11-1-1987

A New Method of Image Compression Using Irreducible Covers of Maximal Rectangles

Y. Cheng
Louisiana State University

S. S. G. Iyengar
Louisiana State University

R. L. Kashyap
Purdue University

A New Method of Image Compression Using Irreducible Covers of Maximal Rectangles

Y. Cheng
S. S. Iyengar
R. L. Kashyap

TR-EE 87-44
November 1987

School of Electrical Engineering
Purdue University
West Lafayette, Indiana 47907

A New Method of Image Compression Using Irreducible Covers of Maximal Rectangles

Y. Cheng[*]

Department of Mathematics

Louisiana State University

Baton Rouge, LA 70803

S. S. Iyengar

Department of Computer Science

Louisiana State University

Baton Rouge, LA 70803

R. L. Kashyap[†]

Department of Electrical Engineering

Purdue University

West Lafayette, IN 47907

[*]Partial support for the research of this author was provided by a grant from the Louisiana Education Quality Support Fund.
[†]Partial support for the research of this author was provided by the National Science Foundation and Office of Naval Research.

A New Method of Image Compression Using Irreducible Covers of Maximal Rectangles

Y. Cheng, S. S. Iyengar and R. L. Kashyap

Abstract

In recent years there has been a tremendous spurt in research and activity in finding efficient compression techniques for image processing applications. Particularly when an image is structured over a non-rectangular region it is always advantageous to define a method of covering a region by minimal numbers of maximal rectangles. Towards this objective, we analyze the binary image compression problem using irreducible cover of maximal rectangles. We also give a bound on the minimum rectangular cover problem for image compression under certain conditions that previously have not been analyzed. It is demonstrated for a simply connected image that, the irreducible cover proposed here uses less than four times the number of the rectangles in a minimum cover. With n pixels in a square, the parallel algorithm of obtaining the irreducible cover presented in the paper uses $(n/\log n)$ *concurrent-read-exclusive-write* (CREW) processors in $O(\log n)$ time.

Key words and phrases: Image compression, maximal rectangles, covering algorithms

A New Method of Image Compression Using Irreducible Covers of Maximal Rectangles

Y. Cheng, S. S. Iyengar and R. L. Kashyap

1. Introduction

Effective methods of representation of binary digital images are required in many image processing tasks. Currently hierarchical representations like quadtrees and octtrees are very popular [8,9]. One criterion of evaluation of different representations is the degree of information compression achieved by the scheme. The information contained in any representation can be measured by the length of the program needed to transmit the same. For instance, in quadtrees, one needs to transmit the program corresponding to the quadtree including declarations of the leaf nodes which correspond to the pixels or groups of pixels having a 'one'. It is well-known that the *minimum* code length required for transmitting an $n \times n$ binary image is $2\log_2 n$. The image compression efficiency associated with a particular representation can be measured by the ratio of the length of the program to the above minimum, namely $2\log_2 n$. Typically, the ratio is greater than one. The closer the ratio is to one, the greater will be the degree of image compression achieved.

The quadtree corresponds to dividing an image into *non-overlapping* squares by particular tree scanning procedure. In this paper, we explore the possibility of describing each connected part of an image by means of irreducible and maximal rectangles which may be overlapping. A rectangle is described by a quadruple, namely the sizes of the two sizes and two coordinates of some specific corner (say northwest). The image will be described by an *unordered* set of the quadruples corresponding to the various rectangles. We believe that the compression achieved by schemes like this are, in general, superior to those obtained by quadtrees since (i) they do not involve any additional algorithms like tree traversal or ordering, and (ii) the basic unit in quadtrees are squares, not rectangles. These schemes may also be useful for real time dynamization, i.e., dynamically altering the representation in real time as the image changes in real time. However, we do not explore this aspect in this paper.

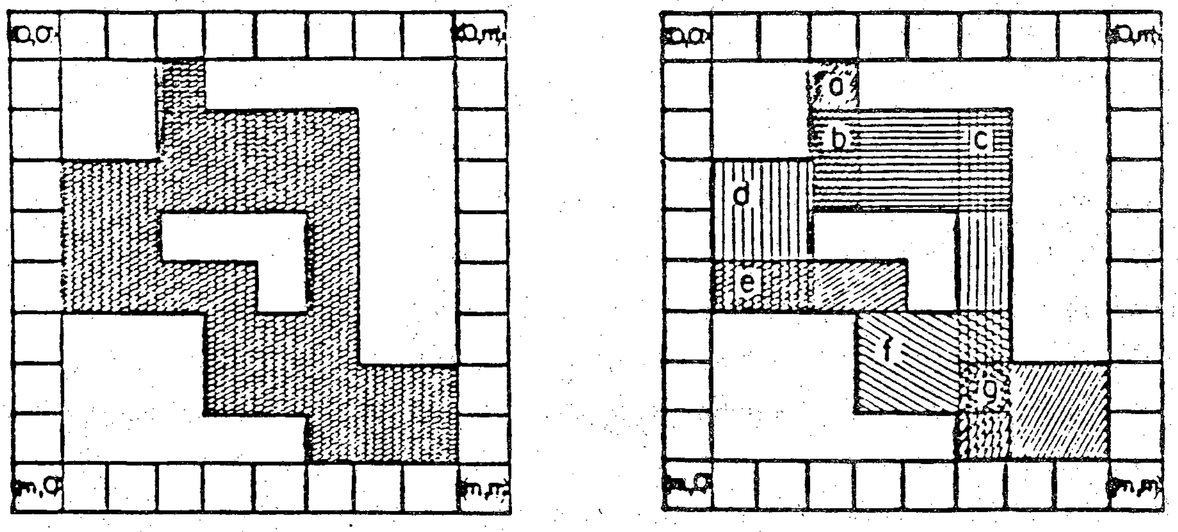
Ferrari et al. [3] considered the representation of images via a partition with (non-overlapping) maximal rectangles. Moitra et al. [6] used maximal irreducible cover with squares; the squares being possibly overlapping. However, Masek showed that the constriction of minimal covers with rectangles is an NP-complete problem. For a broader treatment on this see [2]. In this paper, we plan to describe an image by an irreducible cover made up of maximal rectangles. We also present an algorithm to find an irreducible cover. With n pixels in a square, the parallel implemention of the algorithm can be executed with $(n/\log n)$ *concurrent-read-exclusive-write* (CREW) processors in $O(\log n)$ time. Hence the parallel algorithm is optimal.

It is important to point out that the cover generated is irreducible, but not minimal. The usefulness of the representation is intimately connected to the question of the ratio of the number of rectangles in the irreducible cover of this paper to the number of rectangles in a *minimal* cover. The smaller the ratio, the greater will be the usefulness of the representation. We show that the number of rectangles in the irreducible cover is less than four times the number of rectangles in a minimal cover.

The remainder of this paper is organized as follows: Section 2 describes some basic definitions and the main focus of our problem. Section 3 describes the greedy algorithm and there we develop motivation for the proposed method. Section 4 describes an overview of the proposed algorithm with a detailed proof to show that our method produces an irreducible cover for the image. Section 5 describes a parallel version of the algorithm. Section 6 discusses the number of rectangles used in the irrducible cover and that of a minimum cover under some restricted conditions. Section 7 concludes the paper.

