Concise Papers

# A Strip-Splitting-Based Optimal Algorithm for Decomposing a Query Window into Maximal Quadtree Blocks

Yao-Hong Tsai,
Kuo-Liang Chung, *Senior Member*, *IEEE*, and
Wan-Yu Chen

**Abstract**—Decomposing a query window into maximal quadtree blocks is a fundamental problem in quadtree-based spatial database. Recently, Proietti presented the first optimal algorithm for solving this problem. Given a query window of size $n_1 \times n_2$, Proietti's algorithm takes $O(n_l)$ time, where $n_l = max(n_1, n_2)$. Based on a strip-splitting approach, this paper presents a new optimal algorithm for solving the same problem. Experimental results reveal that our proposed algorithm is quite competitive with Proietti's algorithm.

**Index Terms**—Maximal quadtree blocks, optimal algorithm, spatial database, window queries.

## 1 Introduction

EFFICIENT management of quadtee-based structures is important in many fields such as image databases, geographic information systems, image processing, computer graphics, robotics, and so on [8], [6], [4], [5].

A well-known strategy to perform a window query is first to decompose the corresponding window $W$ into square subwindows, then the window query becomes the integration of the smaller subqueries over smaller subwindows. These subwindows are named maximal quadtree blocks. For convenience, these maximal quadtree blocks are denoted by $B$. In [9], it has been shown that the number of $B$ inside an $n \times n$ $W$ is bounded by $3(2n - \log n) - 5$. It implies that, for an $n_1 \times n_2$ $W$, the number of $B$ inside this $W$ is bounded by $\Theta(n_l)$, where $n_l = \max(n_1, n_2)$. On the other hand, the lower bound for solving this decomposition problem is $\Omega(n_l)$. In [3], a general formula is given for the n-dimensional space.

In 1993, Aref and Samet [1] presented an $O(n_l \log\log T)$-time algorithm for solving this decomposition problem, where $T \times T$ is the size of the queried image. In the past few years, researchers tried to design an optimal algorithm running in $O(n_l)$ time. In 1999, based on a top-down approach, Proietti [7] presented the first $O(n_l)$-time algorithm for solving this decomposition problem. Experimental results reveal that the time performance of Proietti's algorithm is superior to that of Aref and Samet's algorithm considerably.

Based on a strip-splitting approach, this paper presents a new optimal algorithm for solving the same problem. The key concept of the proposed optimal algorithm is quite different from the Proietti's algorithm [7]. Although the time complexity of our proposed algorithm is the same as Proietti's algorithm, the constant factor in the time complexity of our proposed algorithm is half of that of Proietti's algorithm according to a large amount of test data. Experimental results reveal that our proposed algorithm is quite competitive with Proietti's algorithm.

## 2 The Work by Proietti

The quadtree represents a binary image as a set of quadrants. Thus, quadrants correspond to square blocks in the image and they are not overlapped. If the entire image is totally black or white, the image is represented by a single root node; otherwise, it is gray and the image is split into four equal-sized quadrants. The regular decompositions are repeated recursively until all corresponding subquadrants are totally black or white. If a subdivision is either black or white, then its corresponding node is an external (leaf) node; otherwise, it is an internal node. Given a binary image of size $T \times T = 2^N \times 2^N$, the level of the root is $N$ and the four quadrants of the root are at level $N - 1$. Fig. 1 shows an example of a binary image and its quadtree representation.

For performing the window query in a quadtree-based spatial database, a well-known strategy is first to decompose the window into a set of square subwindows according to the quadtree decomposition. Then, the query becomes the integration of those smaller subqueries over those smaller subwindows. The subwindows are the blocks corresponding to the leaf nodes in the quadtree, which represent the window region within the image space. These square subwindows are named maximal quadtree blocks and comprise the set $B$. In the following, a query window will be denoted by $w(x, y, n_1, n_2)$, where $x$ and $y$ represent the $x$ and $y$-coordinates of its upper-left corner, $n_1$ is the height, and $n_2$ is the width of the query window, respectively. For example, we consider the window $W = w(1, 1, 9, 8)$ as shown in Fig. 2, where $W$ is highlighted by four thick lines on its boundary. In total, $W$ can be decomposed into 33 maximal blocks; among these, 24 maximal blocks are of width 1, eight maximal blocks are of width 2, and one maximal block is of width 4.

