Spiking Neural P Systems – A Natural Model for Sorting Networks

Rodica Ceterchi, Alexandru Ioan Tomescu

Faculty of Mathematics and Computer Science, University of Bucharest
Academiei 14, RO-010014, Bucharest, Romania
E-mails: rceterchi@gmail.com, alexandru.tomescu@gmail.com

Summary. This paper proposes two simulations of sorting networks with spiking neural P systems. A comparison between different models is also made.

1 Introduction

Sorting is one of the most studied problem in Computer Science, as it has a wide range of applications, including sequential and parallel algorithms. Over the last decades, it has been investigated under parallel architectures, as utilizing many functional units to sort concurrently can improve performance. Batcher introduced the bitonic sorting network and the odd-even sorting network in [5], which can sort N keys in $O(\log^2 N)$ time, and with $O(N \log^2 N)$ comparators. Various improvements over these networks have been proposed in [2, 17, 18], which provide better bounds for depth or number of comparators.

Spiking Neural (SN) P systems were introduced in [10]. They simulate the behavior of neurons sending signals through axons, consisting of membranes which contain a number of occurrences of only one symbol, and release them through connections towards other membranes.

In the paper [11] an application of SN P systems for sorting N numbers has been proposed. We introduce in this paper a different approach, by first constructing SN P systems which act as comparators, and next by assembling these building blocks according to the topology of a sorting network.

Sorting has been modeled or simulated with a variety of P systems. In this paper we introduce first a model which uses a SN P system comparator (of two values), and next another model based on an n-comparator. Section 2 presents preliminaries on sorting networks. Section 3 introduces the SN P systems used as ascending/descending comparators, and shows how to connect them by classical sorting networks in order to obtain sorting SN P systems. Section 4 presents yet another model, an n-comparator generalization. This question is related to optimal data layouts for networks of processors capable of holding more than one piece of

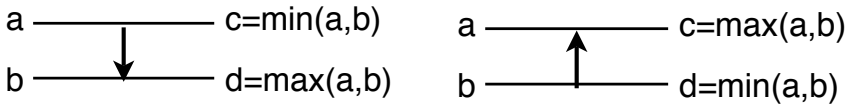
data. Finally, in Section 5 a comparison is made between the three models, the one introduced in [11], and the two other ones presented in this paper.

2 Preliminaries on Sorting Networks

A bitonic sequence is a concatenation of two monotnic sequences, one ascending, and the other one descending, or a sequence such that a cyclic shift of its elements would put them in such a form.

The key components of a bitonic network are the bitonic splitters and the bitonic mergers. The splitter of size N takes as input a bitonic sequence of length N and partitions it in two bitonic sequences of equal length, such that all the elements in the first sequence are smaller than (or greater than) all the elements in the second sequence. A bitonic merger of size N consists of a splitter of size N and of two mergers of size $N/2$, of opposite direction. It accepts as input a bitonic sequence and sorts it in ascending or descending order (direction).

As any sequence of two numbers is bitonic, the sorting network uses bitonic mergers of increasing size and alternating direction to construct bitonic sequences of increasing length. The last such merger, of size N, renders the whole sequence of N numbers sorted.



(a) Increasing comparator (b) Decreasing comparator

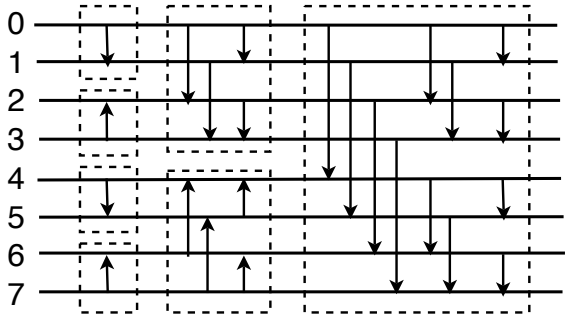
Fig. 1. Network devices

Following [14] it is customary to represent a network as an ordered set of N *lines* (wires) connected by a set of compare-exchange devices (*comparators*, for brevity). A comparator has two input terminals, a and b, and produces two output terminals c and d. If the comparator is increasing, Fig. 1(a), then $c = \min(a, b)$ and $d = \max(a, b)$, while if the comparator is decreasing, Fig. 1(b), $c = \max(a, b)$ and $d = \min(a, b)$. A bitonic sorting network for $N = 8$ is represented in Fig. 2(a).

