Sorting Omega Networks Simulated with P Systems: Optimal Data Layouts

Rodica Ceterchi, Mario J. Pérez Jiménez, Alexandru I. Tomescu

August 2008

Bitonic Sorting

- one input key on each wire
- ascending / descending comparators between wires
- sorts $N$ keys in $O(\log^2 N)$ time
Each wire has one associated value

One wire can communicate with any other wire

Time complexity of the original sorting network, $O(\log^2 N)$

We want to sort in ascending order the sequence of distinct integers

\[
\langle x_0, x_1, \cdots, x_{N-1}\rangle,
\]

codified as the multiset

\[
\mathcal{W} = v_0^{x_0} v_1^{x_1} \cdots v_{N-1}^{x_{N-1}}.
\]
One membrane

- ascending comparator
  \[ C_\oplus = \{ v_0 \rightarrow a, v_1 \rightarrow b \} \cup \{ ab \rightarrow c^+ d^+, a \rightarrow d^+, b \rightarrow d^+ \} \cup \{ c^+ \rightarrow v_0, d^+ \rightarrow v_1 \}. \]

- descending comparator
  \[ C_\ominus = \{ v_0 \rightarrow a, v_1 \rightarrow b \} \cup \{ ab \rightarrow c^- d^-, a \rightarrow c^-, b \rightarrow c^- \} \cup \{ c^- \rightarrow v_0, d^- \rightarrow v_1 \}. \]

\[ v_0^{x_0} v_1^{x_1} \rightarrow a^{x_0} b^{x_1} \rightarrow c^{+ \min(x_0,x_1)} d^{+ \max(x_0,x_1)} \rightarrow v_0^{\min(x_0,x_1)} v_1^{\max(x_0,x_1)} \]

\[ S^+ = \{ s_0^+, \ldots, s_{2^k-1}^+ \}, S^- = \ldots \]

\[ T^+ = \{ t_0^+, \ldots, t_{2^k-1}^+ \}, T^- = \ldots \]
Rewrite all symbols of $V$ into start symbols for appropriate comparators, using the sets of rules

\[ \{ v_{2j} \rightarrow s_{2j}^+, v_{2j+1} \rightarrow s_{2j+1}^+ | 0 \leq j \leq 2^{2k-1} - 1, j \text{ even} \} \cup \\
\cup \{ v_{2j} \rightarrow s_{2j}^-, v_{2j+1} \rightarrow s_{2j+1}^- | 0 \leq j \leq 2^{2k-1} - 1, j \text{ odd} \} \].

Apply in parallel the rewritings of symbols which correspond to the simulations of the comparators:

\[ \{ s_{2j}^+ s_{2j+1}^+ \rightarrow t_{2j}^+ t_{2j+1}^+, s_{2j}^+ \rightarrow t_{2j+1}^+, s_{2j+1}^+ \rightarrow t_{2j+1}^+ | 0 \leq j \leq 2^{2k-1} - 1, j \text{ even} \} \cup \\
\cup \{ s_{2j}^- s_{2j+1}^- \rightarrow t_{2j}^+ t_{2j+1}^-, s_{2j}^- \rightarrow t_{2j}^-, s_{2j+1}^- \rightarrow t_{2j}^- | 0 \leq j \leq 2^{2k-1} - 1, j \text{ odd} \} \].

Rewrite back all symbols of $T$'s into $V$.

\[ \{ v_{2j} \leftarrow t_{2j}^+, v_{2j+1} \leftarrow t_{2j+1}^+ | 0 \leq j \leq 2^{2k-1} - 1, j \text{ even} \} \cup \\
\cup \{ v_{2j} \leftarrow t_{2j}^-, v_{2j+1} \leftarrow t_{2j+1}^- | 0 \leq j \leq 2^{2k-1} - 1, j \text{ odd} \} \].
One membrane for each wire

- use a P System with Dynamic Communication Graphs
- for an input of size $N$, take $N$ membranes each holding two objects / values
- membranes are disposed on a 2D mesh architecture $\sqrt{N} \times \sqrt{N}$ (each membrane can communicate only with its four vertical and horizontal neighbors)
- use the *shuffled row-major* indexing on the mesh to minimize communication
- time complexity $O(\sqrt{N})$, optimal for the 2D mesh
A $P$ system with dynamic communication graphs is a tuple

$$\Pi = \langle V, P_0, \cdots, P_{k-1}, R_\mu = \{R_i, G_i\}_{i \in I} \rangle,$$

where:

- $V$ is an alphabet of symbols (used to codify integers contained in membranes).
- $P_0, \cdots, P_{k-1}$ are elementary membranes.
- $R_\mu = \{R_i, G_i\}_{i \in I}$ is a sequence of pairs $[rules, graph]$.  
  - If $graph \subseteq G_{ld}$, its rules are rewriting rules.
  - If $graph \subseteq G_{total} \setminus G_{ld}$, its rules are communication rules.

A $P$ system with dynamic communication graphs with finite sequential support if $R_\mu$ is a finite sequence.
Bitonic sorting on a 2D mesh of $4 \times 4$

Stage 1

$t = 1$

Stage 2

$t = 2$

Stage 3

$t = 3$

Stage 4

$t = 4$
Bitonic sorting on a 2D mesh of $4 \times 4$

Stage 4

$t = 4$

$t = 3$

$t = 2$

$t = 1$

Result

Rodica Ceterchi, Mario J. Pérez Jiménez, Alexandru I. Tomescu
Sorting Omega Networks Simulated with P Systems: Optimal Data Layouts
Algorithm 5: Compare-interchange operation for an array of neighbour membranes associated to processors situated on the same line of the mesh

**Input:** array $a$ of membrane indices, integer $i$, and sorting order $\text{order}$

**Output:** the sequence of integers stored in the sequence of membranes $\langle a[0], a[1], \ldots, a[2^{i-1}-1] \rangle$ is ordered w.r.t. $\text{compare-interchange-membr}(a, i, \text{order})$

forall $j \leftarrow 0$ to $2^{i-1} - 1$ in parallel do

// route right one unit in the $B$ registers - (rAB) rule

$E(Id_1^t) \leftarrow E(Id_1^t) \cup \{(j, j)\}$; $\text{rules}_{0,1}^{t}((j, j)) \leftarrow \{a \rightarrow a^*\}$

$E(G_0^t) \leftarrow E(G_0^t) \cup \{(j, j+1)\}$; $\text{rules}_{0}^{t}((j, j+1)) \leftarrow \{(a^*, \text{out})\}$

$E(Id_2^t) \leftarrow E(Id_2^t) \cup \{(j+1, j+1)\}$;

$\text{rules}_{0,2}^{t}((j+1, j+1)) \leftarrow \{a^* \rightarrow b\}$

for $k \leftarrow 1$ to $2^{i-1} - 1$ do

// shift the values to the second half of the array - (rBB) rule

forall $j \leftarrow 0$ to $2^{i-1} - 1$ in parallel do

$E(G_k^t) \leftarrow E(G_k^t) \cup \{(j+k, j+1+k)\}$

$\text{rules}_{k}^{t}((j+k, j+1+k)) \leftarrow \{(b, \text{out})\}$

forall $j \leftarrow 2^{i-1}$ to $2^i - 1$ in parallel do

// compare internally - (C) rule

$E(Id_C) \leftarrow E(Id_C) \cup \{(j, j)\}$

if $\text{order}$ is ascending then

$\text{rules}_{C}^{t}((j, j)) \leftarrow \{ab \rightarrow ab, a \rightarrow b, b \rightarrow b\}$

else

$\text{rules}_{C}^{t}((j, j)) \leftarrow \{ab \rightarrow ab, a \rightarrow a, b \rightarrow a\}$

for $k \leftarrow 2^{i-1} - 1$ downto 1 do

// shift back the results - (rBB) rule

forall $j \leftarrow 0$ to $2^{i-1} - 1$ in parallel do

// final routing back in the $A$ registers - (rBA) rule
Analysis of proposed models

Advantages:

▶ preserve the time complexity of the original parallel architecture

Disadvantages:

▶ the evolution rules depend on the size of the input
▶ the number of membranes depends on the size of the input
▶ the communication graphs depend on the size of the input
▶ big communication overhead
Suppose we have a system of fixed size $n$

- to sort $N$ numbers use $P = N/n$ such systems and “combine” the results
- the evolution rules do not depend on the size of the input
- “combination” of results is done according to easy-to-compute communication graphs
Framework:

- sort $N$ values using $P = N/n$ systems, according to a bitonic sorting algorithm ($N > P$)
- we call data layout a function $D : \{0, \ldots, N - 1\} \rightarrow \{0, \ldots, P - 1\}$.
- $D(i) = j$ iff wire $i$ is mapped to system $j$.
- comparisons are allowed only between wires assigned to the same system

Goal:

- find a sequence of data layouts such that communication between systems is minimized
Sequences of data layouts at work

- at each step $s$ of the omega network we have comparators between wires differing on bit $s$
at each step $s$ of the omega network we have comparators between wires differing on bit $s$

the 1st data layout: wire $i = a_1 \ldots a_n a_{n+1} \ldots a_{\log N}$ is mapped to system $j = a_{n+1} \ldots a_{\log N}$
 Sequences of data layouts at work

- at each step $s$ of the omega network we have comparators between wires differing on bit $s$
- the 1st data layout: wire $i = a_1 \ldots a_n a_{n+1} \ldots a_{\log N}$ is mapped to system $j = a_{n+1} \ldots a_{\log N}$
- perform log $n$ steps of the sorting algorithm internally
Sequences of data layouts at work

- at each step $s$ of the omega network we have comparators between wires differing on bit $s$
- the 1st data layout: wire $i = a_1 \ldots a_n a_{n+1} \ldots a_{\log N}$ is mapped to system $j = a_{n+1} \ldots a_{\log N}$
- perform $\log n$ steps of the sorting algorithm internally
- remap values according to the 2nd data layout: wire $i = a_1 \ldots a_n a_{n+1} \ldots a_{2n} a_{2n+1} \ldots a_{\log N}$ is mapped to system $j = a_1 \ldots a_n a_{2n+1} \ldots a_{\log N}$
Sequences of data layouts at work

- at each step $s$ of the omega network we have comparators between wires differing on bit $s$
- the 1st data layout: wire $i = a_1 \ldots a_n a_{n+1} \ldots a_{\log N}$ is mapped to system $j = a_{n+1} \ldots a_{\log N}$
- perform $\log n$ steps of the sorting algorithm internally
- remap values according to the 2nd data layout: wire $i = a_1 \ldots a_n a_{n+1} \ldots a_{2n} a_{2n+1} \ldots a_{\log N}$ is mapped to system $j = a_1 \ldots a_n a_{2n+1} \ldots a_{\log N}$
- sort internally
Sequences of data layouts at work

- at each step $s$ of the omega network we have comparators between wires differing on bit $s$
- the 1st data layout: wire $i = a_1 \ldots a_n a_{n+1} \ldots a_{\log N}$ is mapped to system $j = a_{n+1} \ldots a_{\log N}$
- perform $\log n$ steps of the sorting algorithm internally
- remap values according to the 2nd data layout: wire $i = a_1 \ldots a_n a_{n+1} \ldots a_{2n} a_{2n+1} \ldots a_{\log N}$ is mapped to system $j = a_1 \ldots a_n a_{2n+1} \ldots a_{\log N}$
- sort internally
- and so on ...
Rodica Ceterchi, Mario J. Pérez Jiménez, Alexandru I. Tomescu  Sorting Omega Networks Simulated with P Systems: Optimal Data Layouts
Let $D_k$, $k \geq 1$ be a one-step network of $N = 2^k$ lines with a device between the pair of lines $(i, i + N/2)$, for $i = 0 \ldots N/2 − 1$. Then the omega network $OM_k$ is recursively defined as

$$OM_k = D_k(OM_{k-1} \circ OM_{k-1}).$$

![Diagram of Omega network](image)
We have to sort $N = 2^k$ keys using $P$ processors, $N > P$, each processor holding $n = N/P$ keys.

The number of parallel steps of $OM_k$ is $k$, and step $t$ of the omega network $OM_k$ contains devices linking lines whose bit representations differ of bit $t$, with $1 \leq t \leq k$.

$bc_t : \{0, 1, \ldots, 2^k - 1\} \rightarrow \{0, 1, \ldots, 2^k - 1\}$, the bit complement of the $t$-th bit,

$$bc_t(a_1a_2\cdots a_t\cdots a_k) = a_1a_2\cdots \bar{a}_t\cdots a_k$$
An omega network of size 32. Three data layouts for the omega network $OM_5$.