2. Preliminaries

In this paper, we consider a binary image as an array $P[0..m, 0..m]$ of binary valued pixels, where $m = \sqrt{n} + 1$. For convenience, we assume that the image is only within $P[1..\sqrt{n}, 1..\sqrt{n}]$. The value of a pixel $P_{i,j}$ is represented as both *true/false* or its synonymous value *black/white*. In Figure 1, we give an example of an image and an irreducible cover of rectangles for it.



Image

Cover with maximal rectangles a, b, c, d, e, f, g.

Figure 1.

A rectangle can be represented as `rect<row,column,size1,size2>`, where `row` and `column` are the coordinates of the northwest corner pixel of the rectangle, and `size1` and `size2` are the numbers of rows and columns in the rectangle. A black rectangle is *maximal* if it is not contained in any other black rectangle. In this paper, a rectangle always means a black rectangle. A collection C of rectangles is called a *cover* of the image if every black pixel is contained in at least one of the rectangles in C. A cover C is irreducible if no proper subset of C is a cover of the image. A greedy algorithm to obtain an irreducible cover from a cover will be given in section 3.

Neighborhood characterization of black pixels. A black pixel $P_{i,j}$ is a *top* (respectively, *bottom*, *left* or *right*) pixel if the pixel $P_{i-1,j}$ (respectively, $P_{i+1,j}$, $P_{i,j-1}$, $P_{i,j+1}$,) is not black. It is easy to show that a rectangle is maximal if and only if it contains top, bottom, left and right pixels. A column of black pixels is called a *maximal column* if it is not contained in any other column of black pixels. Hence, a rectangle is a maximal column if and only it contains only one column and it contains a top pixel and a bottom pixel. For example, in Figure 1, the pixels $P_{6,5}$ and $P_{7,5}$ form a maximal column and pixels $P_{3,2}, P_{4,2}, P_{5,2}$ form another one. Of course, the maximal column is uniquely determined by its top pixel. If the top pixel is $P_{i,j}$, we write the maximal column as max_col<i,j>. Hence the two maximal columns in Figure 1 we just mentioned are denoted as max_col<6,5> and max_col<3,2>, respectively. The notation max_col<i,j> is defined only when $P_{i,j}$ is a top pixel. Similarly, we can define maximal rows. A set of black pixels is said to be *covered* by a collection of rectangles if every pixel in the set is contained in at least one rectangle of this collection. A sequence of consecutive top (respectively, bottom, left, right) pixels is called a *top* (respectively, *bottom*, *left*, *right*) *edge*. Figure 2 illustrates these terms.

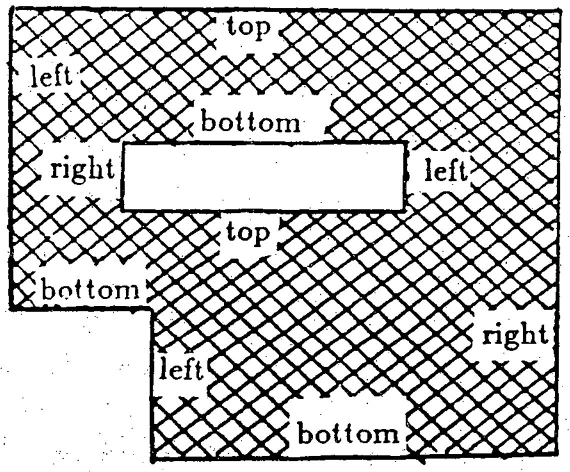


Figure 2.

A cover with minimum number of rectangles is called a *minimum cover*. Clearly, every minimum cover is irreducible. The vice versa is not true, an irreducible cover need not be minimal. Also, since every rectangle is contained in at least one maximal black rectangle, we can obtain a minimum cover with maximal rectangles from any minimum cover. An ideal way to store binary image data is to use a minimum cover. However, it is a difficult problem to find a minimum cover for an image. The problem that whether an image has a minimum cover with k rectangles is known to be NP-complete (Masek's unpublished work cited in [2]). Therefore, it is reasonable to use irreducible covers instead of minimum covers. In this paper, we present an algorithm to find an irreducible cover. For a simply connected image, i.e., a conected image without holes, we show that the irreducible covers uses less than four times the number of rectangles in a minimum cover.

Lemma 1. *Every maximal column is contained in a unique maximal rectangle.*

Proof. Let max_col<t,j> be a maximal column with m rows with $P_{t,j}$ as its top pixel and $P_{b,j}$ its a bottom pixel with $b = t + m - 1$ as in Figure 3. Now let l be the smallest integer such that P_{k_1,k_2} are black pixels for all $t \leq k_1 \leq b$ and $l \leq k_2 \leq j$. There exists an integer p such that $t \leq p \leq b$ and $P_{p,l-1}$ is not a black pixel. Hence $P_{p,l}$ is a left pixel as in Figure 3. Let r be the largest integer such that P_{k_1,k_2} are black pixels for all $t \leq k_1 \leq b$ and $j \leq k_2 \leq r$. There exists an integer q such that $t \leq q \leq b$ and $P_{q,r+1}$ is not a black pixel. Hence $P_{q,r}$ is a right pixel. Pixels P_{k_1,k_2}, $t \leq k_1 \leq b$, $l \leq k_2 \leq r$ form the maximal rectangle rect<t,l,m,$r-l+1$>. It contains the maximal column max_col<t,j>. Since this is essentially the unique way to construct a maximal rectangle which contains max_col<t,j>, the maximal rectangle we obtained is the unique one. containing max_col<t,j>. *Q.E.D.*

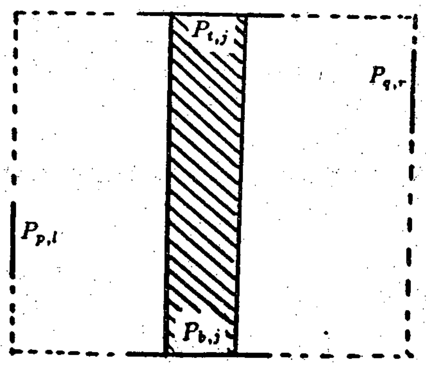


Figure 3. Illustration for the proof of Lemma 1.

The unique maximal rectangle containing `max_col<`t,j`>` is denoted by $\Re_{t,j}$. We note that different maximal columns may be contained in the same maximal rectangle. For example, in Figure 1, $\Re_{2,4} = \Re_{2,5} =$ `rect<2,3,2,4>`.

Finally, we note that, for sets A and B, we denote the set of elements of A which is not in B by $A \setminus B$.

3. Greedy Algorithm

The greedy algorithm can be easily described as follows. We begin with a cover $\mathcal{C}$ of rectangles. When we find one rectangle $\Re$ which is covered by $\mathcal{C} \setminus \{\Re\}$, we delete this rectangle from the cover. That is, $\mathcal{C} \leftarrow \mathcal{C} \setminus \{\Re\}$. We do this process until we cannot find any one in the updated cover $\mathcal{C}$, which satisfies the above condition. Each time when we delete one rectangle, we know that the updated cover is a true *cover*. Hence, at the end, the set of rectangles left is also a cover. It is an irreducible cover because this is the reason that we stop the process.