Each maximal block in $B$ is denoted by $MB(x, y, s)$, where $(x, y)$ is the $x$ and $y$-coordinates of its upper-left corner and $s$ is the width of the maximal block. Therefore, these 33 maximal blocks are denoted by $MB(1, 1, 1)$, $MB(1, 2, 1), \ldots, MB(4, 4, 4) = B_3$, $MB(8, 4, 2) = B_4$, and $MB(8, 6, 2) = B_5$.

We now take an example to briefly review the optimal algorithm proposed by Proietti [7] for solving the decomposition problem. As shown in Fig. 3a, suppose the input image is of size $8 \times 8$ and the query window is $W = w(1, 1, 5, 4)$. According to Lemma 3 in [7] and the given query window $W$, we have $k_1 = 4$ and $k_2 = 4$, where $k_1$ and $k_2$ are the greatest power of two integers such that $k_1 \leq 5$ and $k_2 \leq 4$. Then, we obtain the smallest square $W' = w(0, 0, 16, 16) = (\lfloor\frac{1}{2k_1}\rfloor \times 2k_1, \lfloor\frac{1}{2k_2}\rfloor \times 2k_2, 4 \times k_1, 4 \times k_2)$ that contains $W$. The obtained square $W'$ is depicted in boldface (see the boundary of Fig. 3b). Similarly, if the given query window is $W = w(1, 1, 9, 8)$ in a $2^4 \times 2^4$ image space, the determined smallest square is $W' = w(0, 0, 32, 32)$.

Since $W' \neq W$, the determined square $W'$ is divided into four quadrants, say $B^{(1)}$ (to the west-north direction), $B^{(2)}$ (to the west-south direction), $B^{(3)}$ (to the east-north direction), and $B^{(4)}$ (to the east-south direction). Each $B^{(i)}$ and the query window $W$ must satisfy one of the following three conditions:

1. $B^{(i)} \bigcap W = \emptyset$,
2. $B^{(i)} \bigcap W \subset W$, and
3. $B^{(i)} \bigcap W = B^{(i)}$.

If condition 1 holds, $W'$ doesn't need to be divided into four quadrants further. If condition 2 holds, the quadrant $B^{(i)}$ is divided into four quadrants. If condition 3 holds, the quadrant $B^{(i)}$ is reported as one member of the maximal quadtree blocks B. In [7],

• *The authors are with the Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology, No. 43, Sec. 4, Keelung Rd., Taipei, Taiwan 10672, R.O.C.*
*E-mail: {yhtsai, klchung, wychen}@cs.ntust.edu.tw.*

*Manuscript received 29 June 2001; revised 24 Sept. 2002; accepted 7 Jan. 2003.*
*For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number 114452.*

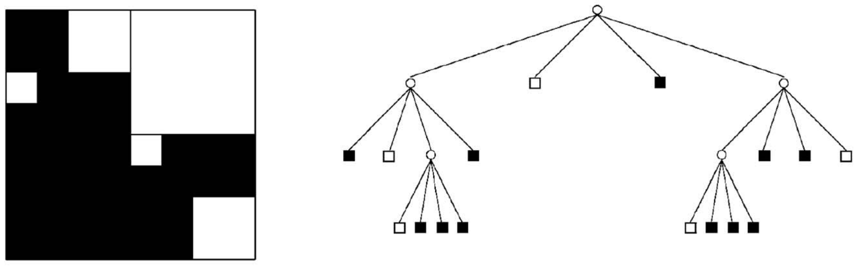

Fig. 1. A $2^3 \times 2^3$ binary image and its corresponding quadtree.

the above division and testing are performed recursively until all the maximal quadtree blocks are obtained.

Returning to Fig. 3b, after dividing $W'$, we have four quadrants, $B^{(1)} = w(0,0,8,8)$, $B^{(2)} = w(7,0,8,8)$, $B^{(3)} = w(0,7,8,8)$, and $B^{(4)} = w(7,7,8,8)$. Among the four quadrants, $B^{(2)}$, $B^{(3)}$, and $B^{(4)}$ satisfy condition 1, so they don't have any intersection with $W$. However, this redundant test is an overhead. Excepting the determination of those 12 $1 \times 1$ maximal quadtree blocks and two $2 \times 2$ (see Fig. 3), all the redundant tests on those 26 white blocks for sizes $8 \times 8$, $2 \times 2$, and $1 \times 1$ are the overhead.

