

Efficient BSR-Based Parallel Algorithms for Geometrical Problems

David Semé and Jean-Frédéric Myoupo
Laboratoire de Recherche en Informatique d'Amiens (LaRIA)
Université de Picardie Jules Verne
Faculté de Mathématiques et d'Informatique
33 rue Saint-Leu, 80039 Amiens Cedex 1, France
e-mail: {myoupo, seme}@laria.u-picardie.fr

Abstract

This paper presents BSR-parallel algorithms for three geometrical problems : point location, convex hull and smallest enclosing rectangle. These problems are solved in constant time using the BSR model introduced by S. Akl in [2]. The first algorithm uses $O(N)$ processors (N is the number of edges of the polygon R). The second uses $O(N'^2)$ processors (N' is the number of points) and the third one uses $O(N'^2)$ processors (it need the convex hull) to solve the smallest enclosing rectangle problem. These new results suggest that many other geometrical problems can be solved in constant time using the BSR model.

1. Introduction

This paper describes BSR (Broadcasting with Selective Reduction) solutions to three geometrical problems : point location, convex hull and smallest enclosed rectangle. These problems are not only central to practical applications, but are also a vehicle for the solution of a number of apparently unrelated questions arising in computational geometry.

Point location is one of the classical problems in computational geometry and has various applications of practical relevance, for example, in the areas of geographic information systems (GIS) or computer-aided design and engineering (CAD/CAE). The problem is well studied in computational geometry literature and several theoretically optimal algorithms have been proposed [21, 14, 15, 8, 3]. The computation of the convex hull of a finite set of points, particularly in the plane, has been studied extensively [21, 14] and has applications, for example, in pattern recognition [5, 7], image processing [22] and stock cutting and allocation [10, 24, 11]. The concept of convex hull of a set of points S is natural and easy to understand. By definition, it is the smallest convex set containing S .

Intuitively, if S consists of a finite set of points in the plane, imagine surrounding the set by a large, stretched rubber band; when the band is released it will assume the shape of the convex hull. One BSR solution of this problem has been described in [2].

The smallest enclosing rectangle problem is an extension of the convex hull problem [11, 13]. Thus, this problem has applications in pattern recognition, image processing and stock cutting and allocation. The concept of smallest enclosing rectangle of a set of points S is natural and easy to understand. By definition, it is the smallest set containing S which is a rectangle.

The Parallel Random Access Machine (PRAM) is undoubtedly the most popular model of parallel computation [1, 12]. The model consists of a number of processors sharing a common memory. The processors solve a computational problem in parallel by executing the steps of an algorithm simultaneously. The shared memory stores data and results, and also serves as the communication medium for the processors. An interconnection unit (IU) allows the processors to gain access to the memory locations for the purpose of reading or writing. The model is further specified by defining the mode of memory access. Three variants are most commonly used. In the Exclusive-Read Exclusive-Write (EREW) PRAM, no two processors can gain access to the same memory location simultaneously whether for reading or for writing. The Concurrent-Read Exclusive-Write (CREW) PRAM allows more than one processor to read from, but not write into, the same memory location at the same time. Finally, in the Concurrent-Read Concurrent-Write (CRCW) PRAM it is possible for several processors to gain access to the same memory location either for the purpose of reading or for the purpose of writing. (Note that the fourth logical variant, the Exclusive-Read Concurrent-Write PRAM, has a very restricted range of applications [1]).

Broadcasting with Selective Reduction (BSR) introduced in [2] and widely used in [16, 18, 19, 20, 23, 25] is an extension of the EREW PRAM, permitting an additional form of concurrent access of shared memory, namely the BROADCAST instruction.

The BROADCAST instruction allows all processors to simultaneously write to all shared memory locations. Each processor produces a *tag* and a *datum*, according to expressions given in the BROADCAST statement. Each shared variable, having been previously assigned a selection operation, σ , selects among the incoming data by examining the accompanying tag values, and testing the condition *tag σ limit*. The *limit* parameter must be set up before the BROADCAST, and is (potentially) unique for each shared variable. For all tags satisfying this condition, the accompanying data are “accepted” by the shared variable, and combined under the reduction operation \mathcal{R} . The BROADCAST instruction executes in one cycle, just as any shared memory access.

