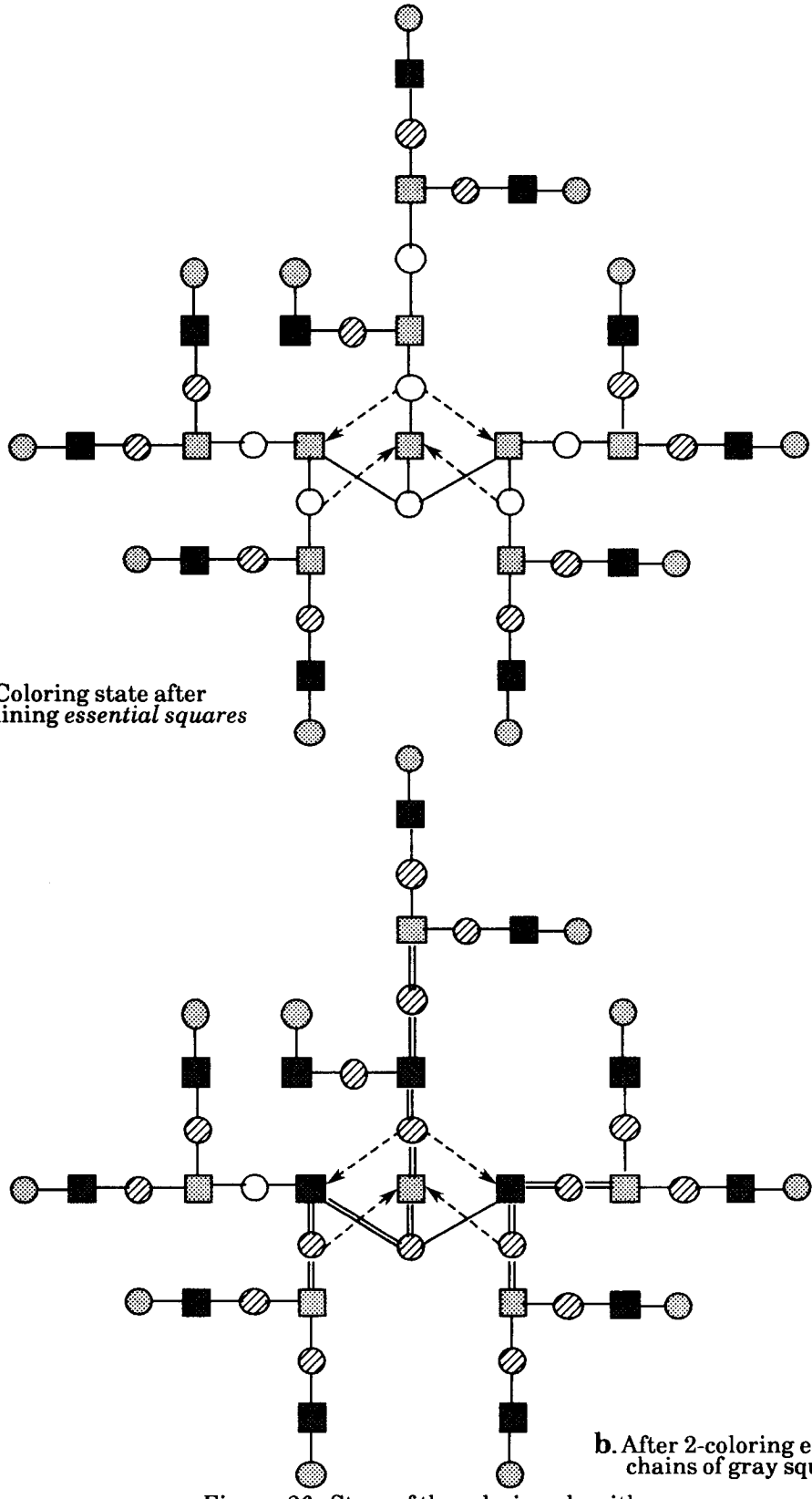


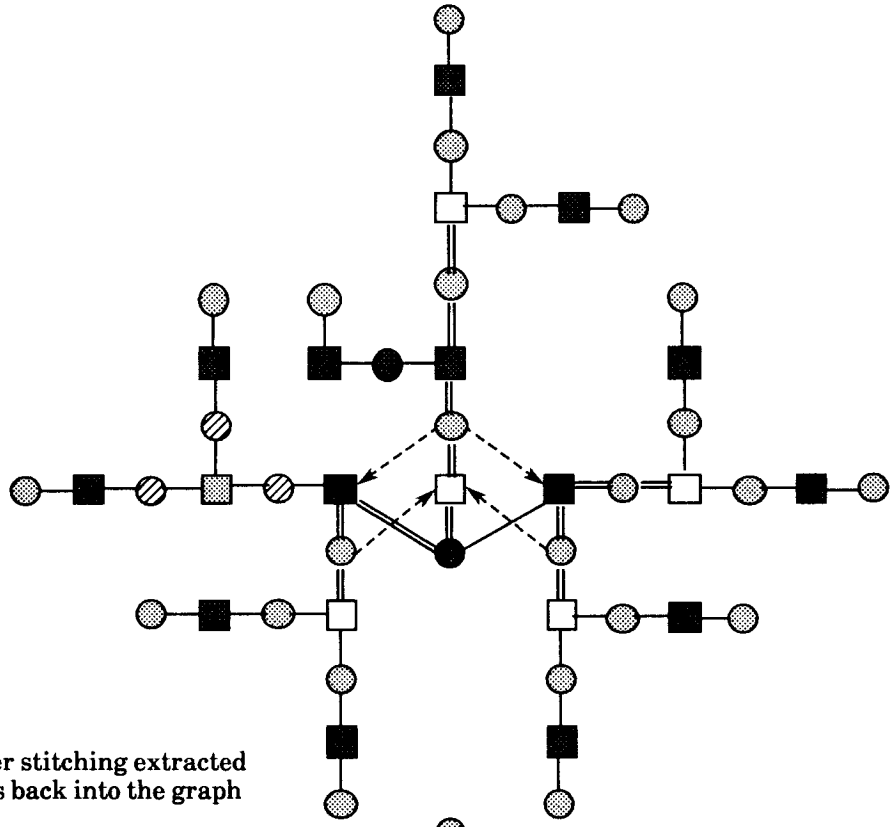
Figure 25(b). Color key for figure 26