In [7], Proietti has shown that his optimal algorithm takes $O(n_l)$ time when the query window is of size $n_1 \times n_2$, where $n_l = \max(n_1, n_2)$. The motivation of this research is to present a new optimal algorithm without the overhead mentioned above.

## 3 Proposed Algorithm Based on a Strip-Splitting Approach

### 3.1 Strip Splitting Approach

Based on the quadtree decomposition rule, the concept of maximal zones [1] inside a query window is employed to develop our proposed strip-splitting-based optimal algorithm. In what follows, the maximal zones is defined first.

Suppose the query window $w(x_0, y_0, n_1, n_2)$ is considered. Let the query window be divided by $p+1$ vertical lines, $\{x = v_i \text{ for } 0 \leq i \leq p\}$, and $q+1$ horizontal lines, $\{y = h_j \text{ for } 0 \leq j \leq q\}$, where $v_p = x_0 + n_2$ and $h_q = y_0 + n_1$. For convenience, let $F(v_j, 2^{k_i}) = (v_j \bmod 2^{k_i})$. For example, $v_0 = x_0 = 1$ and $k_i = 1$, we have $F(1, 2^1) = 1$. The above $p+1$ vertical lines, for $i = 1, \ldots p$, are defined by $x = v_i = v_{i-1} + 2^{k_i}$ such that $F(v_{i-1}, 2^{k_i}) = 0$, $F(v_{i-1}, 2^{k_i+1}) \neq 0$, and $v_{i-1} + 2^{k_i} \leq v_p$.

For example, the window $W$ in Fig. 2 has five vertical lines and they are $x = v_0 = 1$, $x = v_1 = 2$, $x = v_2 = 4$, $x = v_3 = 8$, and $x = v_4 = 9$ for $k_1 = 0$, $k_2 = 1$, $k_3 = 2$, and $k_4 = 0$. Similarly, the above $q+1$ horizontal lines are given by $y = h_j = h_{j-1} + 2^{k_j}$ such that $F(h_{j-1}, 2^{k_j}) = 0$, $F(h_{j-1}, 2^{k_j+1}) \neq 0$, and $h_{j-1} + 2^{k_j} \leq h_q$. For example, the window $W$ in Fig. 2 has five horizontal lines, $y = h_0 = y_0 = 1$, $y = h_1 = 2$, $y = h_2 = 4$, $y = h_3 = 8$, and $y = h_4 = 10$ for $k_1 = 0$, $k_2 = 1$, $k_3 = 2$, and $k_4 = 1$. It is easy to verify that all the possible values of $k_i$s ($k_j$s) to be checked in determining these $p+1(q+1)$ vertical (horizontal) lines are bounded by $2 \times \lfloor \log n_2 \rfloor (2 \times \lfloor \log n_1 \rfloor)$. On the other hand, there are at most $2 \times \lfloor \log n_2 \rfloor (2 \times \lfloor \log n_1 \rfloor)$ vertical (horizontal) lines to be determined based on the above constraints.

A maximal zone denoted by $MZ(v_i, h_j; 2^{k_{j+1}}, 2^{k_{i+1}})$ for $0 \leq k_j \leq \lfloor \log n_1 \rfloor$ and $0 \leq k_i \leq \lfloor \log n_2 \rfloor$ is the rectangular region between two successive vertical lines, $x = v_i$ and $x = v_i + 2^{k_{i+1}}$, and two successive horizontal lines, $y = h_j$ and $y = h_j + 2^{k_{j+1}}$. In addition, each of the height and width of the maximal zone is a power of two. An example of such a decomposition of the window $W$ is shown in Fig. 4. There are 16 maximal zones in $W$, e.g., the maximal zone $Z_1$ is denoted by $MZ(v_1, h_2; 2^2, 2^1) = MZ(2, 4; 2^2, 2^1)$ which is located at (2,4) and has size $4 \times 2$.

From the definitions of maximal blocks and maximal zones, we have the following property.

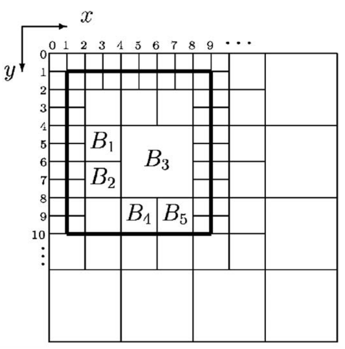

Fig. 2. A query window of size $9 \times 8$ on the binary image of size $2^4 \times 2^4$.