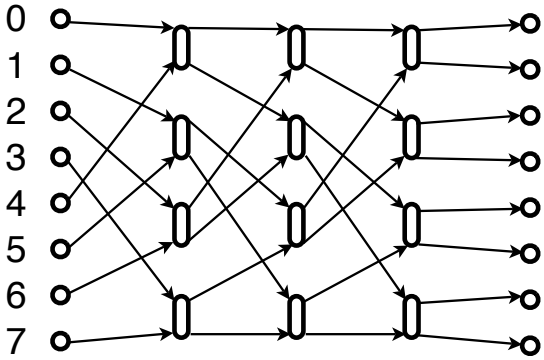
A network can also be represented as a directed acyclic graph [8].

Definition 1 (Network). *A network T of size N is a directed acyclic graph such that:*

1. *there are N nodes, called input terminals, with in-degree 0 and out-degree 1, labeled from 0 to $N - 1$;*



(a) A bitonic sorting network of size $N = 8$. The network can be partitioned in three stages, each containing bitonic mergers of size 2, 4, and 8, respectively.



(b) The bitonic merger for $N = 8$ represented as a graph.

Fig. 2. The bitonic sorting network and the bitonic merger of size 8.

2. *there are N nodes, called output terminals, with in-degree 1 and out-degree 0, labeled from 0 to $N - 1$;*
3. *all the remaining nodes u, representing comparators, have in-degree and out-degree 2.*

In Fig. 2(b) is represented the bitonic merger under the above formalism.

We define the depth of a node u of network T, $d(u)$, as the length of the longest path in T from an input node to u. The depth of network T, $d(T)$, is the maximum

depth of a node of in-degree and out-degree 2 in T. Any network can be viewed as a series of steps, each containing at most $N/2$ parallel devices. Each step t contains the nodes of T at depth t.

The arcs of a network can be partitioned in N arc-disjoint paths, each joining an input node to an output node. Such a partition yields a *line-representation* of T, as in [14].

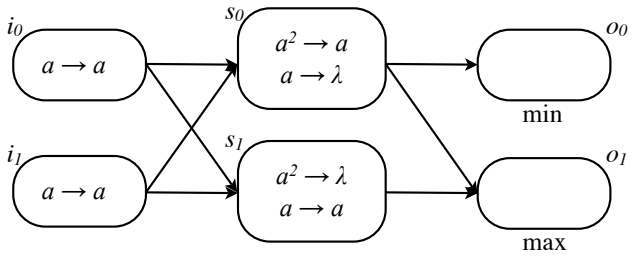
3 Spiking Neural P Systems for Sorting Networks

We note that the above representation is a theoretical model which indicates the comparisons between input values. However, in the context of SN P systems, this model has a straightforward implementation. Each wire is now represented by a synapse between two neurons, and each value x travels between two neurons as x spikes, one spike per time unit. Comparators are implemented by a set of neurons which send the minimum and the maximum (as number of spikes) through designated synapses. Once these two ingredients are at hand, we proceed to construct a SN P system in the same way the original sorting network was constructed.

