Parallel Convex Hull Construction

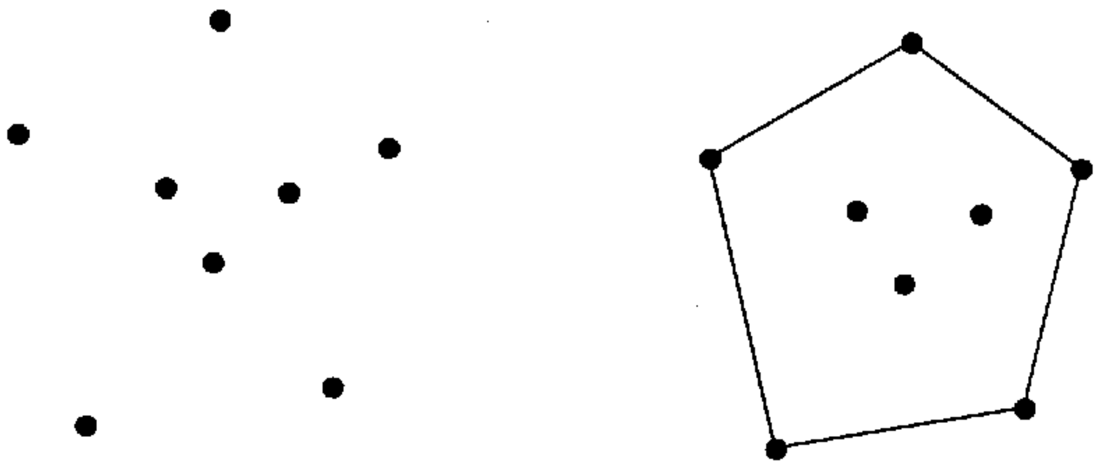
Harald Deischinger
deischi@geocities.com

June 21, 1998

Implementation of a parallel algorithm for constructing the convex hull of a set of planar points using DAJ (Distributed Algorithms in Java).

Convex Hull – The Problem

- given a set $S = \{p_1, p_2, \ldots, p_n\}$ of points in the plane
- find the smallest convex polygon $\mathtt{CH}(S)$ that includes all points of S



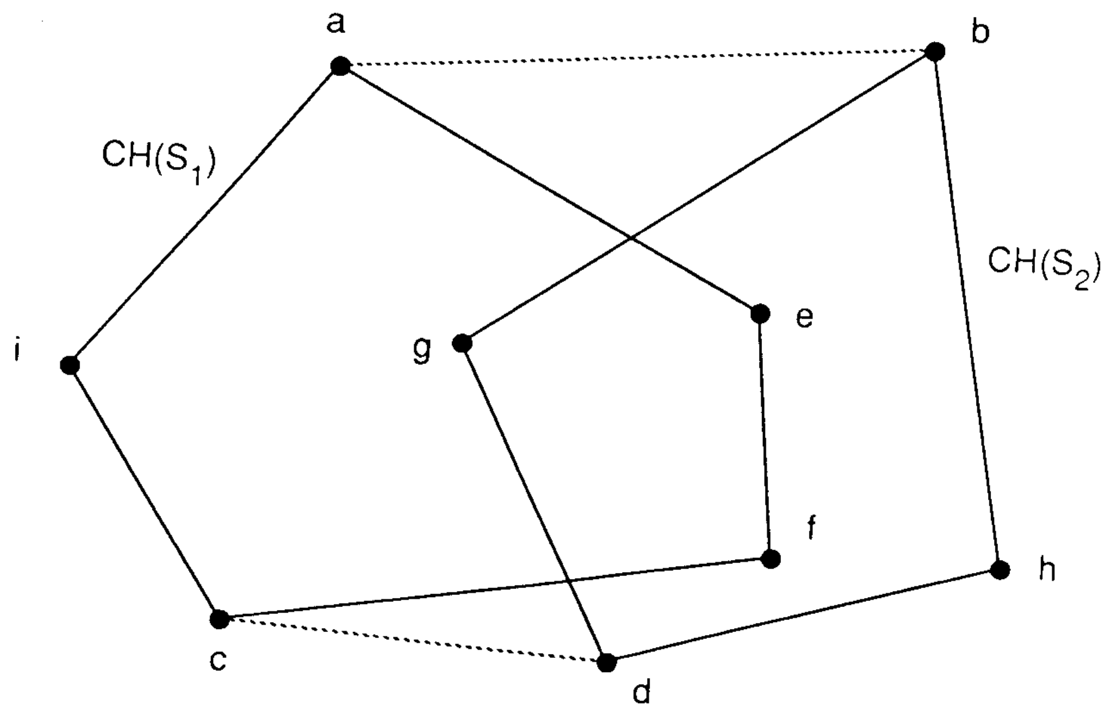
(a) (b)