Greedy Algorithm

input: A cover $\mathcal{C}$ of rectangles for a binary image.
output: A subset $\mathcal{C}_1$ of $\mathcal{C}$ which forms an irreducible cover for the image.

```
$\mathcal{C}_1 \leftarrow \mathcal{C}$
while (there exists $M \in \mathcal{C}_1$ which is covered by $\mathcal{C}_1 \setminus \{M\}$)
    $\mathcal{C}_1 \leftarrow \mathcal{C}_1 \setminus \{M\}$
end;
```

This above algorithm is sequential in nature since we can delete one rectangle at a time. In the next section, we shall outline a cover so that we can perform the deletion concurrently.

4. New Algorithm

We sketch an outline of our algorithm for finding an irreducible cover for any image.

Algorithm A

input: binary image $P_{i,j}, 1 \leq i,j < m$.

output: an irreducible cover of maximal rectangles for the image.

1. *Determine all maximal rectangles which contain some maximal columns.* This collection of maximal rectangles is denoted by $\mathcal{C}$. Here, for a given maximal column, we find the unique maximal rectangle which contains this column by the method given in the proof of Lemma 1. We note that we may get the same rectangle from different columns.
2. *Eliminate repetitions of the maximal rectangles obtained in the previous step.* After this elimination, every rectangle of $\mathcal{C}$ is uniquely determined by one particular maximal column, or equivalently, by one particular top pixel.
3. *Determine those rectangles $\Re$ of $\mathcal{C}$ whose corresponding maximal column are covered by $\mathcal{C} \setminus \{\Re\}$.* This collection of maximal rectangles is denoted by $\mathcal{D}$.
4. *Finally, $\mathcal{C} \setminus \mathcal{D}$, the collection of rectangles in $\mathcal{C}$ but not in $\mathcal{D}$, is an irreducible cover.*

The rest of this section is devoted to the proof that $\mathcal{C} \setminus \mathcal{D}$ is an irreducible cover.

Lemma 2. *$\mathcal{C}$ is a cover of the image.*

Proof. Let $P_{i,j}$ be a black pixel. Let t be the smallest integer such that $P_{k,j}$ are black pixels for all $t \leq k \leq i$. Since $P_{t-1,j}$ is not black, $P_{t,j}$ is a top pixel. Similarly, let b be the largest integer such that $P_{k,j}$ are black pixels for all $i \leq k \leq b$. Then $P_{t,j}$ is a bottom pixel. Hence pixels $P_{k,j}$, $t \leq k \leq b$, form the maximal column $\texttt{max\_col<}t,j\texttt{>}$, which contains $P_{i,j}$. By Lemma 1, $\texttt{max\_col<}t,j\texttt{>}$ is contained in the unique maximal rectangle $\Re_{t,j} \in \mathcal{C}$. This proves that $\mathcal{C}$ is a cover for the image. *Q.E.D.*

In the next lemma, we prove that if the maximal rectangle $\Re_{i_2,j_2}$ covers a pixel P_{i,j_1}, which is contained in a maximal column $\texttt{max\_col<}i_1,j_1\texttt{>}$, then $\Re_{i_2,j_2}$ covers the row of $\Re_{i_1,j_1}$ containing P_{i,j_1}. Figure 4 illustrates this result.

Lemma 3. *Let* $\Re_{i_1,j_1} = \texttt{rect<}i_1, l_1, r_1, c_1\texttt{>}$ *and* $\Re_{i_2,j_2} = \texttt{rect<}i_2, l_2, r_2, c_2\texttt{>}$. *Suppose that* $P_{i,j_1} \in \Re_{i_2,j_2} \cap \texttt{max\_col<}i_1, j_1\texttt{>}$. *Then* $r_2 \leq r_1$ *and* $P_{i,k} \in \Re_{i_2,j_2}$ *for all* k *with* $l_1 \leq k \leq l_1 + c_1 - 1$.

Proof. Suppose that $i_2 < i_1$. Then $\Re_{i_2,j_2}$ contains the pixel P_{i_1-1,j_1}. This pixel is not black since P_{i_1,j_1} is a top pixel. However, $\Re_{i_2,j_2}$ contains black pixels only. Hence $i_2 \geq i_1$. Similarly, since $P_{i_1+r_1-1,j_1}$ is a bottom pixel, we have that $i_2 + r_2 - 1 \leq i_1 + r_1 - 1$. Hence $r_2 \leq r_1$. Also, $P_{i_2,k} \in \Re_{i_2,j_2}$ for all k with $l_2 \leq k \leq l_2 + c_2 - 1$. This implies that $l_2 \leq l_1$ and $l_2 + c_2 - 1 \geq l_1 + c_1 - 1$. Therefore, $c_1 \leq c_2$ and $P_{i,k} \in \Re_{i_2,j_2}$ for all k with $l_1 \leq k \leq l_1 + c_1 - 1$. *Q.E.D.*

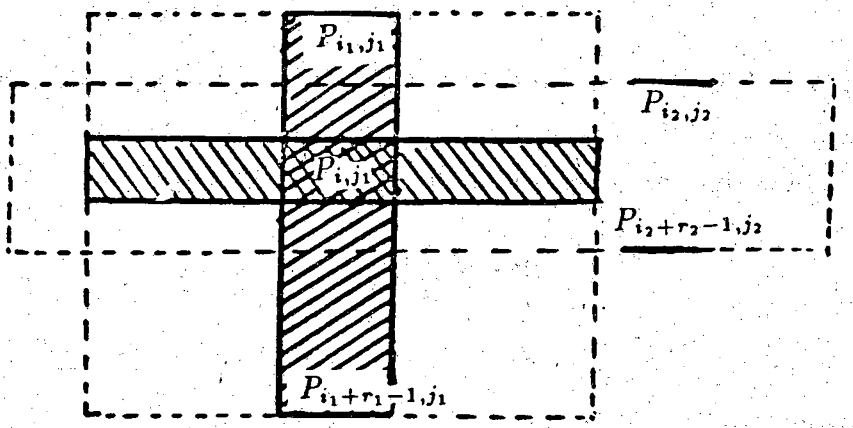


Figure 4. Illustration for the proof of Lemma 3.

Since the sequential and parallel versions to implement step 2 of algorithm A are essentially different, we will simply assume, in this section, that we choose one particular $\texttt{max\_col<}i,j\texttt{>}$ in each rectangle $\Re$ of $\mathcal{C}$ and call the top pixel $P_{i,j}$ an *active* pixel. Hence

$$\mathcal{C} = \{ \Re_{i,j} \mid P_{i,j} \text{ is active} \}$$

and if P_{i_1,j_1} and P_{i_2,j_2} are distinct active pixels, then $\Re_{i_1,j_1} \neq \Re_{i_2,j_2}$. By definition,

$$\mathcal{D} = \{ \Re_{i,j} \mid P_{i,j} \text{ is active and } \texttt{max\_col} <i,j> \text{ is covered by } \mathcal{C} \setminus \{\Re_{i,j}\} \}. \quad (4.1)$$

Lemma 4. *For an active $P_{i,j}$, $\Re_{i,j} = \texttt{rect<}i,l,r,c\texttt{>}$ is covered by $\mathcal{C} \setminus \{\Re_{i,j}\}$ if and only if $\texttt{max\_col<}i,j\texttt{>}$ is covered by $\mathcal{C} \setminus \{\Re_{i,j}\}$.*

Proof. Suppose that $\texttt{max\_col<}i,j\texttt{>}$ is covered by $\mathcal{C} \setminus \{\Re_{i,j}\}$. For fixed i_1 with $i \leq i_1 \leq i+r-1$, $P_{i_1,j} \in \texttt{max\_col<}i,j\texttt{>}$. Hence there exists $\Re_{i_2,j_2} \neq \Re_{i,j}$ such that $P_{i_1,j_1} \in \Re_{i_2,j_2}$. By Lemma 3, $\Re_{i_2,j_2}$ contains $P_{i_1,k}$ for all k with $l \leq k \leq l+c-1$. This proves that $\Re_{i,j} = \texttt{rect<}i,l,r,c\texttt{>}$ is covered by $\mathcal{C} \setminus \{\Re_{i,j}\}$. The other part of the lemma is trivial. *Q.E.D.*

Now we can prove our main result in this section that $\mathcal{C} \setminus \mathcal{D}$ is an irreducible cover.