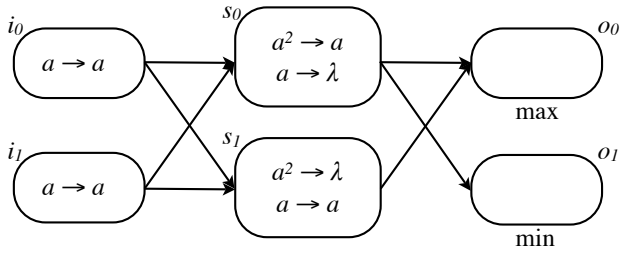
Ionescu and Sburlan [11] introduced a SN P system which sorts N numbers, and consisted of 3 layers of N neurons each. The first layer was made up of input neurons which in the initial configuration contained the input values codified as numbers of spikes. At each time unit these neurons sent one spike each to the second layer. This layer decanted the spikes to the third layer, where the output neurons were located. After a number of steps equal to the maximum value of the N numbers, the ith output neuron received the ith smallest value, codified as number of spikes, sorting thus in ascending order. In a way, the idea of the algorithm is the same as that of bead sort [4].

In this section we are concerned only with comparators of two elements, hence with SN P systems which sort two numbers (called for brevity SN P comparators). In Fig. 3(a) we give an ascending comparator, and in Fig. 3(b) we give a descending comparator. The SN P ascending comparator functions in the following way: the first layer of neurons (labelled with i) initially contains the values to be compared, codified as number of spikes. At each step they instantaneously send one spike to both s_0 and s_1. As long as both s_0 and s_1 receive spikes, only s_0 sends one spike to o_0 and o_1. After one input neuron has consumed its spikes, the minimum is obtained in o_0. There will be only one input neuron to send spikes to s_0 and s_1. In this case, s_0 forgets its spikes, and s_1 sends them to o_1, where the maximum is obtained.

Consider the SN P system modeling an ascending comparator and the numbers x and y to be sorted. In order to be able to use these SN P systems as building blocks of a bitonic sorting network, we assume that instead of loading the numbers x and y as spikes in i_0 and i_1 in the initial configuration, they are fed one by one to these input neurons by another neuron.



(a) Increasing comparator



(b) Decreasing comparator

Fig. 3. SN P systems modeling comparators

Lemma 1 (Composition lemma). *Suppose that in each time unit from t_0 until $t_0+(x-1)$ neuron i_0 receives one spike and that in rest it does not receive any spike. Analogously, suppose that in each time unit from t_0 to $t_0+(y-1)$ neuron i_1 receives one spike, and that in rest it does not receive any spike. Then neuron o_0 does not receive any spike, except for time moments from t_0+2 until $t_0+2+(\min(x,y)-1)$, when it receives one spike at each moment. Analogously, neuron o_1 does not receive any spike, except for time moments from t_0+2 until $t_0+2+(\max(x,y)-1)$, when it receives one spike at each moment.*

Proof. Consider the time moments t, with $t_0 \leq t \leq t_0 + (\min(x,y) - 1)$. Both neurons i_0 and i_1 receive spikes and in turn send them through the synapses (by the rule $a \to a$). s_0 and s_1 receive two spikes each, neuron s_0 sends one spike to o_0 and o_1 (by the rule $a^2 \to a$), while neuron s_1 forgets them (by the rule $a^2 \to \lambda$). Therefore at time moment $t + 2$ neurons o_0 and o_1 receive one spike each. From time moment $t_0 + \min(x,y)$ onward, only one neuron of i_0 and i_1 sends spikes, hence the configuration of the synapses of s_0 and s_1 prevent o_0 from receiving other spikes. The first part of the claim is proved.

At each time moment t, with $t_0 + \min(x, y) \leq t \leq t_0 + (\max(x, y) - 1)$, neuron o_1 receives one spike at moment $t + 2$. After time moment $t_0 + \max(x, y)$ there are no other spikes entering in system, hence from time moment $t_0 + 2 + \max(x, y)$ onward there will be no other spikes entering neuron o_1.

A similar lemma is valid in the case of a SN P decreasing comparator.

Assume that we are given a network T as a graph, and that we have a line-representation of it (i.e. a set of N arc-disjoint path linking input terminals with output terminals). Hence, we extend Definition 1, by labeling edges, apart from input and output terminals. For every path that begins with input terminal labeled i, we label all its edges with i. More formally, we have the following definition.

