Block Sorting is Hard

Wolfgang W. Bein
Lawrence L. Larmore
University of Nevada, Las Vegas
Department of Computer Science

I. Hal Sudborough
University of Texas at Dallas
Department of Computer Science

Shahram Latifi
University of Nevada, Las Vegas
Department of Electrical and Computer Engineering

May 2002
What is Block Sorting?

How did they do it?
Formalizing or Example:
- How did they do it?

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A F C B E D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>How ? they did it do</td>
<td></td>
</tr>
<tr>
<td>A C B E F D</td>
<td>move F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>How they did it ? do</td>
<td></td>
</tr>
<tr>
<td>A C B E D</td>
<td>and delete</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>How they did do</td>
<td></td>
</tr>
<tr>
<td>A B C E D</td>
<td>move C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>How did they do</td>
<td></td>
</tr>
<tr>
<td>A E D</td>
<td>and delete</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>do</td>
<td></td>
</tr>
<tr>
<td>A D E</td>
<td>move D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>do</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>and delete</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>do</td>
<td></td>
</tr>
</tbody>
</table>
Reversals

The number of reversals gives a lower bound on the number of moves
Red edge between two elements (Hal, Kim):

There is a sorting where Hal and Kim are joined before either is moved.
There is a perfect sorting

\[\iff\]

There exists a spanning tree of blue and red edges
There is not always a Perfect Sorting

Red Edges must not “overlap” in either order, i.e. horizontally or vertically
The problem of whether there exists a perfect sorting is \( \mathcal{NP} \)-complete.

Proof: 3-SAT can be reduced to Perfect Sorting. ■

Block Sorting is \( \mathcal{NP} \)complete.
\[(\bar{x}_3 + x_2 + x_1)(x_4 + \bar{x}_3 + \bar{x}_2)\]
Since Block Sorting is $\mathcal{NP}$-complete, what can be done?

- Identify “good” moves
- Get numerous lower bounds
- Get a “competitive” algorithm
Primal Sorting Problem

Ann Bob Dan Eve Hal Joe Kim Sam Tom
"primal–dual move"

"primal obstruction"

"dual obstruction"
removal of $u$ creates a primal obstruction at $v$

removal of $u$ creates a dual obstruction at $v$

removal of $u$ creates a dual obstruction at $v$
Local Property Algorithm

For each element (black dot) keep four pointers

\[ x.\text{left} \quad x.\text{right} \quad x.\text{down} \quad x.\text{up} \]

<table>
<thead>
<tr>
<th>Type</th>
<th>Condition</th>
<th>Effect of Deletion</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>x.r.u = x</td>
<td>( n' \leq n - 1 )</td>
</tr>
<tr>
<td>B</td>
<td>x.d.r.d = x</td>
<td>( n' \leq n - 2 )</td>
</tr>
<tr>
<td>C</td>
<td>x.l.u.l = x</td>
<td>( n' \leq n - 2 )</td>
</tr>
<tr>
<td>D</td>
<td>x.l.u.l.l = x</td>
<td>( n' \leq n - 1 ), ( x.r ) becomes type C</td>
</tr>
<tr>
<td>E</td>
<td>x.d.r.d.d = x</td>
<td>( n' \leq n - 1 ), ( x.u ) becomes type B</td>
</tr>
<tr>
<td>F</td>
<td>x.l.u.u.l = x</td>
<td>( n' \leq n - 1 ), ( x.l.u ) becomes type B</td>
</tr>
</tbody>
</table>

- Can be implemented in \( O(1) \) per step

- Overall: \( O(n \log n) \)
A lower bound

\[ \begin{align*}
    k & \quad \text{blue edges} \\
    l & \quad \text{green edges} \\
\end{align*} \]

\[ \# \text{ moves} \geq \max \left\{ \frac{n - k - \ell - 1}{k}, \frac{n - k - \ell - 1}{\ell} \right\} \]

\[ \Rightarrow \]

\[ \# \text{ moves} \geq \left\lceil \frac{n - 1}{3} \right\rceil \]
Odd-Cycle Lower Bound

\[
\text{\# moves} \geq \frac{1}{2} (n - 1 - \text{oddcycles})
\]
Competitiveness

How good does an algorithm behave in the worst case compared to an optimal scheme?

Local Property algorithms are 3 - competitive

Open Problem:

Is there a polynomial-time 2 - competitive algorithm?