Theorem 1. *$\mathcal{C} \setminus \mathcal{D}$ is an irreducible cover of the image.*

Proof. We first prove that $\mathcal{C} \setminus \mathcal{D}$ is a cover. So, let $P_{i,j}$ be a black pixel. By Lemma 2, $P_{i,j} \in \Re_{i_1,j_1} = \texttt{rect<}i_1,l_1,r_1,c_1\texttt{>}$ for an active P_{i_1,j_1}. Among all these possible $\Re_{i_1,j_1}$, we choose one with minimum r_1. That is, $P_{i,j} \in \Re_{i_1,j_1} = \texttt{rect<}i_1,l_1,r_1,c_1\texttt{>}$ and if $P_{i,j} \in \Re_{i',j'} = \texttt{rect<}i',l',r',c'\texttt{>}$, then $r \leq r'$. Now we claim that $\Re_{i_1,j_1} \in \mathcal{C} \setminus \mathcal{D}$. Suppose it is not so. Then $\Re_{i_1,j_1} \in \mathcal{D}$ and $\texttt{max\_col<}i_1,j_1\texttt{>}$ is covered by $\mathcal{C} \setminus \{\Re_{i_1,j_1}\}$ by (4.1). By Lemma 4, $\Re_{i_1,j_1}$ is covered by $\mathcal{C} \setminus \{\Re_{i_1,j_1}\}$. Since $P_{i,j} \in \Re_{i_1,j_1}$, we have that $i_1 \leq i \leq i_1 + r_1 - 1$. Consider the black pixel P_{i,j_1} which is covered by $\mathcal{C} \setminus \{\Re_{i_1,j_1}\}$. Say, $P_{i,j_1} \in \Re_{i_2,j_2} = \texttt{rect<}i_2,l_2,r_2,c_2\texttt{>}$, with $\Re_{i_2,j_2} \neq \Re_{i_1,j_1}$. By Lemma 3, $r_2 \leq r_1$ and $P_{i,j} \in \Re_{i_2,j_2}$. By the minimality of $\Re_{i_1,j_1}$, $r_2 = r_1$. Now, both $\Re_{i_1,j_1}$ and $\Re_{i_2,j_2}$ contain P_{i,j_1} and have the same number of rows. Hence, they both contain $\texttt{max\_col<}i_1,j_1\texttt{>}$. By Lemma 1, they are the same rectangle, a contradiction. This proves that $P_{i,j}$ is contained in $\Re_{i_1,j_1} \in \mathcal{C} \setminus \mathcal{D}$. Hence $\mathcal{C} \setminus \mathcal{D}$ is a cover.

To prove that $\mathcal{C} \setminus \mathcal{D}$ is an irreducible cover, we have to prove that $\mathcal{C} \setminus (\mathcal{D} \cup \{\Re_{i,j}\})$ is not a cover for every $\Re_{i,j} \notin \mathcal{D}$ with active $P_{i,j}$. Suppose that $\mathcal{C} \setminus (\mathcal{D} \cup \{\Re_{i,j}\})$ is a cover for some $\Re_{i,j} \notin \mathcal{D}$ with active $P_{i,j}$. Then $\mathcal{C} \setminus \{\Re_{i,j}\}$ is also a cover. In particular, $\texttt{max\_col<}i,j\texttt{>}$ is covered by $\mathcal{C} \setminus \{\Re_{i,j}\}$. By definition, $\Re_{i,j} \in \mathcal{D}$, which is a contradiction. This completes the proof of this theorem. *Q.E.D.*

5. Parallel Implementation of Algorithm A

The first step in algorithm A is to determine all maximal rectangles which contain some maximal columns. Parallel algorithms 1 and 2 returns maximal rectangles $\Re_{i,j} \leftarrow$ rect<$top[i,j], left\_bound[i,j], col\_size[i,j], row\_size[i,j]$ >. For an illustration, please see Figure 5.

Algorithm 1

input: binary image $P_{i,j}$, $1 \leq i,j < m$.

output: boundaries $left[i,j]$, $right[i,j]$ of the maximal row containing black pixel $P_{i,j}$. boundaries $top[i,j]$, $bottom[i,j]$ of the maximal column containing black pixel $P_{i,j}$.

```
forall $1 \leq i,j < m$ pardo
    if $P_{i,j}$ then
        begin
        $left[i,j] \leftarrow \min(\{l \mid \bigwedge_{0<l\leq k\leq j} P_{i,k}\})$
        $right[i,j] \leftarrow \max(\{r \mid \bigwedge_{j\leq k\leq r<m} P_{i,k}\})$
        $top[i,j] \leftarrow \min(\{t \mid \bigwedge_{0<t\leq k\leq i} P_{k,j}\})$
        $bottom[i,j] \leftarrow \max(\{b \mid \bigwedge_{i\leq k\leq b<m} P_{k,j}\})$
        end
    od
end;
```

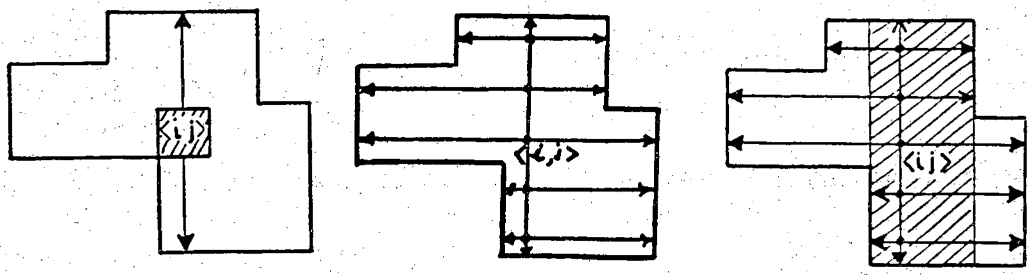


Figure 5. Illustration for Algorithm 1.