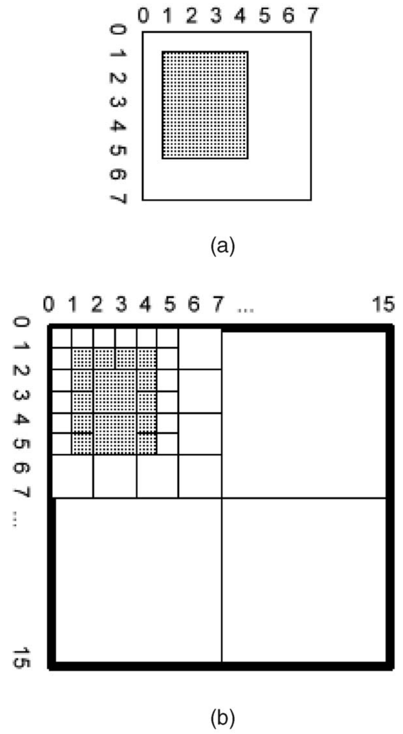

Fig. 3. One example for Proietti's algorithm. (a) The $8 \times 8$ image and the query window $W = w(1, 1, 5, 4)$. (b) The simulation result using Proietti's algorithm.

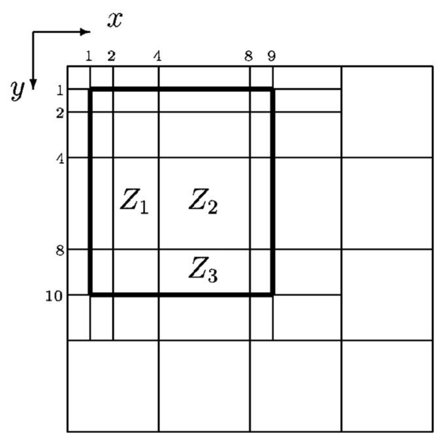

Fig. 4. The maximal zones of the query window $W$ in Fig. 2.

**Property 1: [1].** *Each maximal block in a query window is entirely contained in one and only one maximal zone of that window.*

Return to Fig. 2. Considering maximal blocks $B_1$ and $B_2$, they are in the maximal zone $Z_1$ in Fig. 4. The maximal blocks $B_4$ and $B_5$ are in $Z_3$, while the maximal block $Z_2$ has a single maximal block $B_3$. From Property 1, we see that each maximal block is in one and only one maximal zone. Since each maximal block is a square, we have the following property.

**Property 2.** *A maximal zone of size $2^{m_1} \times 2^{m_2}$ for $m_1 \leq m_2 (m_2 \leq m_1)$ can be decomposed into $2^{m_2-m_1} (2^{m_1-m_2})$ equal-sized square blocks (i.e., maximal blocks) using $2^{m_1} (2^{m_2})$ as the width of those square blocks.*

For example, the maximal zone $Z_1$ of size $4 \times 2$ in Fig. 4 can be decomposed into two maximal blocks, each with size $2 \times 2$, i.e., $B_1$ and $B_2$ (see Fig. 2). Similarly, the maximal zone $Z_3$ of size $2 \times 4$ also has two maximal blocks, each with size $2 \times 2$, i.e., $B_4$ and $B_5$ (see Fig. 2).

**Property 3.** *There are at most two columns (rows) of maximal zones, each maximal zone with the same width (height), at the boundary of the query window. If these columns (rows) of maximal zones are stripped off, that window will become a smaller one, but there are still at most two columns (rows) of maximal zones at the boundary of that smaller window until that window becomes null.*

For example, at the boundary of the query window of Fig. 4, there are two columns of maximal zones, the leftmost column of maximal zones and the rightmost column of maximal zones, where each maximal zone is of width 1. However, there is only one row of maximal zones, the top (bottom) row of maximal zones, where each maximal zone is of height 1 (2) . In Fig. 4, the leftmost column of maximal zones in $w(1,1,9,8)$ are $MZ(1,1;2^0,2^0)$, $MZ(1,2;2^1,2^0)$, $MZ(1,4;2^2,2^0)$, and

$$MZ(1,8;2^1,2^0).$$

If that column of maximal zones is stripped off, the window becomes a smaller one, namely, $w(2,1,9,7)$. In that smaller window, the maximal zones along the topmost row are $MZ(2,1;2^0,2^1)$, $MZ(4,1;2^0,2^2)$, and $MZ(8,1;2^0,2^0)$. After performing the above two stripping steps, the maximal zones along the rightmost column of the remaining window $w(2,2,8,7)$ are $MZ(8,2;2^1,2^0)$, $MZ(8,4;2^2,2^0)$, and $MZ(8,8;2^1,2^0)$. This is the basic concept of the proposed strip-splitting approach and each column (row) of maximal zones can be viewed as a strip. By Property 2, each strip can be decomposed into a set of equal-sized maximal blocks.