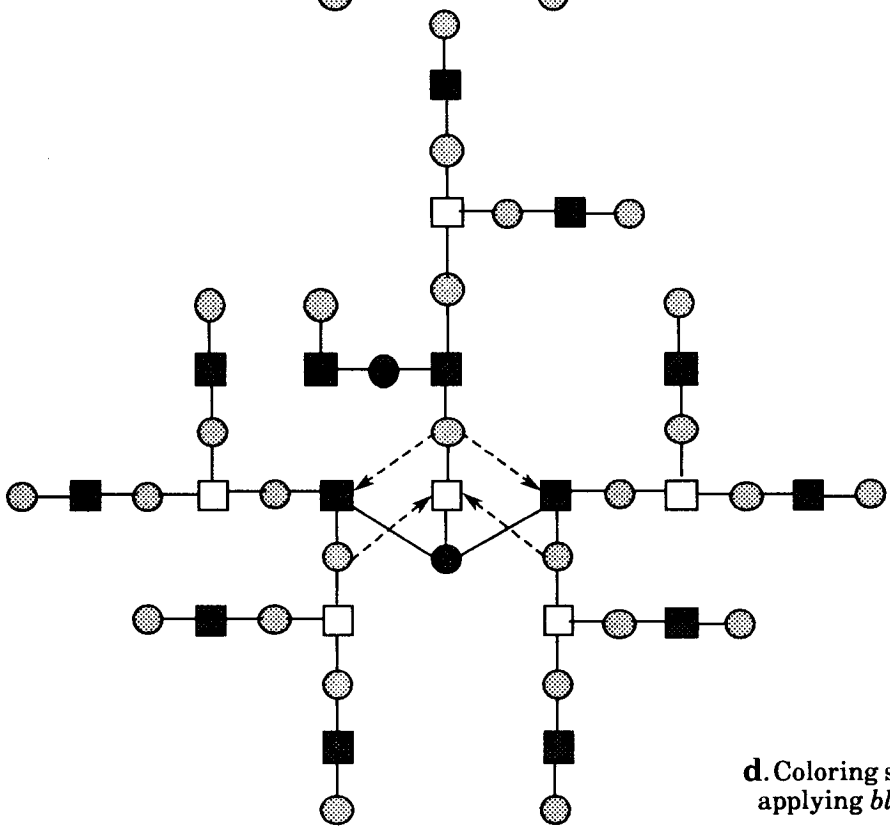
a. Coloring state after determining *essential squares*