Algorithm 2

input: output of Algorithm 1.

output: maximal rectangles $\Re_{i,j}$ and maximal columns `max_col`$<i,j>$ which contains top pixels $P_{i,j}$.

```
forall $1 \leq i,j < m$ pardo
    if ($P_{i,j}$ and not($P_{i-1,j}$)) then
        begin
        $left\_bound[i,j] \leftarrow \max(\{left[k,j] | top[i,j] \leq k \leq bottom[i,j]\})$
        $right\_bound[i,j] \leftarrow \min(\{right[i,k] | top[i,j] \leq k \leq bottom[i,j]\})$
        $col\_size[i,j] \leftarrow bottom[i,j] - top[i,j]$
        $row\_size[i,j] \leftarrow right\_bound[i,j] - left\_bound[i,j]$
        $\Re_{i,j} \leftarrow$ rect<$top[i,j], left\_bound[i,j], col\_size[i,j], row\_size[i,j]$ >
        max_col<$i,j$> $\leftarrow$ rect<$i, j, 1, row\_size[i,j]$ >
        end
    od
end;
```

Algorithm 1 can be executed with n^2 *concurrent read-write* processors in $O(1)$ time. As suggested in Moitra and Moitra [6], it can also be executed with $(n/\log n)$ *concurrent-read-exclusive-write* processors in $O(\log n)$ time as follows. It can be obtained by allocating one processor to every pixel whose row index is a multiple of $\log n$. We describe the method to obtain $right[i,j]$ only. The other three can be obtained similarly. In $\log n$ sequential steps, each processor links each of the (next $\log n$) pixels to the rightmost one which either terminates a horizontal sequence of black pixels and/or is $\log n$ columns away. Then in at most $\log n$ parallel steps, all the processors find the right end of horizontal strips which are wider than $\log n$ columns. Finally, in a sequential $\log n$ steps, each processor links each of the (next at most $\log n$) pixels which belongs to a sequence of black pixels wider than $\log n$ columns, to the rightmost one which terminates the sequence.

Similar arguments show that the comparisons in algorithm 2 can be executed in $\log n$ time with $(n/\log n)$ CREW processors.

The purpose of the next algorithm is to implement the second step in Algorithm A.

Algorithm 3

input: output of Algorithm 2.
output: activities, $A_{i,j}$, of top pixels $P_{i,j}$.
(Explained below).

There are two methods to achieve the goal. In the first method, we use array boolean variables $B[i_1, j_1, v_1, v_2], 1 \leq i, j, v_1, v_2 \leq m$, and the CREW model. All top pixels $_{i,j}$ concurrently attemp to write $B[i_1, j_1, v_1, v_2]$, where $\Re_{i,j} = B[i_1, j_1, v_1, v_2]$. Those asses to $B[i_1, j_1, v_1, v_2]$ successfully can get $A_{i,j} \leftarrow 1$. Otherwise, $A_{i,j} \leftarrow 0$. This can be done in $\log n$ time by using $n/\log n$ processors.

The second method to achieve the goal in Algorithm 3 can be described as follows. We first define a linear ordering on the rectangles we obtained in algorithms 1 and 2 as follows. Let $\Re_{i,j} = \texttt{rect<}v_1, v_2, r_1, c_1\texttt{>}$ and $\Re_{i',j'} = \texttt{rect<}v_1', v_2', r_1', c_1'\texttt{>}$. Define $\Re_{i,j} < \Re_{i',j'}$ if and only if

$$(v_1, v_2, r_1, c_1, i, j) < (v_1', v_2', r_1', c_1', i', j')$$

in the lexicographical order. Now we can apply the optimal *random* sorting algorithm of Reif [7], which can be executed in $O(\log n)$ time using $(n/\log n)$ P-RAM processors, to eliminate repetitions among the rectangles obtained in algorithms 1 and 2. That is, among those rectangles $\Re_{i,j}$ with the same $\texttt{rect<}v_1, v_2, r_1, c_1\texttt{>}$, we choose the one with the smallest j. (All i are the same for the same rectangles). For this choice, we say that the pixel $P_{i,j}$ is *active*. Or say $A_{i,j} = 1$ if $P_{i,j}$ is active and 0 otherwise. Therefore, every rectangle obtained in algorithms 1 and 2 is uniquely determined by an active pixel $P_{i,j}$. We can also apply Leighton's deterministic method [5], to achieve the goal by using n processors in time $\log n$. This apparently uses more processes.

In algorithm 4, we assign a sign $sign[i', j']$ to each pixel so that it is 1 if $P_{i',j'} \in (\Re_{i,j} \setminus \texttt{max\_col<}i, j\texttt{>})$ for some active $P_{i,j}$. Now, for every active pixel $P_{i,j}$, we have that $sign[i', j'] = 1$ for all $P_{i',j'} \in \texttt{max\_col<}i, j\texttt{>}$ if and only if $\texttt{max\_col<}i, j\texttt{>}$ is covered by $\mathcal{C} \setminus \{\Re_{i,j}\}$. Equivalently, $\Re_{i,j} \in \mathcal{D}$. For this $P_{i,j}$, we change its activity $A_{i,j}$ from 1 to 0 in algorithm 5. Finally, those $P_{i,j}$ with $A_{i,j} = 1$ form the irreducible cover $\mathcal{C} \setminus \mathcal{D}$ according to Theorem 1.

Algorithm 4

input: output of algorithms 2 and 3.

output: *sign*$[i,j]$ for active $P_{i,j}$. The meaning of *sign*$[i,j]$ is explained above.

```
forall $1 \leq i,j < m$ pardo
    $sign[i,j] = 0$
    od
forall $1 \leq i,j < m$ pardo
    if $A_{i,j}$ then
        $sign[i',j'] \leftarrow 1$ for all $P_{i',j'} \in \Re_{i,j} \setminus$ max_col<$i,j$>
    od
end;
```

Algorithm 5

input: output of algorithms 2, 3 and 4.

output: change some $A_{i,j}$ from 1 to 0. Those $\Re_{i,j}$ with $A_{i,j} = 1$ form the irreducible cover $\mathcal{C} \setminus \mathcal{D}$.

```
forall $1 \leq i,j < m$ pardo
    if ($A_{i,j}$ and $\bigwedge\{sign[i',j'] \mid P_{i',j'} \in$ max_col<$i,j$>$\}$) then
        $A_{i,j} \leftarrow 0$
    od
end;
```

To sketch the sequential algorithm, we first assume that all edges of the image are stored. For a fixed top edge and a fixed bottom, we find all bottom edges under this top edge. Now we fix one of these bottom edge and consider the pair of the top edge and the bottom edge. For a pair of top edge and bottom edge, we construct a maximal rectangle as in Lemma 1. To eliminate repetitions, we simply check all maximal rectangles. This can be done in $O(k^2)$ time, where k is the number of rectangles. To determine those rectangles in $\mathcal{D}$, we can also fix one rectangle and check all other rectangles in $\mathcal{C}$. Again, this can be done in $O(k^2)$ time.

6. Geometry of Binary Images

In this section, we shall state an upper bound on the cardinality of $\mathcal{C}$, $\#\mathcal{C}$, which is the number of rectangles in the cover obtained in Algorithm A. This bound is also an upper bound for the cardinality of the irreducible cover $\mathcal{C} \setminus \mathcal{D}$. After presenting a formula for the number of convex and concave corners of a simply connected binary image, we shall prove that $\#\mathcal{C}$ is at most $4 \cdot \#\mathcal{M} - 3$ for any minimal cover $\mathcal{M}$.

In order to obtain those bounds we mentioned above, we shall introduce some concepts in the geometry of binary images. Although these concepts are pretty much well known in the field of geometry, the authors know no references for our particular need in the study of binary images.