Sequential Algorithm

- Divide and Conquer Algorithm

  $|S| \leq 3$ then $\texttt{CH}(S) = S$, else

  - Divide S into S_1 and S_2, s.t. both have equal size
  - Compute $\texttt{CH}(S_1)$ and $\texttt{CH}(S_2)$
  - Merge $\texttt{CH}(S_1)$ and $\texttt{CH}(S_2)$ into $\texttt{CH}(S)$ using an upper tangent (a, b) and a lower tangent (c, d)



- Runtime of $O(n \log n)$

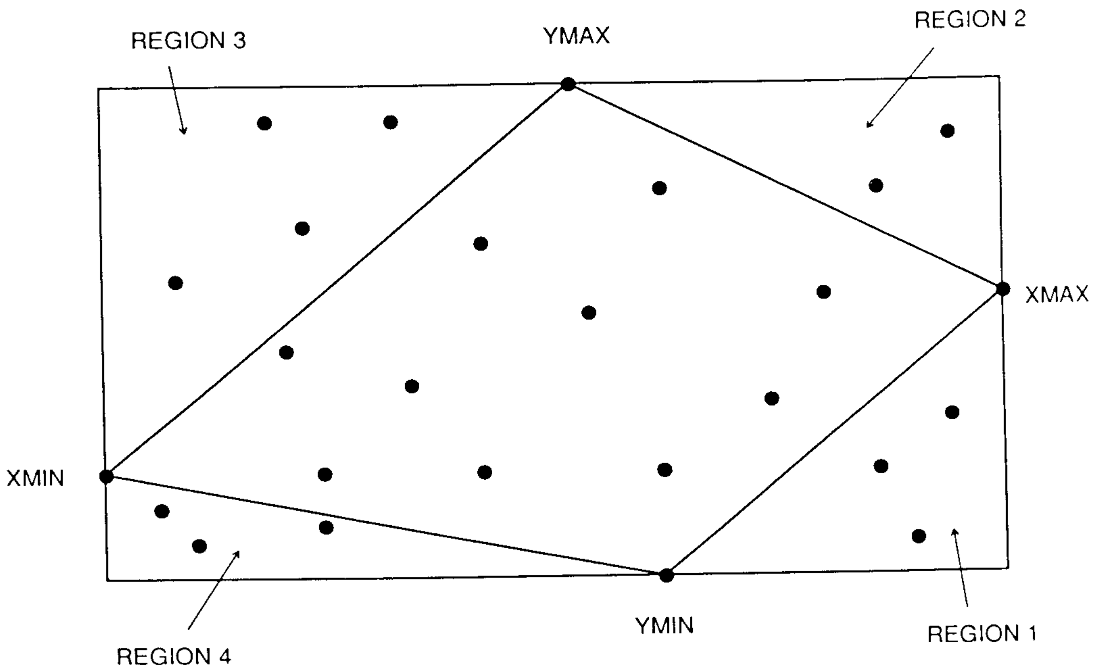
A Parallel Algorithm

- Assumptions
  - no two points have same x coordinate
  - no three points fall on the same straight line
- Three central ideas (properties of the convex hull)
  - Identifying Extreme Points
  - Identifying Hull Edges
  - Identifying the Smallest Angle

Identifying Extreme Points

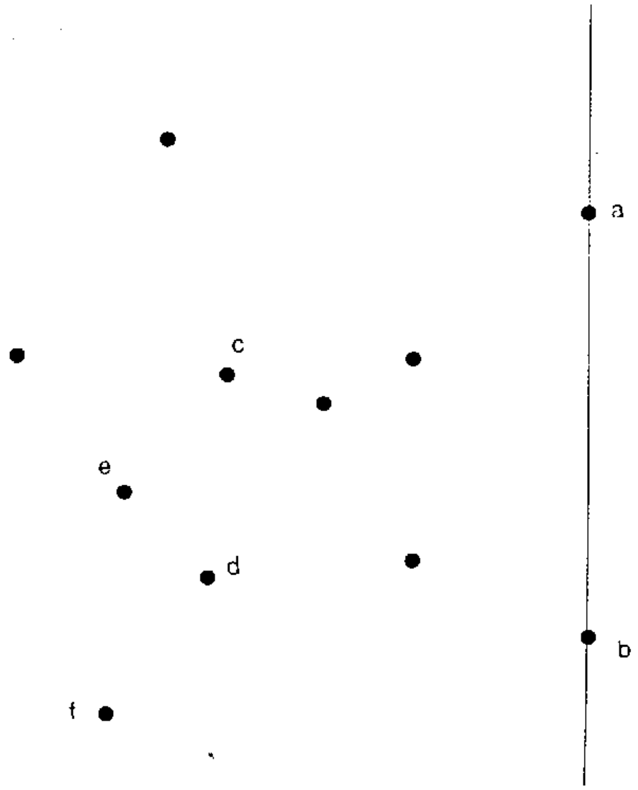
Find points with minimum x, minimum y, maximum x and maximum y coordinate.