For decomposing the initial query window

$$W^{(0)} = w(x,y,n_1,n_2)$$

into a set of maximal blocks, the $k$th strip-splitting process for $1 \leq k \leq \lfloor \log(\max(n_1,n_2)) \rfloor$ is denoted by the procedure $SS^{(k)}(W^{(k-1)})$, including four steps $V_l$, $H_t$, $V_r$, and $H_b$ for splitting the leftmost boundary, topmost boundary, rightmost boundary, and bottommost boundary, respectively. Each of $V_l$, $H_t$, $V_r$, and $H_b$ is used to split one boundary strip of the current window and to generate a possible set of maximal blocks, each with size $2^{k-1} \times 2^{k-1}$. Suppose the current window is $W^{(k-1)}$. After performing the $k$th strip-splitting process on $W^{(k-1)}$, we have $W^{(k)} = SS^{(k)}(W^{(k-1)})$. In other words, we have $W^{(1)} = SS^{(1)}(W^{(0)})$, $W^{(2)} = SS^{(2)}(W^{(1)}), \ldots,$ and

$$W^{(\lfloor \log n_l \rfloor)} = SS^{(\lfloor \log n_l \rfloor)}(W^{(\lfloor \log n_l \rfloor - 1)}),$$

where $n_l = \max(n_1,n_2)$. For example, the initial window $W^{(0)}$ is set to $W = w(1,1,9,8)$, as shown in Fig. 2. Then, $W^{(1)} = w(2,2,8,6)$ is obtained by performing $SS^{(1)}(W^{(0)})$, i.e., splitting the leftmost, topmost, and rightmost strips of $W^{(0)}$. The four steps, $V_l$, $H_t$, $V_r$, and $H_b$, in each strip-splitting process are listed as follows: In fact, each step is quite similar only different in the related coordinates:

```
1. $V_l$: If $F(x,2^k) \neq 0$, then
       {output $\frac{n_1}{2^{k-1}}$ maximal blocks, $MB(x,y,2^{k-1})$,
           $MB(x,y+2^{k-1},2^{k-1}), \ldots,$ and
           $MB(x,y+n_1-2^{k-1},2^{k-1})$;
       $x = x + 2^{k-1}$;
       $n_2 = n_2 - 2^{k-1}$}

2. $V_t$: If $F(y,2^k) \neq 0$, then
       {output $\frac{n_2}{2^{k-1}}$ maximal blocks, $MB(x,y,2^{k-1})$,
           $MB(x+2^{k-1},y,2^{k-1}), \ldots,$ and
           $MB(x+n_2-2^{k-1},y,2^{k-1})$;
       $y = y + 2^{k-1}$;
       $n_1 = n_1 - 2^{k-1}$}

3. $V_r$: If $F(x+n_2,2^k) \neq 0$, then
       {output $\frac{n_1}{2^{k-1}}$ maximal blocks,
       $MB(x+n_2-2^{k-1},y,2^{k-1})$,
           $MB(x+n_2-2^{k-1},y+2^{k-1},2^{k-1}), \ldots,$ and
           $MB(x+n_2-2^{k-1},y+n_1-2^{k-1},2^{k-1})$;
       $n_2 = n_2 - 2^{k-1}$}

4. $V_b$: If $F(y+n_1,2^k) \neq 0$, then
       {output $\frac{n_2}{2^{k-1}}$ maximal blocks,
       $MB(x,y+n_1-2^{k-1},2^{k-1})$,
           $MB(x+2^{k-1},y+n_1-2^{k-1},2^{k-1}), \ldots,$ and
           $MB(x+n_2-2^{k-1},y+n_1-2^{k-1},2^{k-1})$;
       $n_1 = n_1 - 2^{k-1}$}.
```