We first define convex and concave corners. Each pixel has four corners. Each corner of a black pixel $P_{i,j}$ has 3 neighbor pixels. The two neighbor pixels which share common edges with $P_{i,j}$ are called *edge neighbors* of the corner and the other one the *vertex neighbor* of the corner. Figure 6 illustrates the definitions.

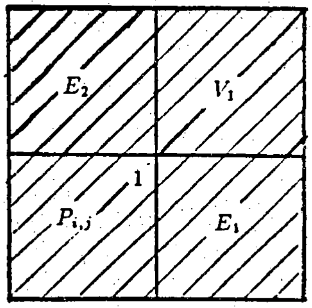


Figure 6. E_1 and E_2 are edge neighbors of corner 1 in Pixel $P_{i,j}$.
V_1 is the vertex neighbor of corner 1 in Pixel $P_{i,j}$.

A corner of black pixel $P_{i,j}$ is called a *convex corner* if both its edge neighbors are white pixels. It is called a *concave corner* if both its edge neighbors are black pixels and its vertex neighbor is a white pixel. Figure 7 illustrates the definitions.

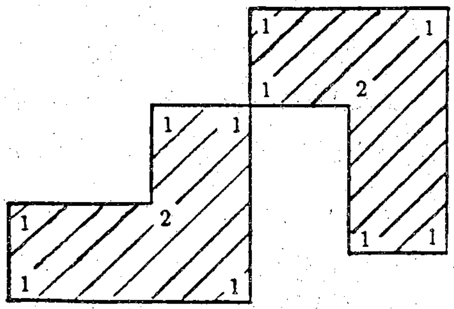


Figure 7. The image has 10 convex corners, labeled 1, and 2 concave corners labeled 2.

Let

$$P_{i_1,j_1}, P_{i_2,j_2}, \ldots, P_{i_t,j_t} \tag{6.1}$$

be a sequence of black pixels. It is a *path* if P_{i_k,j_k} and $P_{i_{k+1},j_{k+1}}$ share a common edge, for all $1 \leq k \leq t-1$. A path (6.1) is *simple* if all the black pixels are distinct. The path is a *cycle* if $P_{i_1,j_1} = P_{i_t,j_t}$. It is a *simple cycle* if the black pixels in (6.1) are distinct, except for $P_{i_1,j_1} = P_{i_t,j_t}$, which are identical pixels. We can use a simple cycle to partition the whole set of pixels into three parts: the cycle itself, and pixels inside the cycle and pixels outside the cycle, called the inner and outer part, respectively. An example to illustrate the concept is given in Figure 8.

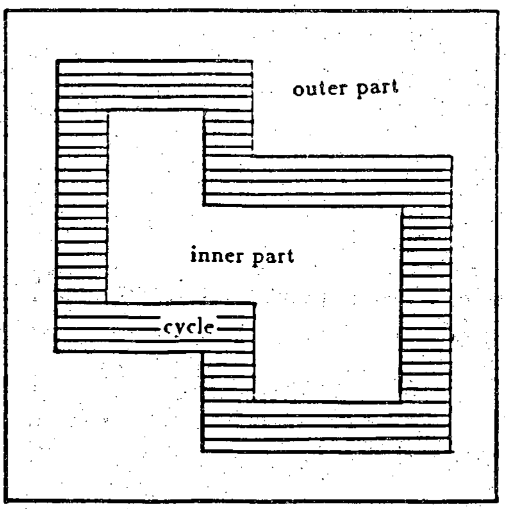


Figure 8. Inner and outer parts of a cycle.

An image is said to be *connected* if there is a path connecting any two black pixels. A *connected component* is a subset of black pixels which is connected, and is maximal subject to the connected condition. Clearly, an image can be partitioned into connected components. An image is connected if and only if it has exactly one connected component. A connected image is said to be *simply connected* if, the inner parts of all simple cycles consist no white pixels.

For the remainder of this section, we let $\alpha(P)$ be the number of convex corners of the image P, $\beta(P)$ the number of concave corners, and $\gamma(P)$ the number of connected components. We also denote $\mathcal{C}(P)$ as the cover of P obtained in Algorithm A.

Theorem 2. *If all connected components of an image P are simply connected, then* $\alpha(P) = \beta(P) + 4\gamma(P)$.

Proof. Let Γ be the collection of images such that all their connected components are simply connected. We use induction on $\#P$, the number of black pixels in P, to prove the statement that

$$\alpha(P) = \beta(P) + 4\gamma(P) \tag{6.2}$$

for $P \in \Gamma$. If $\#P = 1$, then P consists of a single black pixel. In this case, (6.2) is certainly true.

Consider a $P \in \Gamma$, and $\#P > 1$. Let P' be obtained from P by deleting a black rectangle from P such that $P' \in \Gamma$ and P' differ in at least one of the values of α, β and γ. We want to generate systematically all the possible P' obeying the above conditions. We will show that the quantity $[\alpha(\cdot) - \beta(\cdot) - 4\gamma(\cdot)]$ is invariant. Since $[\alpha(\cdot) - \beta(\cdot) - 4\gamma(\cdot)] = 0$ for the single pixel image, we can conclude that

$$\alpha(P) = \beta(P) + 4\gamma(P) \text{ for all } P \in \Gamma$$

Consider a maximal left edge $\Im$ of P.

Then $\Im$ is a maximal column in P. By Lemma 1, $\Im$ is contained in a unique maximal rectangle $\Re$. Let $\aleph$ be the right edge of the rectangle $\Re$. $\aleph$ contains a right pixel. Let $\aleph'$ be the unique maximal column containing $\aleph$. For an illustration of these terms, please see Figure 9.

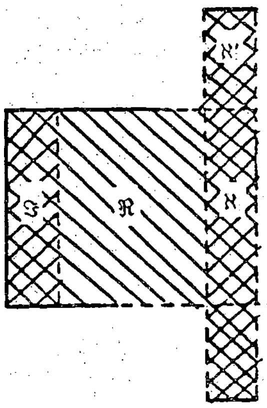
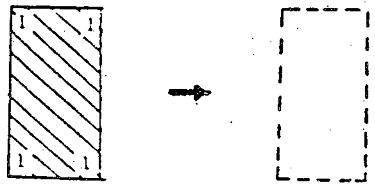
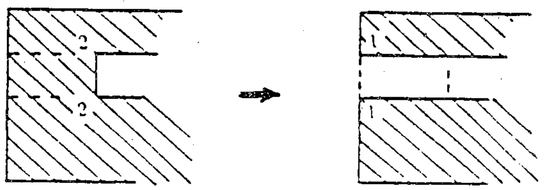


Figure 9. Illustration for the proof of Theorem 2.