Definition 2 (Edge labeling). *Given a graph T as in Definition 1 representing a sorting network, and a line-representation of T, we attach to each edge $e \in E(T)$ that belongs to a path in the line representation of T beginning with i, label $l(e) = l(i)$ (supposing that i is labeled with $l(i)$).*

For example, in Figure 5 we have a labeled bitonic merger.

A SN P system modeling a sorting network given as a graph is obtained in the following way. For each input terminal node i we have a corresponding input neuron i_i. For each comparator (ascending / descending) we have the s- and o-neurons of a SN P comparator (ascending / descending). For each edge of the graph between two comparators we have synapses between corresponding SN P comparators. The output terminal nodes are the o-neurons of the last SN P comparators. Additionally, we add to all o-neurons, except the output ones, the rule $a \to a$.

More formally, we construct and label the SN P system in the following recursive way.

i) for each input terminal node i we have a corresponding input neuron $i_i = i_{i,1}$, $0 \leq i \leq n - 1$;
ii) for each comparator at depth $1 \leq k \leq d(T)$ with incident edges labeled with i and j, $i < j$, we add the s- and o-neurons of a SN P comparator, connected in the previously specified way. With the notations in Figure 3, let s_0 and s_1, and o_0 and o_1 be the s-, and o-neurons, respectively, just added. We add synapses between the following pairs of neurons: $(i_{i,k}, s_0)$, $(i_{i,k}, s_1)$, $(i_{j,k}, s_0)$, $(i_{j,k}, s_1)$. Additionally, if $k < d(T)$, we label o_0 with $o_{i,k} = i_{i,k+1}$, and o_1 with $o_{j,k} = i_{j,k+1}$; else we label o_0 with $o_{i,k} = o_i$, and o_1 with $o_{j,k} = o_j$.

As an example, Figure 4 depicts a SN P system which models the bitonic merger of size $N = 8$.

Theorem 1. *For any SN P comparator at depth k corresponding to a comparator with incident edges $i < j$ which carry values x and y, respectively, we have that*

1a) in each time moment from $2(k-1)$ until $2(k-1) + x$ neuron $i_{i,k}$ receives one spike;

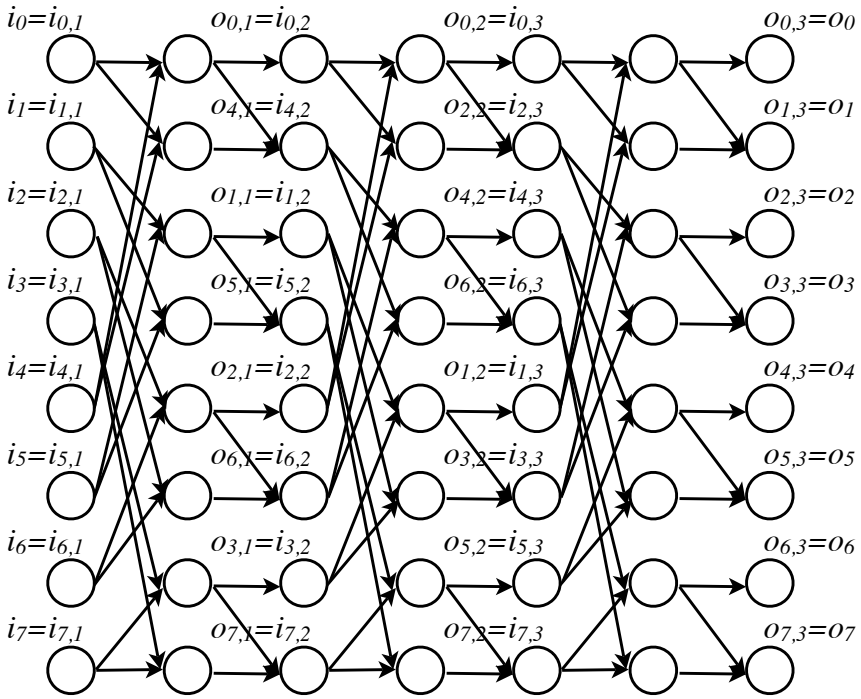


Fig. 4. A SN P system modeling the bitonic merger of size $N = 8$.