- The extreme points are vertices of $\mathtt{CH}(S)$
- All points falling within the quadrilateral formed by the extreme points are not vertices of $\mathtt{CH}(S)$
- Regions can be investigated independently



Identifying Hull Edges

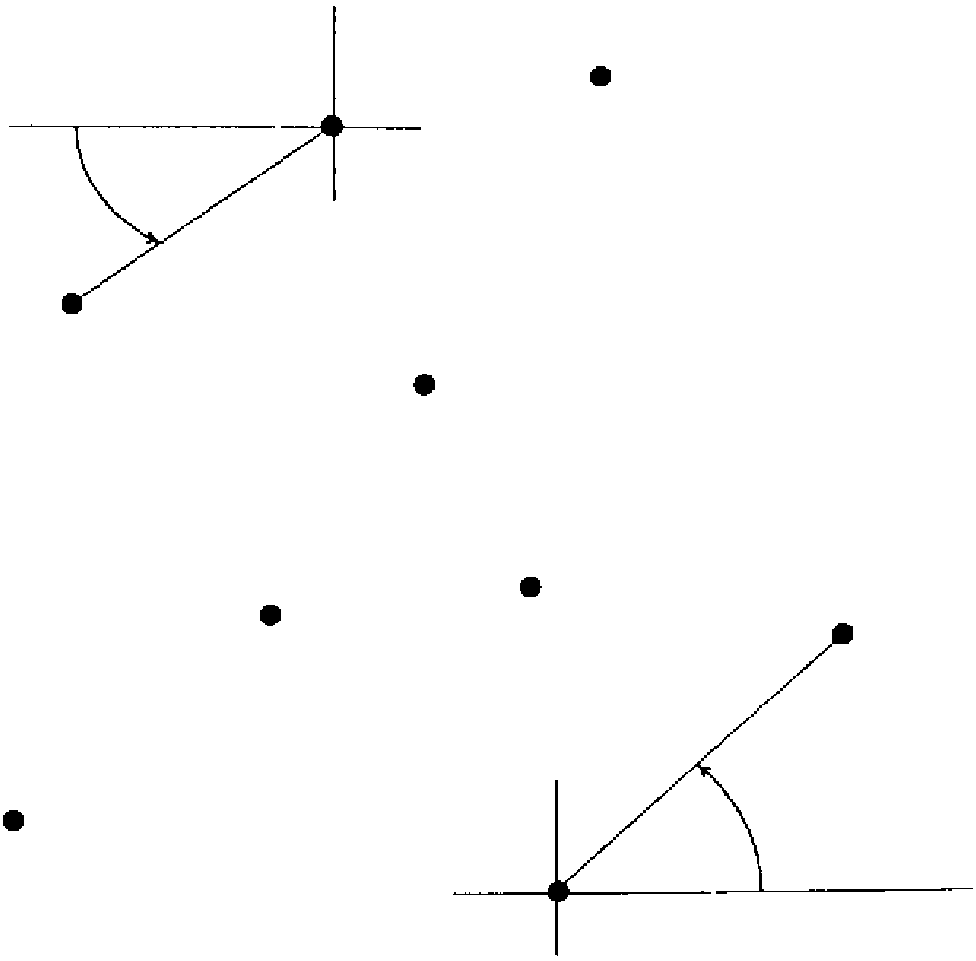
An edge (p_i, p_j) is an edge of $\texttt{CH}(S)$ if and only if all the remaining points fall on one side of the infinite line defined by (p_i, p_j).



Identifying the Smallest Angle

Let p_i and p_j be consecutive vertices of $\mathtt{CH}(S)$. And let p_i be an origin for polar coordinates.

Then, among all points in S, p_j forms the smallest angle w.r.t. the positive (or negative) x-axis



Interconnection Structure and Initial Data Layout

- Mesh of Trees: "Array" of N by N Nodes. Connections within rows and columns using trees.

  Roots are in first row/column.

- Communication is implemented using directed channels. There are up to 6 input and the same number of output channels per node

- Intially the nodes in row 0 store the elements of S. Each node $(x, 0)$ stores a set S_x, s.t. S_x are a partition of S and S_x have approximately equal size.

Stage 1 – Extreme Points

- S is sent through column connections to nodes in first 4 rows

- Using row connections extreme points are formed. Row 0 searches xmin, row 1 searches xmax, row 2 searches ymin and row 3 searches ymax. Result of these reduction operations in column 0.