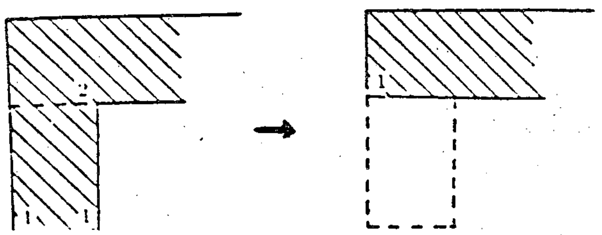
If $\aleph = \aleph'$, then we have three possible situations as shown in Figure 10 (a)-(c). We note that we omit the situation which is a reflection of (c). If $\aleph \neq \aleph'$, then we have two possible situations as shown in Figure 10 (d) and (e). We note that we omit the situation which is a reflection of (d). In each of these situations, we delete a black rectangle, as shown in Figure 10 (a)-(e). It is easy to check that the resulting image P' is also in Γ. Since $\#P' < \#P$, (6.2) holds for P' by induction hypothesis. By using those information we obtained in Figure 10, we can easily check that (6.2) holds for P. This completes the proof of Theorem 2. *Q.E.D.*



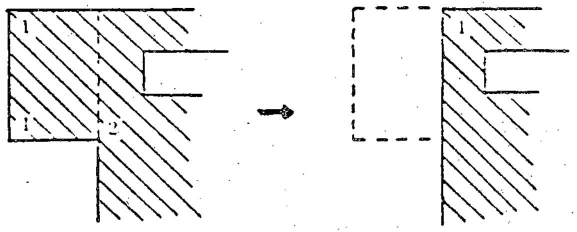
(a) $\alpha(P') = \alpha(P) - 4$, $\beta(P') = \beta(P)$, $\gamma(P') = \gamma(P) - 1$.



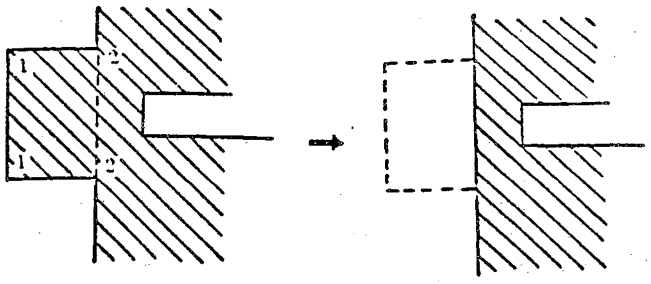
(b) $\alpha(P') = \alpha(P) + 2$, $\beta(P') = \beta(P) - 2$, $\gamma(P') = \gamma(P) + 1$.



) $\alpha(P') = \alpha(P) - 1$, $\beta(P') = \beta(P) - 1$, $\gamma(P') = \gamma(P)$.



(d) $\alpha(P') = \alpha(P) - 1$, $\beta(P') = \beta(P) - 1$, $\gamma(P') = \gamma(P)$



(e) $\alpha(P') = \alpha(P) - 2$, $\beta(P') = \beta(P) - 2$, $\gamma(P') = \gamma(P)$.

Figure 10. Illustration for the proof of Theorem 2.

The proof of the next result is similar to the proof of Theorem 2 and involving more cases.

Theorem 3. *If all connected components of an image P are simply connected, then* $\#\mathcal{C}(P) \leq \beta(P) + \gamma(P)$.

Proof. Let Γ be the collection of images such that all their connected components are simply connected. We use induction on $\#P$, the number of black pixels in P, to prove the statement that

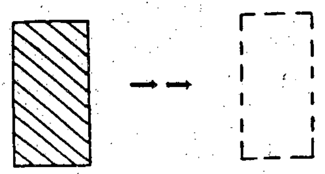
$$\#\mathcal{C}(P) \leq \beta(P) + \gamma(P). \tag{6.3}$$

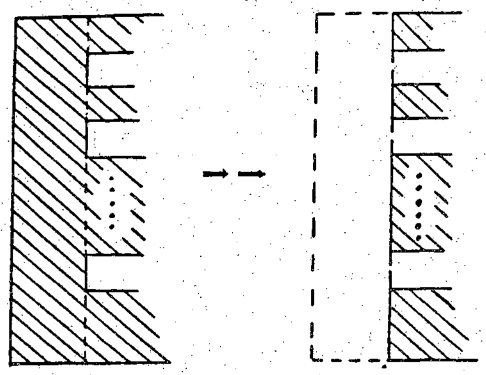
for $P \in \Gamma$. If $\#P = 1$, then P consists of a single black pixel. In this case, (6.3) is certainly true.

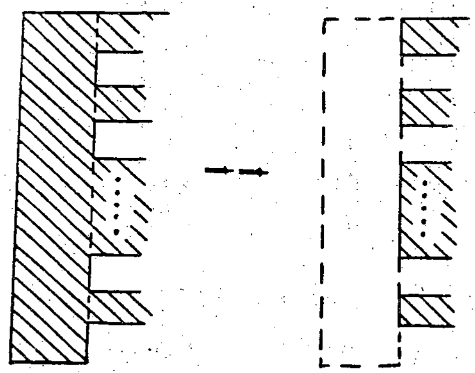
Now we assume that $P \in \Gamma$, and $\#P \neq 0$. As we did in the proof of Theorem 2, we consider a maximal left edge $\Im$ of P. Then $\Im$ is is a maximal column in P. By Lemma 1, $\Im$ is contained in a unique maximal rectangle $\Re$. Let $\aleph$ be the right edge of the rectangle $\Re$. $\aleph$ contains a right pixel. Let $\aleph'$ be the unique maximal column containing $\aleph$. For an illustration of these terms, please see Figure 9.

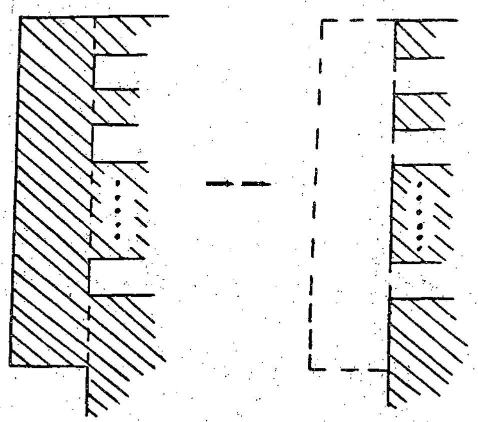
If $\aleph = \aleph'$, then we have seven possible situations as shown in Figure 11 (a)-(g). We note that we omit situations which are reflections of (c),(d) and (f). We also note that these seven situations come from the consideration of the northeast and southeast corners of the rectangle $\Re$. The corners can be convex, concave corners or can be none of the above two types. We assume that there are t right edges of P in $\aleph$.

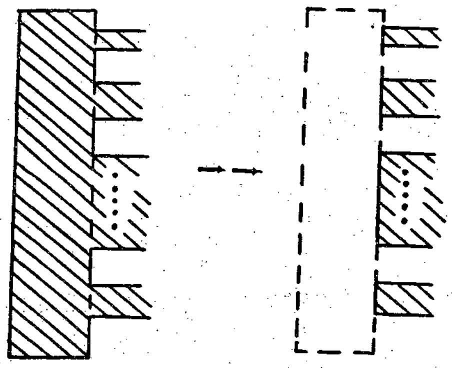
If $\aleph \neq \aleph'$, then we have two possible situations as shown in Figure 11 (h)-(i). We note that we omit the situation which is a reflection of (h). In each of these situations, we delete a black rectangle, as shown in Figure 11 (a)-(i). It is easy to check that the resulting image P' is also in Γ. Since $\#P' < \#P$, (6.3) holds for P' by induction hypothesis. By using those information we obtained in Figure 11, we can easily check that (6.3) holds for P. This completes the proof of Theorem 3. *Q.E.D.*

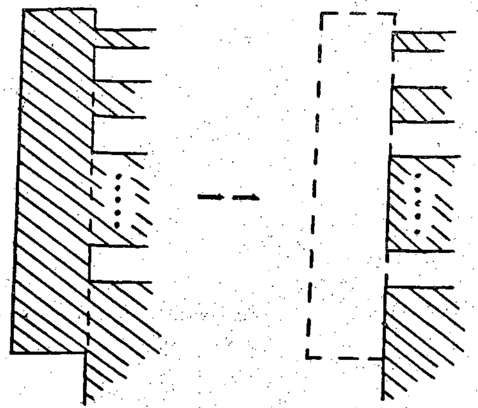

(a) $\#C(P') = \#C(P) - 1$, $\beta(P') = \beta(P)$,
$\gamma(P' = \gamma(P) - 1$.