1b) in each time moment from $2(k-1)$ *until* $2(k-1)+y$ *neuron* $i_{j,k}$ *receives one spike.*

In case of an ascending comparator,

2a) in each time moment from $2k$ *until* $2k + \min(x, y)$ *neuron* $o_{i,k}$ *receives one spike;*

2b) in each time moment from $2k$ *until* $2k + \max(x, y)$ *neuron* $o_{j,k}$ *receives one spike.*

In case of a descending comparator,

3a) in each time moment from $2k$ *until* $2k + \max(x, y)$ *neuron* $o_{i,k}$ *receives one spike;*

3b) in each time moment from $2k$ *until* $2k + \min(x, y)$ *neuron* $o_{j,k}$ *receives one spike.*

Proof. We prove the claim by induction on k. When $k = 1$ we are at time moment $t = 0$. We have explained previously that the behaviour of the system when the spikes are loaded initially in the input neurons is identical to when they are fed one by one to these neurons. Claims 2 and 3 are true from Lemma 1 and $t_0 = 0$.

We now suppose that the claim is true for k, with $1 \leq k < \log N$, and prove it for $k+1$. From claims $2b$ and $3b$ of the induction hypothesis, we know that $o_{i,k} = i_{i,k+1}$ receives one spike from $2k$ until $2k+u$, where u is the value carried by wire i before the comparator at depth $k+1$. Analogously, $o_{j,k} = i_{j,k+1}$ receives one spike from $2k$ until $2k+v$, where v is the value carried by wire t before the comparator at depth $k+1$. This proves claims $1a$ and $1b$. If we take $t_0 = 2k$, $x = u$, and $y = v$ in Lemma 1, we have that claims 2 and 3 are true.

Corolary 1 *Given a network T of size N, if we replace the comparator nodes by the appropriate SN P systems sub-networks, the result is still a sorting network.*

The sorting network obtained with SN P systems works differently than the initial one. At time moment $2d(T) + \min\{x_0, \ldots, x_{N-1}\}$ all output neurons contain the value $\min\{x_0, \ldots, x_{N-1}\}$ as number of spikes a. Let us denote with $min_1 = \min(\{x_0, \ldots, x_{N-1}\} \setminus \{\min(x_0, \ldots, x_{N-1})\})$. Then at time moment $2d(T) + min_1$ all output neurons $o_1, \ldots, o_{N-1}$ contain min_1 spikes and o_0 remains with $\min\{x_0, \ldots, x_{N-1}\}$. Finally, at $2d(T) + \max\{x_0, \ldots, x_{N-1}\}$ we have in o_{N-1} the value $\max\{x_0, \ldots, x_{N-1}\}$, and all other output neurons contain the initial set in ascending order.

4 An n-Comparator Improvement

In the previous section we were concerned with constructing a SN P system which implemented a given sorting network, each comparator having a corresponding SN P systems. We now address the problem of comparators of more than two values, and show how we can transform a network given as in Definition 1 into a generalized one, with n-comparators which can sort n values. The only restriction we make is that the network has comparators on only one direction (ascending or descending). This is not a limiting assumption in our treatment of sorting with spiking neural P systems, as, for example, a bitonic sorting network is a serial and parallel connection of bitonic mergers, which have comparators of the same direction. From [11] we have at hand a SN P system which can sort n values, therefore we show how to assemble these building blocks to get a sorting SN P system which can sort N values.