(c) An omega network of size 32. Lines marked with same shape are assigned to the same processor in one data layout.

(d) Keys mapped to processor 0.
More formally...

Lemma
At each step $1 \leq t \leq k$ of $OM_k$, and for any $0 \leq i < 2^k$, line $i$ is linked by a device only with line $bc_t(i)$.

Lemma
In $OM_k$, for any $0 \leq i < 2^{k-m}$, $1 \leq m \leq k$ and $0 \leq t \leq k - m$, in steps $t + 1, \ldots, t + m$ there is no device linking lines in the set $P_{i}^{t,m} = \{a_1a_2\cdots a_k \mid a_1 \cdots a_t a_{t+m+1} \cdots a_k = i, \text{ where } a_1 \cdots a_k \text{ is a bit representation} \}$ with lines from $\{0, \ldots, 2^k - 1\} \setminus P_{i}^{t,m}$. 
Definition
Given $N = 2^k$ keys and $P = 2^{k-m}$ processors, which can store $n = 2^m$ values, $m \geq 1$, the sequence of optimal data layouts consists of $\lceil \log N / \log n \rceil = \lceil k/m \rceil$ data layouts. In each data layout $D_s$, $0 \leq s \leq \lceil k/m \rceil - 1$, values in the set $P_i^{sm,m}$ are mapped to processor $P_i$, for all $0 \leq i \leq 2^{k-m}$. More formally, for any $0 \leq u < 2^k$ such that $u \in P_i^{sm,m}$, we have $D_s(u) = i$.

Lemma
The maximum number of successive steps of the omega network that can be executed locally, under any data layout is $\log n$, where $n = N/P$. 

Rodica Ceterchi, Mario J. Pérez Jiménez, Alexandru I. Tomescu
Sorting Omega Networks Simulated with P Systems: Optimal
Inside one processor/membrane

Inside one processor, several comparisons are performed, in parallel, between the $n$ pieces of data

\[
\text{for } t \leftarrow 1 \textbf{ to } m \textbf{ do} \\
\quad \text{forall } i < bc_t(i) \textbf{ in parallel do} \\
\quad \quad \text{compare}(a_i, a_{bc_t(i)});
\]

**Algorithm 1**: A parallel algorithm for the bitonic merger
Inside one membrane

The parallel comparisons at each step $t$

forall $i < bc_t(i)$ in parallel do
\[ \text{compare}(a_i, a_{bc_t(i)}) ; \]

will be simulated in a membrane $P$ by the rules

\[
\{a_i \rightarrow s_i \mid i = 0, 1, \ldots, n-1\} \cup \\
\cup \{s_is_j \rightarrow t_it_j, s_i \rightarrow t_j, s_j \rightarrow t_j \mid i = 0, 1, \ldots, n-1, i < j = bc_t(i)\} \cup \\
\cup \{t_i \rightarrow a_i \mid i = 0, 1, \ldots, n-1\}.
\]
A P System with dynamic communication graphs which simulates the Omega Network

\[ \Pi = < V = \{ a_0, \ldots, a_{n-1} \} \cup A, \]

\[ \left< [a_0^0, a_1^0, \ldots, a_{n-1}^0]_0, \ldots, [a_0^{P-1}, a_1^{P-1}, \ldots, a_{n-1}^{P-1}]_{P-1} \right>, R_\mu > \]

\( R_\mu \) sequence of pairs [graph, rules]. \( R_\mu \) is generated algorithmically.
Lemma

Given $N = 2^k$ keys and $P = 2^{k-m}$ membranes, which can store $n = 2^m$ values, $m \geq 1$, after the computation for the data layout $\mathcal{D}_s$ is finished, symbol $a_i$ of membrane $j$ codifies the value corresponding to wire $u \in \{0, \ldots, N-1\}$, where the bit representation of $u$ is $u = j_1 \ldots j_{sm} i_1 \ldots i_m j_{sm+1} \ldots j_{k-m}$. By $j_1 \ldots j_{k-m}$ and by $i_1 \ldots i_m$ we denoted the bit representations of $j$, and $i$, respectively.
Algorithmic generation of the communication graph

\[ E(C^j_s) \leftarrow \emptyset ; \]