Notation for the BROADCAST

A mathematical notation is employed to describe the BROADCAST instruction.

Let us define the following symbols :

- p_i : processor i
- d_i : datum broadcast by p_i
- t_i : tag broadcast by p_i
- x_j : the j -th element of shared array x
- l_j : limit value associated with j -th element
- m : the size of a shared array
- σ : selection operation, where
 $\sigma \in \{<, \leq, =, \geq, >, \neq\}$
- \mathcal{R} : binary associative reduction operation, where
 $\mathcal{R} \in \{\sum, \prod, \wedge, \vee, \oplus, \cap, \cup\}$,
denoting sum, product, AND, OR, XOR, max,
and min respectively
- N : number of processors
- M : number of shared memory locations, where
 $M = cm$, for some $c \geq 1$

The BSR BROADCAST instruction is denoted by :

$$x_j := \mathcal{R}_{1 \leq i \leq m} d_i \quad \text{for } 1 \leq j \leq n$$

When the ranges of the variables are understood, this can be abbreviated as :

$$x_j := \mathcal{R}_{t_i \sigma l_j} d_i$$

The above notation can be interpreted as follows. For each

memory location x_j (with an associated limit value l_j), the operation $(t_i \sigma l_j)$ is tested over all broadcast pairs t_i, d_i . In every case for which t_i satisfies the proposition, d_i is “accepted” by location x_j . The set of all data accepted by x_j is reduced to a single value by means of the binary associative operation \mathcal{R} , and stored in shared variable x_j . If no data are accepted by a given memory location, then the value of that shared variable is not affected, then x_j is assigned the value of that datum. It is important to note that on an N processors, M memory locations BSR model, the above BROADCAST instruction takes $O(T(N, M))$ time. In the other hand, on a CRCW PRAM with the same number of processors and memory locations we do not know how to perform the same computations in less than $O(N * T(N, M))$ time.

This paper presents constant time solutions for the three problems. Algorithms use the BSR with multi-criteria introduced by Akl and Stojmenovic in [4]. In this model, these algorithms use either a linear (the point location algorithm) or a quadratic (convex hull and smallest enclosing algorithms) number of processors.

Section 2 presents the point location problem and its BSR solution. The convex hull problem is presented in the section 3 and the section 4 presents the smallest enclosing rectangle problem. The conclusion ends the paper.

2 The point location problem

2.1 Statement of the problem

The point location problem can be described as follows : given a query point z , we want to determine whether it lies in a region R [21]. Here, we consider the version where z is a planar point and R is a simple polygon. In addition, we can extend our work to answer m queries : given m points and simple polygon, determine which points are in the polygon. One point can be located in a simple polygon R in constant time and m points can be located in a simple polygon R in $O(m)$ on BSR model.

2.2 The BSR solution of the point location problem

Consider the horizontal line l passing through z . If l does not intersect R , then z is external. So, assume that l intersects R and consider firstly the case of l not going through any vertex of R . Let C the number of intersections of l with the boundary of R to the left of z . Since R is bounded, the left extremity of l lies in the exterior of R . Consider moving right on l from $-\infty$, towards z . At the leftmost crossing with the boundary of R , we move

into the interior. At the next crossing, we return outside again, and so on. Thus z is internal if and only if C is odd. Next consider the degenerate situation where l passes through vertices of R . An infinitesimal counterclockwise rotation of l around z does not change the classification of z (internal/external) but removes the degeneracy. Thus, if we imagine to perform this infinitesimal rotation, we recognize : if both vertices of an edge belong to l , this edge must be ignored; if just one vertex of an edge belongs to l , then it must be counted if it is the vertex with larger ordinate, and ignored otherwise. In summary, we have the following program executed by each processor (Step 1 to 6).

Let N be the number of the edges of the polygon R . In the following algorithm ZX and ZY represent the coordinates of the point to be located z . (RX_i, RY_i) is the coordinates of the i -th point of the polygon R and (ZX_j, ZY_j) is the coordinates of the j -th point to be located. Notations RX, RY, ZX and ZY are used for convenience.