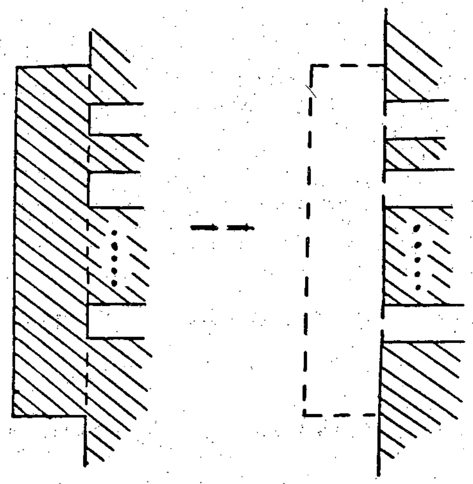

(b) $\#C(P') = \#C(P) - 1$, $\beta(P') = \beta(P) - 2t$,
$\gamma(P' = \gamma(P) + t$.


(c) $\#C(P') = \#C(P) - 1$, $\beta(P') = \beta(P) -2t$,
$\gamma(P' = \gamma(P) + t - 1$.

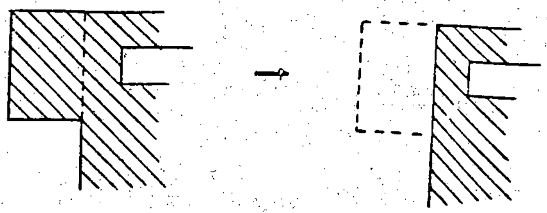

(d) $\#C(P') = \#C(P) - 1$, $\beta(P') = \beta(P) -2t - 1$,
$\gamma(P' = \gamma(P) + t$.


(e) $\#C(P') = \#C(P) - 1$, $\beta(P') = \beta(P) -2t + 2$,
$\gamma(P' = \gamma(P) + t - 2$.

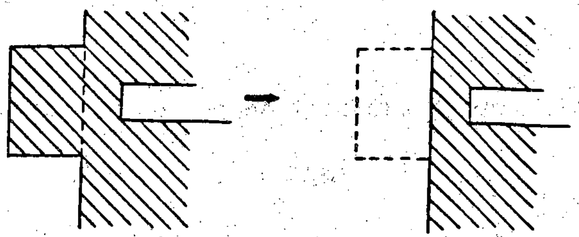

(f) $\#C(P') = \#C(P) - 1$, $\beta(P') = \beta(P) -2t$,
$\gamma(P' = \gamma(P) + t - 1$.



(g) $\#C(P') = \#C(P) - 1$, $\beta(P') = \beta(P) - 2t - 2$, $\gamma(P' = \gamma(P) + t$.



(h) $\#C(P') = \#C(P) - 1$, $\beta(P') = \beta(P) - 1$, $\gamma(P' = \gamma(P)$.



(i) $\#C(P') = \#C(P) - 1$, $\beta(P') = \beta(P) - 2$, $\gamma(P' = \gamma(P)$.

Figure 11. Illustration for the proof of Theorem 3.

Corollary 1. *If all connected components of an image P are simply connected, then $\#C(P) \leq \alpha(P) - 3\gamma(P)$.*

Proof. This result follows from Theorms 2 and 3. *Q.E.D.*

Corollary 2. *If an image P has only one simply connected component, then $\#C(P) \leq \alpha(P) - 3$. In particular, $\#C(P) \leq 4 \cdot M(P) - 3$, for every minimum cover M.*

Proof. The first part follows from Corollary 1 directly, since $\gamma(P) = 1$. To prove the second part, we fix a minimum cover M. Clearly, every convex corner of P is covered by at least one rectangle in M. Also, every rectangle contains at most four convex corners of P. Therefore, $\alpha(P) \leq 4 \cdot \#M$. This proves the corollary. Q.E.D.

This corollary also implies the following result.

Theorem 4. *Suppose that an image P is simply connected and $\mathcal{C} \setminus \mathcal{D}$ is the irreducible cover obtained in* Algorithm A. *Then $\#(\mathcal{C} \setminus \mathcal{D}) \leq 4 \cdot \#\mathcal{M} - 3$, for every minimum cover $\mathcal{M}$.*

7. Conclusions

The search for an optimal covering for a binary image is fundamental to many image processing applications. In this paper, we propose an efficient way of compressing digital image using irreducible covers of maximal rectangles. The principal results are:

1. If all connected components of an image P are simply connected, then the number of convex corners = number of concave corners $+4\cdot$ the number of connected components of the image.
2. For a simply connected image, the cover $\mathcal{C}$ proposed in this paper uses less than four times the number of rectangles in a minimum cover. This bound is also an upper bound for the number of rectangles used in the irreducible cover $\mathcal{C} \setminus \mathcal{D}$.
3. The parallel algorithm of finding the irreducible cover $\mathcal{C} \setminus \mathcal{D}$ uses $(n/\log n)$ *concurrent-read-exclusive-write* (CREW) processors in $O(\log n)$ time.
4. The geometry of binary images described in this paper is very unique in characterizing the mathematical correspondence of minimum cover problem.

References

1. S. Chaiken, D.J. Kleitman, M. Saks and J. Shearer, "Covering Regions by Rectangles," *SIAM J. Alg. Discrete Methods*, vol. 2, pp. 394-410, Dec. 1981.
2. H. E. Conn and J. O'Rourke, "Some Restricted Rectangle Covering Problems," *Department of Computer Science, Johns Hopkins University, Technical Report JHU-87/13*, June, 1987.
3. L. Ferrari, P.V. Sankar and J. Sklansky, "Minimal Rectanglar Partitions of Digitized Blobs," *Computer Vision, Graphics, and Image Processing*, vol 28, pp. 58-71, Oct. 1981.
4. 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 Annual ACM Symposium on Theory of Computing*, pp. 167-174, 1984.
5. T. Leighton, "Tight Bounds on the Complexity of Parallel Sorting," *Proc. 16th Annual ACM Symposium on Theory of Computing*, pp. 71-80, 1984.
6. D. Moitra and A. Moitra, "Irreducible Cover for Binary Images Using Maximal Squares," *unpublished preliminary draft*, August, 1987.
7. J. H. Reif, "An Optimal Parallel Algorithm for Integer Sorting," *IEEE Symposium on Foundations of Computer Science*, pp. 496-503, 1985.
8. H. Samet, "The Quadtrees and Related Hierarchical Data Structure," *ACM Computing Surveys*, vol. 16, N0. 2, June 1984.
9. D. S. Scott and S. S. Iyengar, "TID - a Translation Invariant Data Structure for Storing Images," *Communications of the ACM*, vol. 29, no. 5, pp. 418-429, May 1986.