## 3.2 The Proposed Algorithm

Based on the strip-splitting approach described in Section 3.1, the formal algorithm for decomposing a query window into a set of maximal blocks is described in this section. The proposed algorithm indeed performs the strip-splitting $SS^{(k)}(W^{(k-1)})$ for $1 \leq k \leq \lfloor \log n_l \rfloor$, where $n_l = \max(n_1,n_2)$. That is, the number of

TABLE 1
The Simulation for Generating Maximal Blocks of $W$

| $k$ | step | window | $F()$ | Maximal blocks $MB(x,y,s)$ |
|---|---|---|---|---|
| 1 | $V_l$ | $w(1,1,9,8)$ | 1 mod $2^1 \neq 0$ | $MB(1,1,1)$, $MB(1,2,1)$, ..., $MB(1,9,1)$ |
| | $H_t$ | $w(2,1,9,7)$ | 1 mod $2^1 \neq 0$ | $MB(2,1,1)$, $MB(3,1,1)$, ..., $MB(8,1,1)$ |
| | $V_r$ | $w(2,2,8,7)$ | (2+7) mod $2^1 \neq 0$ | $MB(8,2,1)$, $MB(8,3,1)$, ..., $MB(8,9,1)$ |
| | $H_b$ | $w(2,2,8,6)$ | (2+8) mod $2^1$ =0 | |
| 2 | $V_l$ | $w(2,2,8,6)$ | 2 mod $2^2 \neq 0$ | $MB(2,2,2)$, $MB(2,4,2)$, ..., $MB(2,8,2)$ |
| | $H_t$ | $w(4,2,8,4)$ | 2 mod $2^2 \neq 0$ | $MB(4,2,2)$, $MB(6,2,2)$ |
| | $V_r$ | $w(4,4,6,4)$ | (4+4) mod $2^2$ =0 | |
| | $H_b$ | $w(4,4,6,4)$ | (4+6) mod $2^2 \neq 0$ | $MB(4,8,2)$, $MB(6,8,2)$ |
| 3 | $V_l$ | $w(4,4,4,4)$ | 4 mod $2^3 \neq 0$ | $MB(4,4,4)$ |

strip-splitting processes in the proposed algorithm is bounded by $\lfloor \log\ n_l \rfloor$. The formal algorithm is shown as follows:

```
Algorithm: Decomposing a query window into maximal blocks.
Input: A query window $W^{(0)} = w(x, y, n_1, n_2)$.
Output: Maximal blocks of the query window stored in an output
        array.
  $k$ is initially set to 1;
  Repeat the following steps on $W^{(k-1)}$
    $V_l$; /* stripping the leftmost strip */
    $H_t$; /* stripping the topmost strip */
    $V_r$; /* stripping the rightmost strip */
    $H_b$; /* stripping the bottommost strip */
    $k = k + 1$;
  Until $W^{(k-1)}$ becomes null;
```

Return to Fig. 2. The query window $W$ is used as an example to demonstrate the proposed algorithm. Initially, the window is $w(1,1,9,8)$. The variable $k$ is initially set to one. For $V_l$, since $F(1,2^1) \neq 0$, nine maximal blocks $MB(1,i,1)$ for $1 \leq i \leq 9$ are generated and the window $W$ is split by moving out a vertical strip of size $9 \times 1$. Then, the remaining window is $w(2,1,9,7)$. Similarly, we have $F(1,2^1) \neq 0$, for $H_t$, thus seven maximal blocks $MB(i,1,1)$ for $2 \leq i \leq 8$ are generated. The remaining window is denoted by $w(2,2,8,7)$. Since $F(2+7,2^1) \neq 0$, for $V_r$, eight maximal blocks $MB(2+7-1,i,1)$ for $2 \leq i \leq 9$ are generated and the remaining window is denoted by $w(2,2,8,6)$. For $H_b$, we do nothing since $F(2+8,2^1) = 0$. After performing the above four steps, $k$ is increased by one. That is, $k = 2$. The other maximal blocks can be obtained by the same way and the detailed simulation for decomposing the window $W$ into a set of maximal blocks is illustrated in Table 1. The outputted maximal blocks are stored in an array.