The idea of the algorithm we propose stems from the following observation. Suppose we have N input terminals, and we can use comparators of at most n values. We try to design the first step of the generalized network, and have that the depth of the furthest comparator of the initial network that can be simulated by one n-comparator is $\log_2 n$. Let this device be u and let i and j be its incident edges. This implies that all prior devices involving lines i, j and all other lines that they were connected with, have to be implemented by the same n-comparator as the one which act on i and j. We call these lines *predecessors* of depth $\log n$ of u and we say that they are *mapped* to the same comparator as i and j. In addition, observe that as the n-comparator sorts the whole sequence of predecessors, then it also implemented correctly the standard comparators.

Definition 3 (Predecessors of a node). *We denote by predecessors of depth m of node u the set $P_m(u) = \{l(xy) \mid xy \in E(T)$ and there exists a path from y to u of length $m-1\}$*

Apart from u, at depth $d(u)$ reside other devices between lines mapped to the same comparator as i and j by the above procedure. These devices have to be simulated by the same comparator holding i and j. We call these devices *neighbours* of depth $\log n$ of u, and give the following definition.

Definition 4 (Neighbours of a device). *We denote by neighbours of depth m of node u of T having $d(u) \geq 2$, the set $N_m(u) = \{v \in V(T) \mid d(v) \geq 2$ and $P_m(u) \cap P_m(v) \neq \emptyset\}$*

In order to simulate $\log n$ steps locally in one n-comparator, any n-comparator should accommodate all the lines which compare values in these $\log n$ steps. This imposes a limit on the number of neighbours of depth $\log n$ of a device u. More specifically, we have the following property:

Property 1. We say that a network T of size N admits a generalized network with comparators of n values if $|N_{\log n}(u)| \leq n/2$, for any $1 \leq s \leq d(T)/\log n$, and any node $u \in V(T)$ with $d(u) = s \log n$.

Bearing all this in mind, we give an algorithm to construct $\lceil d(T)/\log n \rceil$ mapping functions $\mathcal{D}_s$ which for any wire $i \in \{0, \ldots, N-1\}$ indicates the comparator to which is mapped at step s of the generalized network T'.

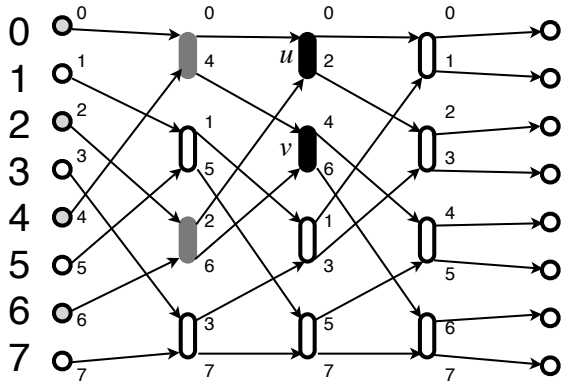


Fig. 5. The bitonic merger of size 8, given as a graph. Edges are labeled as in Definition 2, according to the classical line-representation of the bitonic merger. Neighbouring nodes u and v at depth 2 are shown in black. They have predecessors $\{0, 2, 4, 6\}$. The rest of devices linking these lines are shown in gray.

Input: A network T of size N and a line-representation of it, n the maximum capacity of one comparator. Network T has all comparators of the same direction and satisfies Property 1.

Output: A sequence of functions D_s, $0 \leq s < \lceil d(T)/\log n \rceil$, with domain $\{0, \ldots, N-1\}$ representing a mapping of wires to comparators at stage s of the generalized network T'.