b. After 2-coloring extracted chains of gray squares

Figure 26. Steps of the coloring algorithm



c. After stitching extracted chains back into the graph



d. Coloring state after applying *black rule*

Figure 26 (contd.) Steps of the coloring algorithm

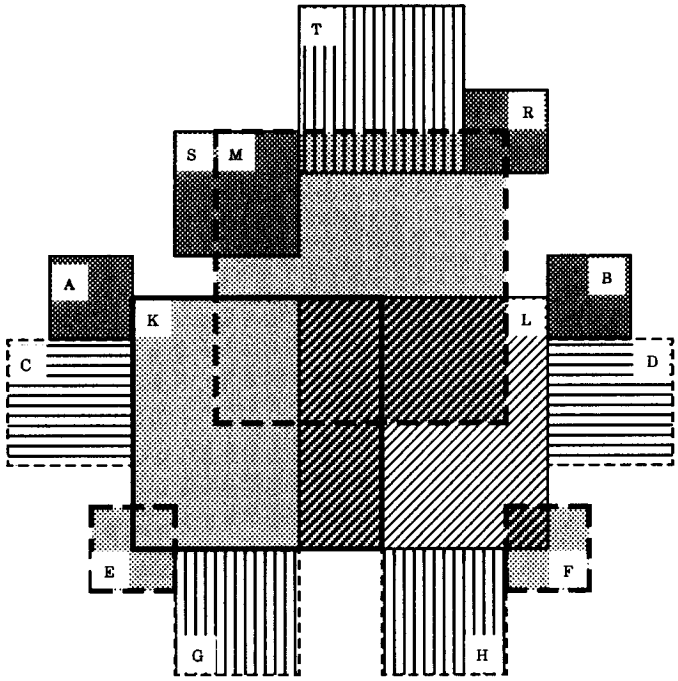


Figure 27. Irreducible cover obtained by the coloring algorithm

8 Conclusions

Our algorithm presently runs in O(log n) time on an EREW-PRAM with (n / log n) processors. It follows from the lower bound on the time for computing the boolean OR / AND of n bits [CDR-86], that minimal covers with maximal squares cannot be found in less than Ω(log n) time on exclusive write PRAMs (EREW / CREW) with a polynomial number of processors. (Given a vector of n bits, we can transform it in O(1) time into a $\sqrt{n} \times \sqrt{n}$ binary array, and ask if $\langle 0, 0 \rangle$ is the origin of a maximal square of size $\sqrt{n}$.) However, except for the single step of obtaining the ranks of elements from the head of each *chain* (linked list) of undiscarded squares, all the other steps can be performed in O(log n / loglog n) time on a CRCW-PRAM with (n loglog n / log n) processors [CoV-88a, CoV-88b]. If the *list ranking* problem gets solved for O(log n / loglog n) time, so will our algorithm. Alternatively, one could consider representing the chains in arrays, by exploiting the bounded degree of the cover graph, as well as the inherent directional ordering amongst its edges. Then a prefix computation would yield the desired result in sub-logarithmic time.

Another problem worth exploring is to cover the polygonal image with maximal rectangles, instead of squares. Clearly a cover with rectangles will always provide better image compression than one with squares. However, it is known that finding a *minimum rectangular cover* (with overlaps allowed), for an arbitrary polygonal region (with or without holes) is *NP-hard* [Mas-79, CoR-87, CuR-88]. Franzblau and Kleitman [FrK-84] find a minimum rectangular cover for

polygons which are convex along one axis, using $O(n^2)$ time. So one can only hope to find a minimal cover with maximal rectangles.

Another possibility is to *partition* the image into rectangles or squares. Ferrari et. al. [FSS-84] give a sequential algorithm for finding the *minimum rectangular partition* of a binary image, but their algorithm relies on finding a minimum vertex cover for a bipartite graph. There does not seem to be any simple way of deriving a deterministic parallel (NC) algorithm utilizing their results, since for an image with holes, their cover graph is not convex. (Although finding the minimum vertex cover for an arbitrary graph is *NP-hard*, Dekel and Sahni [DeS-84] give an NC algorithm for finding the minimum vertex cover for *convex bipartite* graphs.) Also, partitioning a polygon into squares or rectangles will always provide worse compression than would a cover. For example, there exist polygons for which k rectangles (squares) suffice for a cover, but $\Omega(k^2)$ rectangles (squares) are required for a partitioning. However, in applications where it is important for the cover to be of disjoint elements, as in the convolution application suggested by Ferrari et. al., we will need to obtain a partition of the image.