Step 1 : We determine if the point to be located is at the left or at the right of each point of the polygon. Similarly, we determine if the point to be located is at the top or at the bottom of a point of the polygon.

For $0 < i \leq N$:

$$RX'_i = RX_i - ZX_i$$

$$RY'_i = RY_i - ZY_i$$

Step 2 : X_i is equal to 1 if the ZX -coordinate of the point to be located is less than RX_{i-1} and greater than RX_i . Similarly, X'_i is equal to 1 if the ZY -coordinate of the point to be located is less than RY_{i-1} and greater than RY_i .

For $0 < i \leq N$:

IF ($i = 1$)

$$X_i = \bigcap_{\substack{RX'_N > 0 \\ \& RX'_1 \leq 0}} 1$$

$$X'_i = \bigcap_{\substack{RY'_i > 0 \\ \& RY'_N \leq 0}} 1$$

ELSE

$$X_i = \bigcap_{\substack{RX'_{i-1} > 0 \\ \& RX'_i \leq 0}} 1$$

$$X'_i = \bigcap_{\substack{RY'_i > 0 \\ \& RY'_{i-1} \leq 0}} 1$$

ENDIF

Step 3 : X''_i is equal to 1 if the point to be located is on the left of the i -th edge.

For $0 < i \leq N$:

$$X''_i = X_i \vee X'_i$$

Step 4 : In this step we compute the Y -coordinates of all points which are at the left of the point to be located.

For $0 < i \leq N$:

IF ($i = 1$)

$$Y_i = X''_i * (RX'_i * RY'_N - RX'_N * RY'_i) / (RY'_N - RY'_i)$$

ELSE

$$Y_i = X''_i * (RX'_i * RY'_{i-1} - RX'_{i-1} * RY'_i) / (RY'_{i-1} - RY'_i)$$

ENDIF

Step 5 : C contains the number of Y_i 's which are a positive.

For $0 < i \leq 1, 0 < i' \leq N$:

$$C = \sum_{Y_{i'} > 0} 1$$

Step 6 : D is equal to **TRUE** if C is odd, D is equal to **FALSE** otherwise.

For $0 < i \leq 1$:

IF ($C \bmod 2 = 1$)

$$D = \text{TRUE}$$

ELSE

$$D = \text{FALSE}$$

ENDIF

3 The convex hull problem

3.1 Statement of the problem

The concept of convex hull of a set of points S is natural and easy to understand. By definition, it is the smallest

convex set containing S . Intuitively, if S consists of a finite set of points in the plane, imagine surrounding the set by a large, stretched rubber band ; when the band is released it will assume the shape of the convex hull. An example of convex hull is described in Figure 1.

3.2 The BSR solution of the convex hull problem

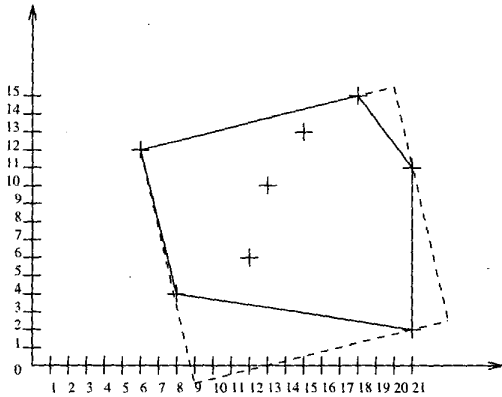


Figure 1. Example of convex hull and smallest enclosing rectangle.

Here we describe in details our BSR algorithm. In drawing the horizontal straight line and the vertical straight line through a point P_i we divide the plan of four regions : A, B, C and D (Figure 2). First, for a point P_i , if there exists points in its four regions, then this point P_i is not in the convex hull (Step 1 to 3 of the BSR algorithm instructions). All points which are not in the convex hull are marked by a 1 in the array E , and the others are marked by a 0 and are called 0-marked points. In Step 4, for each 0-marked point P_i in the array E , we compute the slope of straight lines containing this 0-marked point P_i and others 0-marked points. Then, for each 0-marked point P_i , we keep only two straight lines containing it. These two lines shape the maximal angle at P_i (Steps 5 and 6). Figures 3,4,5,6 show the four cases of location of points in the four regions of a 0-marked point. A dark region means that there exists at least one point in this region. For each point P_i , these lines are kept with respect of the position of other points in the four regions of this point P_i . So, we keep all points having two lines shaping a convex angle (Steps 7 and 8). It remains to establish the coordinates of these points (Steps 9 and 10). The convex hull is made up by these points.