### 3.3 Time Complexity Analysis

**Lemma 1: ([8], p. 60).** *Given a query window, each maximal block of smaller size is not surrounded by other maximal blocks of greater size, i.e., a quadtree node cannot be adjacent to two nodes of greater size on opposite edges or on diagonally opposite vertices.*

By Lemma 1 and the proposed algorithm for decomposing a query window into a set of maximal blocks, the concerning maximal blocks are generated along the boundary of the current window iteratively and the size-sequence of those generated maximal blocks is an increasing sequence. For example, the size-sequence of the maximal blocks of $W$ in Fig. 2 is

$$\langle \underbrace{1\times 1, 1\times 1, \ldots, 1\times 1}_{\text{24 maximal blocks}}, \underbrace{2\times 2, 2\times 2, \ldots, 2\times 2}_{\text{8 maximal blocks}}, \text{ and } 4\times 4\rangle.$$

It is known that, the proposed algorithm includes $k^*$ strip-splittings, $SS^{(1)}$, $SS^{(2)}$, ..., and $SS^{k^*}$, for $k^* \leq \lfloor \log n_l \rfloor$. In one strip-splitting, the time complexity required in each of $V_l$, $H_t$, $V_r$, and $H_b$ ranges from $O(1)$ to $O(n^*)$, where $n^*$ is the number of those generated maximal blocks in each step. Thus, the time complexity of the proposed algorithm is dominated by the total number of generated maximal blocks. Before analyzing the time complexity of the proposed algorithm, we need the following result.

**Lemma 2: ([3]).** *The number of maximal blocks inside an $n \times n$ query window is bounded by $3(2n - \log n) - 5 (= O(n))$.*

By Lemma 2, the total number of maximal blocks of a square window is linearly proportional to one dimension of the query window. For any arbitrary rectangular window of size $n_1 \times n_2$, the total number of maximal blocks of the given window is linearly proportional to $\max(n_1, n_2)$. For example, a $1 \times 100$ window has 100 maximal blocks. Thus, the proposed algorithm takes $O(n_l)$ time to generate all the maximal blocks inside the given window, where $n_l = \max(n_1, n_2)$. In addition, the size of the working memory required is also $O(n_l)$. Thus, we have the following main result.

**Theorem 1.** *Given a query window of size $n_1 \times n_2$, the proposed algorithm takes $O(n_l)$ time to generate all the maximal blocks in that window and needs $O(n_l)$ working memory, where $n_l = \max(n_1, n_2)$.*

Although Proietti's algorithm takes $O(n_l)$ time and is the first optimal algorithm for solving this decomposition problem, it does perform some redundant checks even when no maximal block is in one divided block (see Section 2). However, in our proposed algorithm, there is no such overhead.

## 4 Experimental Results

In this section, both algorithms are coded in $C$ programming language and are executed on a Pentium 133-based $PC$ with the same inputs. Given a $2^{10} \times 2^{10}$ queried image, we randomly generate 10,000 square query windows as inputs.

In the experimentation, we set $n_1 = n_2 = n$ and randomly choose the starting points for the generated query windows. From Theorem 1, the execution time required in the proposed algorithm can be rewritten as $T_{ours} = C_{ours} \times n$ seconds for some constant $C_{ours}$. Similarly, the execution time required in Proietti's algorithm [7] is denoted by $T_{Pr} = C_{Pr} \times n$ seconds for some constant $C_{Pr}$. Experimental results reveal that our proposed algorithm is linearly proportional to the width of the window within a small range centered around $2.5 \times 10^{-5}$ and the execution time of the algorithm by Proietti [7] has constant term $C_{Pr}$ centered around $5.3 \times 10^{-5}$. The improvement ratio of the average execution time required in our proposed algorithm over Proietti's algorithm is denoted by $R_s = \frac{T_{Pr} - T_{ours}}{T_{Pr}} \times 100\%$ and experimental results reveal that our proposed new algorithm has about 48-57 percent time improvement.

## 5 Conclusion

We have presented a new optimal algorithm for decomposing a query window into a set of maximal quadtree blocks. From the big-O complexity notation, the proposed algorithm has the same time complexity as the previous algorithm of Proietti [7]; however, the proposed algorithm has about 48-57 percent time improvement due to the computational advantage of strip-splitting approach. In fact, the proposed algorithm in this paper can be applied to the gray images decomposed by the S-tree representation [1].

## Acknowledgments

Kuo-Liang Chung was supported by the National Science Council under contracts NSC89-2213-E011-017 and NSC91-2213-E011-028.