label edges of T as in Definition 2;
forall $0 \leq s < \lfloor d(T)/\log n \rfloor$ **do**
    set counter $p = 0$;
    reset previous markings;
    forall *nodes* $u \in V(T)$ *not marked, at depth* $d(u) = \log n + s \log n$ **do**
        forall $v \in P_{\log n}(u)$ **do**
            $\mathcal{D}_s(v) = p$;
        forall $v \in N_{\log n}(u)$ **do**
            mark node v;
        $p = p + 1$;
`// treatment of the special case when` $d(T)$ `is not divisible by` $\log n$
remaining-depth $\leftarrow d(T)$ *mod* $\log n$;
if *remaining-depth* > 0 **then**
    $s \leftarrow \lfloor d(T)/\log n \rfloor$;
    set counter $p = 0$;
    reset previous markings;
    forall *nodes* $u \in V(T)$ *not marked, at depth* $d(T)$ **do**
        forall $v \in P_{remaining-depth}(u)$ **do**
            $\mathcal{D}_s(v) = p$;
        forall $v \in N_{remaining-depth}(u)$ **do**
            mark node v;
        $p = p + 1$;

Algorithm 1: Deriving the mapping of wires to comparators in the generalized network.

5 Conclusions and Open Problems

This paper has proposed two models (which we call Model 2 and Model 3) of simulating a sorting network with SN P systems and has proved the correctness of the construction. These systems do not have a simple form as the one in [11] (Model 1), so their usefulness remains to be investigated. We consider here a number of measures of the models: number of neurons, number of synapses, total number of rules in all neurons, maximal length of rules, and time complexity.

Model 1 has three layers, with N neurons each. On the other hand, Model 2 has $1+2+\ldots+\log N$ steps, each being implemented by $2N$ neurons. If we also add

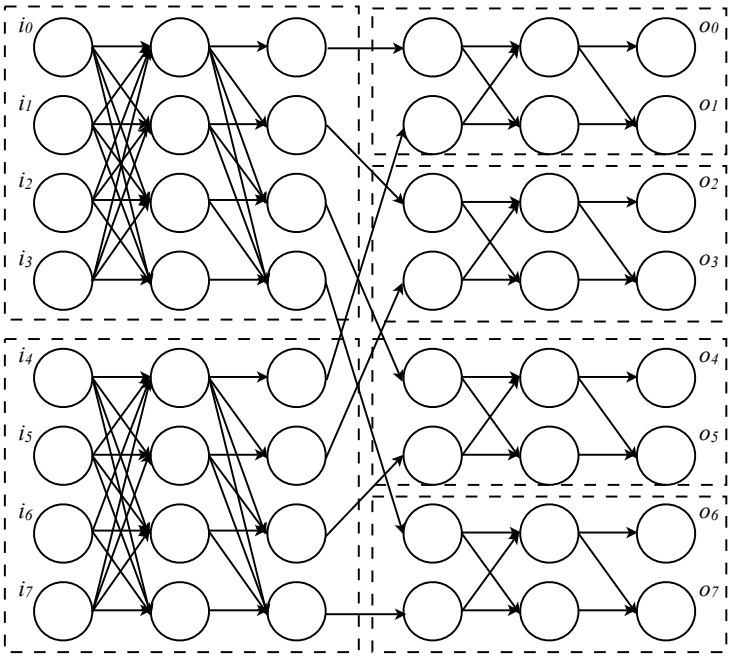


Fig. 6. A SN P system constructed from the generalized network of the bitonic merger of size 8.

the N input neurons, we get a total number of neurons of $N + N \log N(\log N + 1)$. In Model 3 the situation is similar, except that now we have $\frac{\log N(\log N+1)}{2 \log n}$ steps, which give a total number of $N + N\frac{\log N(\log N+1)}{\log n}$ neurons. Even if in the two proposed models this measure has increased by a factor of $\log^2 N$, we will see that concerning other measures, we get a benefit of at least $\frac{\log^2 N}{N}$.

The number of synapses of Model 1 is quadratic in N, as we have synapses between any pair of neurons in the first two layers. The total number of synapses is $\frac{3N^2+N}{2}$. In Model 2, for each step of the bitonic sorting network, we have $2N$ synapses between i-neurons and s-neurons, and $2\frac{N}{2} + \frac{N}{2}$ between s-neurons and o-neurons. In Model 3, at each step of the generalized network we have $\frac{3n^2+n}{2}\frac{N}{n}$ synapses. This gives a total number of synapses in Model 2 of $\frac{7}{2}N\frac{\log N(\log N+1)}{2}$, and in Model 3 of $\frac{3n+1}{2}N\frac{\log N(\log N+1)}{2 \log n}$.