In the following, N' represents the number of points of the set S . So, X_i and Y_i represent coordinates of the i -th point of S called P_i . An example of convex hull is shown in Figure 1.

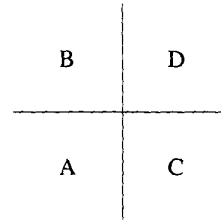


Figure 2. Horizontal line and vertical line passing through a point divide the plane in four region.



Figure 3. These cases correspond to the condition $(A'_i = 0 \text{ AND } D'_i = 0) \text{ OR } (B'_i = 1 \text{ AND } C'_i = 1)$ in the Step 5 of the algorithm.

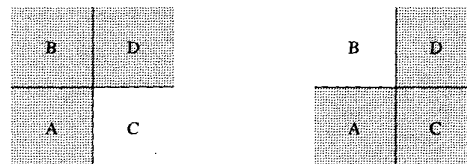


Figure 4. These cases correspond to the condition $(B'_i = 0 \text{ AND } C'_i = 0) \text{ OR } (A'_i = 1 \text{ AND } D'_i = 1)$ in the Step 5 of the algorithm.

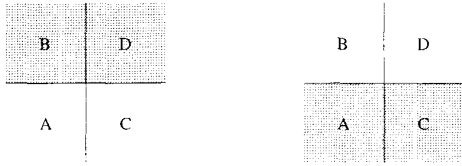


Figure 5. These cases correspond to the condition ($A'_i = C'_i$ AND $B'_i = D'_i$ AND $A'_i \neq B'_i$) in the Step 5 of the algorithm.

Step 1 : A_i is equal to 0 if there is no point on the bottom-left of the i -th point of S . B_i is equal to 0 if there is no point on the top-left of the i -th point of S . C_i is equal to 0 if there is no point on the bottom-right of the i -th point of S . D_i is equal to 0 if there is no point on the top-right of the i -th point of S .

For $0 < i, i' \leq N'$:

$$A_i = \bigcap_{\substack{X_i \geq X_{i'} \\ \& Y_i \geq Y_{i'} \\ \& i \neq i'}} 1$$

$$B_i = \bigcap_{\substack{X_i \geq X_{i'} \\ \& Y_i \leq Y_{i'} \\ \& i \neq i'}} 1$$

$$C_i = \bigcap_{\substack{X_i \leq X_{i'} \\ \& Y_i \geq Y_{i'} \\ \& i \neq i'}} 1$$

$$D_i = \bigcap_{\substack{X_i \leq X_{i'} \\ \& Y_i \leq Y_{i'} \\ \& i \neq i'}} 1$$

Step 2 : E_i is equal to 1 means that the i -th point of S is not in the convex hull.

For $0 < i \leq N'$:

$$E_i = A_i \wedge B_i \wedge C_i \wedge D_i$$

Step 3 : This step is similar to the step 1 but here inequalities are strict.

For $0 < i, i' \leq N'$:

IF ($E_i = 0$)

$$A'_i = \bigcap_{\substack{X_i > X_{i'} \\ \& Y_i > Y_{i'}}} 1$$

$$B'_i = \bigcap_{\substack{X_i > X_{i'} \\ \& Y_i < Y_{i'}}} 1$$

$$C'_i = \bigcap_{\substack{X_i < X_{i'} \\ \& Y_i > Y_{i'}}} 1$$

$$D'_i = \bigcap_{\substack{X_i < X_{i'} \\ \& Y_i < Y_{i'}}} 1$$

ELSE

$$A'_i = A_i$$

$$B'_i = B_i$$

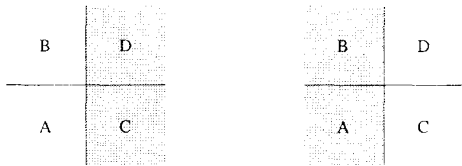


Figure 6. These cases correspond to the condition ($A'_i = B'_i$ AND $C'_i = D'_i$ AND $A'_i \neq C'_i$) in the Step 5 of the algorithm.

$$C'_i = C_i$$

$$D'_i = D_i$$

ENDIF

Step 4 : $F_{i,j}$ is the reverse of the slope of the straight line containing P_i and P_j .