## References

[1] W.G. Aref and H. Samet, "Decomposing a Window into Maximal Quadtree Blocks," *Acta Informatica,* vol. 30, pp. 425-439, 1993.
[2] K.L. Chung and J.G. Wu, "Improved Image Compression Using S-Tree and Shading Approach," *IEEE Trans. Comm.,* vol. 48, no. 5, pp. 748-751, 2000.
[3] C. Faloutsos, H.V. Jagadish, and Y. Manolopoulos, "Analysis of n-Dimensional Quadtree Decomposition of Arbitrary Rectangles," *IEEE Trans. Knowledge and Data Eng.,* vol. 9, no. 3, pp. 373-383, May/June 1997.
[4] E. Nardelli and G. Proietti, "Efficient Secondary Memory Processing of Window Queries on Spatial Data," *Information Sciences,* vol. 84, pp. 67-83, 1995.
[5] E. Nardelli and G. Proietti, "Time and Space Efficient Secondary Memory Representation of Quadtrees," *Information Systems,* vol. 22, pp. 25-37, 1997.
[6] J.A. Orenstein and F.A. Manola, "Probe Spatial Data Modeling and Query Processing in an Image Database Application," *IEEE Trans. Software Eng.,* vol. 14, no. 5, pp. 611-629, 1988.
[7] G. Proietti, "An Optimal Algorithm for Decomposing a Window into Maximal Quadtree Blocks," *Acta Informatica,* vol. 36, no. 4, pp. 257-266, 1999.
[8] H. Samet, *Applications of Spatial Data Structures.* Addison Wesley, 1990.
[9] C.A. Shaffer, "A Formula for Computing the Number of Quadtree Node Fragments Created by a Shift," *Pattern Recognition Letters,* vol. 7, no. 1, pp. 45-49, 1988.

# Discovery of Context-Specific Ranking Functions for Effective Information Retrieval Using Genetic Programming

Weiguo Fan, Michael D. Gordon, and Praveen Pathak

**Abstract**—The Internet and corporate Intranets have brought a lot of information. People usually resort to search engines to find required information. However, these systems tend to use only one fixed ranking strategy regardless of the contexts. This poses serious performance problems when characteristics of different users, queries, and text collections are taken into account. In this paper, we argue that the ranking strategy should be context specific and we propose a new systematic method that can automatically generate ranking strategies for different contexts based on *Genetic Programming* (GP). The new method was tested on TREC data and the results are very promising.

**Index Terms**—Intelligent information retrieval, personalization, search engine, term weighting, ranking function, text mining, genetic programming, contextual information retrieval, information routing, information retrieval.

## 1 Introduction

The Internet has brought far more information than anybody can absorb. Similarly, organizations store a large amount of information in manuals, procedures, documentation, expert knowledge, e-mail archives, news sources, and technical reports. Such a large amount of information serves as a huge information repository for organizations. However, it also makes finding relevant information from it extremely difficult. How to help users find their required information is the central task of any information retrieval (IR) system or search engine. However, precision and recall, the two most commonly used performance measures, of commonly used search engines are usually very low [2].

Retrieval performance of an IR system can be affected by many factors: the ambiguity of query terms, unfamiliarity with system features, as well as factors relating to document representation [6]. Many approaches have been proposed to address these issues. For example, query expansion techniques based on a user's relevance feedback have been used to discover a user's real information need [3]. Similarly, document descriptions have been modified [1]. Another very important factor that is often overlooked by most researchers is the ranking/matching function. It is this ranking function that we focus most of our discussion on.

A ranking function is used to order documents in terms of their predicted relevance to a particular query. It is very difficult to design such a ranking function that can be successful for every query, user, or document collection (which we will call contexts). In this paper, we argue in favor of a method that systematically adapts a ranking function and tailors it to different users' needs (i.e. in different contexts). In particular, we will use Genetic

- *W. Fan is with the Department of Accounting and Information Systems, Virginia Polytechnic Institute and State University, 3007 Pamplin Hall, Blacksburg, VA 24061. E-mail: wfan@vt.edu.*
- *M.D. Gordon is with the Department of Computer and Information Systems, University of Michigan, E2420 Business Adminstration Building, 701 Tappan Stree, Ann Arbor, MI 48109. E-mail: mdgordon@umich.edu.*
- *P. Pathak is with the Decision and Information Sciences Department, Warrington College of Business, University of Florida, PO Box 117169, Gainesville, FL 32611. E-mail: praveen@ufl.edu.*

*Manuscript received 4 June 2002; revised 4 Mar. 2003; accepted 20 May 2003. For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number 116690.*