Acknowledgements

I gratefully acknowledge the many constructive discussions with Gianfranco Bilardi, Kieran Herley, S. Seetharam Iyengar, Kevin Karplus, Abha Moitra and Joseph Mitchell.

References

[ACK-88] L. J. Aupperle, H.E.Conn, J. M. Keil and J.O'Rourke, "Covering Orthogonal Polygons with Squares", *26th Ann. Allerton Conf. on Communications, Control and Computing*, Urbana IL, 28-30 September 1988. (also *Johns Hopkins Univ.*, JHU-88/16.)

[Aup-88] L. J. Aupperle, "An Algorithm for Covering Polygons with Squares", Draft, *Dept. of Comp. Sci., Princeton Univ.*, Oct. 1988.

[CDR-86] S. Cook, C. Dwork and R. Reischuk, "Upper and Lower Time Bounds for Parallel Random Access Machines without simultaneous writes", *SIAM Jour. of Computing*, vol. 15, no. 1, pp. 87-97, Feb. 1986.

[CoR-87] H.E.Conn and J.O'Rourke, "Some Restricted Rectangle Covering Problems", *Dept. of Comp. Sc., Johns Hopkins Univ.*, Tech. Rep. JHU-87/13, June 1987.

[CoV-88a] R.Cole and U.Vishkin, "Faster Optimal Parallel Prefix Sums and List Ranking", *Courant Institute* CS-TR-277, 1988.

[CoV-88b] R.Cole and U.Vishkin, "Approximate Parallel Scheduling, Part I: the basic technique, with applications to Optimal Parallel List Ranking in Logarithmic Time", *SIAM Jour. of Computing*, vol. 17, No. 1, pp. 128-142, Feb. 1988.

[**CuR-88**] J. C. Culberson and R. A. Reckhow, "Covering Polygons is Hard", *Proc. 29th Symp. on Foundations of Computer Science*, pp. 601-611, Oct. 1988.

[**DeS-84**] E.Dekel and S.Sahni, "A Parallel Matching Algorithm for Convex Bipartite Graphs and Applications to Scheduling", *Jour. of Parallel and Distributed Computing*, vol. 1, pp. 185-205, 1984.

[**FrK-84**] D.S.Franzblau and D.J.Kleitman, "An Algorithm for Constructing Regions with Rectangles: Independence and Minimum Generating Sets for Collections of Intervals", *Proc. 16th Ann. ACM Symp. on Theory of Computing*, pp. 167-174, 1984.

[**FSS-84**] L.Ferrari, P.V.Sankar and J.Sklansky, "Minimal Rectangular Partitions of Digitized Blobs", *Computer Vision, Graphics and Image Processing*, vol. 28, pp. 58-71, Oct. 1984.

[**Gav-72**] F. Gavril, "Algorithms for Minimum Coloring, Maximum Clique, Minimum Covering by Cliques and Maximum Independent Set of a Chordal Graph", *SIAM Jour. of Computing*, vol. 1, no. 2, pp. 180-187, June 1972.

[**KRS-85**] C.P.Kruskal, L.Rudolph and M.Snir, "The Power of Parallel Prefix", *IEEE Trans. on Computers*, vol. C-34, no. 10, pp. 965-968, Oct. 1985.

[**Mas-79**] W. J. Masek, "Some NP-complete Set Covering Problems", unpublished manuscript MIT, 1979.

[**Sam-84**] H. Samet, "The Quadtree and Related Hierarchical Data Structures", *ACM Computing Surveys*, vol. 16, pp. 187-260, 1984.

[**ScI-86**] D.S.Scott and S.S.Iyengar, "TID : a Translation Invariant Data Structure for Storing Images", *Comm. of the ACM*, vol. 29, no. 5, pp. 418-429, May 1986.

[**TaP-75**] S. L. Tanimoto and T. Pavlidis, "A Hierarchical Data Structure for Picture Processing", *Computer Graphics and Image Processing*, vol. 4, pp. 104-119, 1975.