For $0 < i, j \leq N'$:

IF ($X_i = X_j$)

IF ($Y_i > Y_j$)

$$F_{i,j} = +\infty$$

ELSE

$$F_{i,j} = -\infty$$

ENDIF

ELSE

IF ($Y_i = Y_j$)

$$F_{i,j} = 0$$

ELSE

$$F_{i,j} = (X_i - X_j)/(Y_i - Y_j)$$

ENDIF

ENDIF

Step 5 : G_i and H_i are made up by the reverses of the slopes of the two lines containing P_i which shape the maximal angle at P_i .

For $0 < i, j \leq N'$:

IF ($E_i = 0$)

IF ($(B'_i = 0 \text{ AND } C'_i = 0) \text{ OR } (A'_i = 1 \text{ AND } D'_i = 1)$)

$$G_i = \bigcap_{F_{i,j} > 0} F_{i,j}$$

$$H_i = \bigcup_{F_{i,j} > 0} F_{i,j}$$

ENDIF

IF ($(A'_i = 0 \text{ AND } D'_i = 0) \text{ OR } (B'_i = 1 \text{ AND } C'_i = 1)$)

$$G_i = \bigcup_{F_{i,j} < 0} F_{i,j}$$

$$H_i = \bigcap_{F_{i,j} < 0} F_{i,j}$$

ENDIF

IF ($A'_i = C'_i \text{ AND } B'_i = D'_i \text{ AND } A'_i \neq B'_i$)

$$G_i = \bigcup_{F_{i,j} > 0} F_{i,j}$$

$$H_i = \bigcap_{F_{i,j} < 0} F_{i,j}$$

ENDIF

IF ($A'_i = B'_i \text{ AND } C'_i = D'_i \text{ AND } A'_i \neq C'_i$)

$$G_i = \bigcup_{F_{i,j} < 0} F_{i,j}$$

$$H_i = \bigcap_{F_{i,j} > 0} F_{i,j}$$

ENDIF

ELSE

$$G_i = 0$$

$$H_i = 0$$

ENDIF

Step 6 : $-\infty$ is changed into $+\infty$ for a best homogeneity of the values of G and H .

For $0 < i \leq N'$:

IF ($G_i = -\infty$)

$$G_i = +\infty$$

ENDIF

IF ($H_i = -\infty$)

$$H_i = +\infty$$

ENDIF

Step 7 : K_i is equal to 1 if the value of G_i exists in H . Similarly, L_i is equal to 1 if the value of H_i exists in G .

For $0 < i, i' \leq N'$:

IF ($E_i = 1$)

$$K_i = \bigcap_{\substack{G_i = H_{i'} \\ \& i \neq i'}} 1$$

$$L_i = \bigcap_{\substack{H_i = G_{i'} \\ \& i \neq i'}} 1$$

ENDIF

Step 8 : M_i is equal to 1 if the point P_i is in the convex hull.

For $0 < i \leq N'$:

$$M_i = K_i \wedge L_i$$

Step 9 : N_i contains indices which will be used in the next step.

For $0 < i, i' \leq N'$:

$$N_i = \sum_{i' \leq i} M_{i'}$$

Step 10 : Xch and Ych contain the coordinates of the points of the convex hull.