- Collect the extreme points in node $(0, 0)$ Node $(0, 0)$ adds xmin, xmax, ymin, ymax to $\texttt{CH}(S)$.

- Broadcast extreme points to all nodes (first within the first row, then along the columns)

Stage 2 – Interior Elimination, Region Identification

- In row 0: All nodes identify for all local points the region $r \in \{1, 2, 3, 4\}$ it belongs to. From that information each node creates 4 (possibly empty) region set R_i^r.

  Interior points are discarded

  Extreme points are not part of any region.

- The region sets R_i^r are broadcasted along column connections to all nodes withing a column.

Stage 3 – Identifying Convex Hull Elements

For region $r \in \{1, 2, 3, 4\}$. Only row operations.

- Send region set R_d^r from nodes in the diagonal of the mesh to the root and broadcast afterwards

  That way all rows of the mesh work on a different subset of S.

- For all points $p_i \in R_d^r$
  - Find point $p_j \in R_x^r \cup \{\text{right extreme point}\}$ s.t. p_j is right of p_i and edge (p_i, p_j) has minimal (for region 1,4) or maximal (for region 2,3) slope.
    Requires one reduce and broadcast operation.
  - Test if all nodes $p_t \in R_x^r \cup \{\text{left extreme point}\}$, $p_t \neq p_i$, $p_t \neq p_j$ fall on one side of (p_i, p_j) If yes: Node in column 0 adds p_i to $\mathtt{CH}(S)$.
    Requires one reduce operation.

Stage 4 – Collect Results and Sort

- In column 0: Select (x_0, y_0) a point in the quadrilateral formed by the extreme points
- In column 0: Compute the angle in polar coordinates of all points of $\texttt{CH}(S)$ w.r.t. (x_0, y_0) as origin.
- Sort $\texttt{CH}(S)$ w.r.t. the angles:
  - Implementation: Collect all elements of $\texttt{CH}(S)$ and the previously computed angle in node $(0, 0)$. Sort sequentially.
  - Should be: Use the mesh of trees structure for sorting.

Runtime Complexity Considerations

- For $N = n$: All steps take $O(\log n)$ operations. But the algorithm needs n^2 processors and is with a cost of $O(n^2 \log n)$ not optimal.

  There exists an optimal algotihm for shared memory systems.

- For $N = 1$: The algorithm takes $O(n^2)$ operations (in Stage 3).

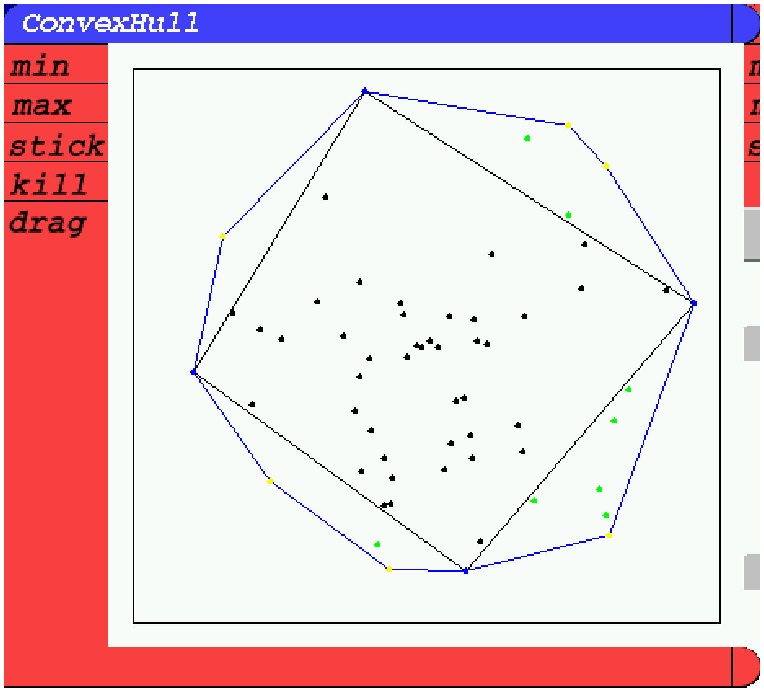
Implementation – Usage

- Command line arguments: n and N, e.g. `ConvexHull.class 60 6`

- Requires that $N \geq 4$ (Step 1)

- Initial S is randomly distributed in a circle around $(0.5, 0.5)$ of radius 0.5.

- Opens an extra window to draw some intermediate steps (extreme points) and the result



Implementation – Experiences with DAJ

- DAJ offers only communication through channels. Need to write non-neighbor communication, reduce and broadcast operations.
- When accessing channels using `in()` and `out()`: channels are only numbered – it is important in what order the channels were created.
- DAJs send is non-blocking. Send buffer must not be modified.
- Necessary to write wrapper classes for sending basic data types, e.g. `IntMessage`.