\textbf{for} \( j \leftarrow 0 \) \textbf{to} \( P - 1 \) \textbf{do}

\begin{itemize}
  \item \textbf{for} \( i \leftarrow 0 \) \textbf{to} \( n - 1 \) \textbf{do}
  \begin{itemize}
    \item let \( j \) have bit representation \( j_1 \cdots j_{smj_{sm+1}} \cdots j_{k-m} \);
    \item let \( i \) have bit representation \( i_1 \cdots i_m \);
    \item // the destination membrane of value encoded by \( a_i \) in membrane \( j \)
    \item \( z \leftarrow j_1 \cdots j_{sm} i_1 \cdots i_m j_{(s+1)m+1} \cdots j_{k-m} \);
    \item // the destination symbol of value encoded by \( a_i \) in membrane \( j \)
    \item \( t \leftarrow j_{sm+1} \cdots j_{sm+m} \);
    \item \( E(C^j_s) := E(C^j_s) \cup \{(j, z)\} \);
    \item \( \text{rules}_{C^j_s}((j, z)) := a_i \rightarrow a'_t \);
  \end{itemize}
\end{itemize}

\textbf{Algorithm 2:} Generation of the sequence of \( P \) communication graphs when passing from data layout \( D_{s-1} \) to \( D_s \), with \( 0 < s \leq \lceil \frac{k}{m} \rceil - 1 \).
Algorithmic generation of internal computation rules

\[ SimOM \leftarrow \lambda; \]
\[ \text{for } t \leftarrow 1 \text{ to } m = \log n \text{ do} \]
\[ \quad \text{forall } p \leftarrow 0 \text{ to } P - 1 \text{ in parallel do} \]
\[ \quad \quad \text{rules}_{t,1}((p, p)) \leftarrow \{a_i \rightarrow s_i \mid i = 0, 1, \ldots, n - 1\}; \]
\[ \quad \quad \text{rules}_{t,2}((p, p)) \leftarrow \{s_i s_j \rightarrow t_i t_j, s_i \rightarrow t_j, s_j \rightarrow t_j \mid i = 0, 1, \ldots, n - 1, i < j = bc_t(i)\}; \]
\[ \quad \quad \text{rules}_{t,3}((p, p)) \leftarrow \{t_i \rightarrow a_i \mid i = 0, 1, \ldots, n - 1\}; \]
\[ \quad SimOM \leftarrow SimOM \cdot [Id, \text{rules}_{t,1}] \cdot [Id, \text{rules}_{t,2}] \cdot [Id, \text{rules}_{t,3}]; \]

**Algorithm 3**: Generation of the sequence \( SimOM \) which simulates the omega network of size \( n \).
Algorithmic generation of $R_\mu$

\[ R_\mu \leftarrow \lambda; \]
\[ \text{for } s \leftarrow 1 \text{ to } \lceil k/m \rceil - 1 \text{ do} \]
\[ R_\mu \leftarrow R_\mu \cdot \text{SimOM}; \]
\[ \text{for } j \leftarrow 0 \text{ to } P - 1 \text{ do} \]
\[ R_\mu \leftarrow R_\mu \cdot [C_j^s, \text{rules}_{C_j^s}]; \]
\[ R_\mu \leftarrow R_\mu \cdot [\text{Id}, \text{rules-endcomm}]; \]
\[ R_\mu \leftarrow R_\mu \cdot \text{SimOM}; \]

**Algorithm 4**: Generation of the sequence $R_\mu$ which guides the computation.
Computation Complexity

- we have \( \frac{\log N}{\log n} \) data layouts;
- in each data layout we have
  - 3\( \log n \) steps are needed for SimOM;
  - \( P + 1 = \frac{N}{n} + 1 \) steps are needed for communication;
- hence the length of \( R_\mu \) is \( 3 \log N + \frac{N \log N}{n \log n} \);
- a sorting network can be obtained by a serial connection of \( \log N \) omega networks, giving a time complexity of

\[
O(\log^2 N + \frac{N \log^2 N}{n \log n})
\]

- for \( n = N \) the complexity is \( O(\log^2 N) \);
- for \( n = 2 \) the complexity increases to \( O(N \log^2 N) \).
Bibliography