For $0 < i, i' \leq N'$:

$$Xch_i = \bigcap_{\substack{M_{i'}=1 \\ \& N_{i'}=i}} X_{i'}$$

$$Ych_i = \bigcap_{\substack{M_{i'}=1 \\ \& N_{i'}=i}} Y_{i'}$$

4 The smallest enclosing rectangle problem

4.1 Statement of the problem

It is well known [11, 13] that the smallest enclosing rectangle of a set of N planar points has at least one side that is an extension of an edge of the convex hull. Hence, the smallest enclosing rectangle may be found by first computing the convex hull; then determining for each convex hull edge, the smallest enclosing rectangle that has one side which is an extension of this edge; and finally determining the smallest of these rectangles (see Figure 1).

4.2 The BSR solution of the smallest enclosing rectangle problem

First, we suppose that the whole plane has rectangular axes namely x-axis and y-axis (figure 1). We also suppose that the convex hull has been computed and that it has N'' points. Let $CH = \{(Xch_i, Ych_i) | 1 \leq i \leq N''\}$ be the set of these N'' points. Consider an edge EDG_i of the convex hull whose two end points are of coordinates (Xch_i, Ych_i) and (Xch_{i+1}, Ych_{i+1}) respectively for all $1 \leq i \leq N'' - 1$. The edge $EDG_{N''}$ is defined by the two end points of coordinates $(Xch_{N''}, Ych_{N''})$ and (Xch_1, Ych_1) respectively. If $Xch_i < Xch_{i+1}$ and $Ych_i < Ych_{i+1}$ or $Xch_i > Xch_{i+1}$ and $Ych_i > Ych_{i+1}$ (the slope EDG_i is positive), then bring the origin of the whole plane at point (Xch_i, Ych_i) .

Update the coordinates of the points of CH with respect to this new origin. Change the axes such that one of the new axis and the edge EDG_i are merged. Update the precedent coordinates of points of CH (obtained with the new origin) with respect of these new axes. Find the maximal value $MaxX_i$ and the minimal value $MinX_i$ of the x-axis values of the points of CH in the new axes. Thus, $MaxX_i - MinX_i$ is the length of one side of the enclosing rectangle. In the same way, look for the maximal value $MaxY_i$ and the minimal value $MinY_i$ along the y-axis. Thus, $MaxY_i - MinY_i$

is the length of the other side of the enclosing rectangle. Therefore the area of the enclosing rectangle is $Arect_i = (MaxX_i - MinX_i) * (MaxY_i - MinY_i)$.

Do the same thing for all the edges EDG_i . We then obtain a set of enclosing rectangles. Then, find the minimum enclosing rectangle by computing the minimum of the $Arect_i$'s. Finally, compute the coordinates of the four points of the smallest enclosing rectangle.

In the following, N'' represents the number of points of the convex hull computed in the next section. So, Xch_i and Ych_i represent coordinates of the i -th point of the convex hull. An example of smallest enclosing rectangle is shown in Figure 1. The full algorithm with a complete example of the smallest enclosing rectangle is described in [17]

5 Concluding remarks

Akl and Guenther in [2] explained how a BSR operation can be carried out in constant time. In [4] Akl and Stojmenovic showed that a multi criteria BSR operation can be also done in constant time. In ours algorithms, we use a bounded number of BSR operations (or multi criteria BSR). These operations are classical operations to BSR (affectation or arithmetic operations) which are carried out in constant time. So, these algorithms are carried out in constant time.

These algorithms use a linear (the point location algorithm) or a quadratic (convex hull and smallest enclosing algorithms) number of processors. The first algorithm uses $O(N)$ processors (N is the number of edges of the polygon R), the second one uses $O(N^2)$ processors (N' is the number of points) and the third one uses $O(N^2)$ processors because of the computation of the convex hull to solve the smallest enclosing rectangle problem. These results suggest that many other geometrical problems can be solved in constant time on the BSR model.

References

- [1] S.G. Akl, The Design and Analysis of Parallel Algorithms, Prentice Hall, Englewood Cliffs, New Jersey (1989)
- [2] S.G. Akl and G.R. Guenther, Broadcasting with Selective Reduction, Information Processing 89, Proceedings of the IFIP 11th World Computer Congress, San Francisco, G.X. Ritter, Ed. North-Holland (1989) 515-520.

- [3] S.G. Akl and K. Qiu, Parallel Point Location Algorithms on Hypercubes, *10th Intern. Conf. Paralle. Distr. Comput. Syst.(PDCS-97)* (1997) 27–30.
- [4] S.G. Akl and I. Stojmenovic, Multiple criteria BSR : An implementation and applications to computational geometry problems, *Proceedings of the Twenty-Seventh Annual Hawaii International Conference on System Sciences* (1994).
- [5] S.G. Akl and G.T. Toussaint, Efficient convex hull algorithms for pattern recognition applications, *Proc. 4th Int' Joint Conf. on Pattern Recognition, Kyoto, Japan* (1978) 483–487.
- [6] E. Delacourt, J.-F. Myoupo and D. Semé, A Constant Time Parallel Detection of Repetition, *Parallel Processing Letters*, (1999) 9(1):81–92.
- [7] R.O. Duda and P.E. Hart, *Pattern Classification and Scene Analysis*, Wiley-Interscience, New York (1973).
- [8] L. Devroye, E.P. Mucke and B. Zhu, A note on point location in Delaunay triangulations of random points, *Submitted to IPL* (1995).
- [9] P. Erdos and A. Szekers, A combinatorial problem in geometry, *Compositio Mathematica* (1935) 2:463–470.
- [10] H. Freeman, Computer processing of line-drawing images, *Comput. Surveys* (1974) 6:57–97.
- [11] H. Freeman and R. Shapira, Determining the minimum-area encasing rectangle for an arbitrary closed curve, *Comm. ACM* (1975) 18(7):409–413.
- [12] A. Gibbons and W. Rytter, *Efficient Parallel Algorithms*, Cambridge University Press, Cambridge, England (1988)
- [13] J. Jang, M. Nigam, V.K. Prasanna and S. Sahni, Constant Time Algorithms for Computational Geometry on the Reconfigurable Mesh, *IEEE Trans. on Paralle. and Dist. syst.* (1997) 8(1):1–12.
- [14] U. Manber, *Introduction to algorithms, a creative approach*, Adisson-Wesley (1989).
- [15] E.P. Mucke, Shapes and Implementations in Three-Dimensional Geometry, Ph. D. thesis, Technical Report UIUCDCS-R-93-1836, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Illinois (1993).
- [16] J.-F. Myoupo and D. Semé, A Parallel Solution of the Sequence Alignment Problem using BSR Model, *10th Intern. Conf. Paralle. Distr. Comput. Syst.(PDCS-97)*, (1997).
- [17] J.-F. Myoupo and D. Semé, Efficient BSR-Based Parallel Algorithms for Geometrical Problems, *Technical Report 99-04, Université de Picardie Jules Verne*, (1999).
- [18] J.-F. Myoupo and D. Semé, Time-Efficient Parallel Algorithms for the Longest Common Subsequence and Related Problems, *Journal of Parallel and Distributed Computing*, (1999) 57:212–223.
- [19] J.-F. Myoupo and D. Semé, Efficient Parallel Algorithms for the LIS and LCS Problems on BSR Model using bounded number of selections, *Parallel Algorithms and Applications*, (2000) 14:187–202.
- [20] J.-F. Myoupo and D. Semé and I. Stojmenovic, Optimal BSR Solutions to Several Convex Polygon Problems, (*submitted*), (1999).
- [21] F.P. Preparata and M.I. Shamos, *Computational Geometry : an introduction*, Springer-Verlag, New York, (1985).
- [22] A. Rosenfeld, *Picture Processing by Computers*, Academic Press, New York, (1969).
- [23] D. Semé, An Efficient Algorithm on the BSR-Based Parallel Architecture for the k-LCS Problem, *Proc. of the Intern. Conf. on Paralle. and Distr. Process. Techn. and Appli., PDPTA-99*, (1999).
- [24] J. Sklansky, Measuring concavity on a rectangular mosaic, *IEEE Trans. Comp.* (1972) C(21):1355–1364.
- [25] I. Stojmenovic, Constant Time BSR Solutions to Parenthesis Matching, Tree Decoding, and Tree Reconstitution From Its Traversals, *IEEE Transactions on Parallel and Distributed Systems* (1996) 7(2):218–224