Concerning the total number of rules, in Model 1 we have again a quadratic dependence $N^2 + N$. In Model 2 we have $3N$ rules in each step of the network, hence $3N\frac{\log N(\log N+1)}{2}$ rules in all. The number of rules per step in Model 3 is $\frac{N}{n}(n + n^2)$, which gives a total of $N(n + 1)\frac{\log N(\log N+1)}{2 \log n}$.

As in each time unit only one spike is discharged from the input neurons, then the complexity of the algorithm of [11] is $O(M)$, where M is the maximum of the N numbers. As, in general, we have to sort N distinct numbers, then the maximum of them is N, hence the complexity of the algorithm is $\Omega(N)$. The time complexity of the two proposed models is $O(M + d(T))$, where $d(T)$ is the depth of the network being simulated (i.e. $\log^2 N$, and $\frac{\log^2 N}{\log n}$, respectively). Usually, the maximum number M does not depend on N, so we have $O(M) = O(M + d(T))$.

We also note that now the length of the rules is constant. An overview of these measures are presented in Table 1.

An open problem that remains to be investigated is how to further reduce the number of neurons of a sorting SN P system. We propose for scrutiny the class of periodic sorting networks, which are composed of a sequence of identical blocks. Since only one block needs to have a SN P system implementation, then a periodic sorting network can be realized by recirculating the output of a block back as its input. This results in savings in neurons and synapses. Consider for example the odd-even sorting network of Batcher [5] which is composed of N identical applications of a period of depth 2. This can provide a linear number of neurons in N, with the same time complexity $O(M)$.

However, the main difficulty behind such an approach is the ability to tell when the numbers are sorted. We note that the output neuron holding the minimum has to stop recirculating spikes before the output neuron holding the maximum. The idea of a global clock holding a number of spikes proportional to the number of times the identical blocks have to be applied is not enough.

Table 1. Comparison between the model proposed in [11] (Model 1), the direct simulation of the bitonic sorting network with a SN P system (Model 2), and the simulation of a generalized bitonic sorting network with n-comparators (Model 3). The models sort N numbers, M being the maximum.

Measure	Model 1 [11]	Model 2	Model 3
Number of neurons	$3N$	$N + N \log N(\log N + 1)$	$N + N\frac{\log N(\log N+1)}{\log n}$
Number of synapses	$\frac{3N^2+N}{2}$	$\frac{7}{2}N\frac{\log N(\log N+1)}{2}$	$\frac{3n+1}{2}N\frac{\log N(\log N+1)}{2\log n}$
Number of rules	$N^2 + N$	$3N\frac{\log N(\log N+1)}{2}$	$N(n+1)\frac{\log N(\log N+1)}{2\log n}$
Maximal length of rules	$N + 1$	3	$n + 1$
Time complexity	$O(M)$	$O(M + \log^2 N) = O(M)$	$O(M + \frac{\log^2 N}{\log n}) = O(M)$

References

1. A. Aggarwal, A.K. Chandra, M. Snir, "Communication Complexity of PRAMs", *Theoretical Computer Science,* vol. 71, no.1, pp. 3 - 28, Mar. 1990.

2. M. Ajtai, J. Komlos, and E. Szemeredi, "An $O(N \log N)$ Sorting Network", *Proc. 15th Ann. ACM Symp. Theory of Computing,* pp. 1-9, May 1983.
3. A. Alexandrov, M. Ionescu, K.E. Schauser, C. Scheiman, "LogGP: Incorporating Long Messages into the LogP model", *Journal of parallel and distributed computing,* vol. 44, no. 1, pp. 71-79, 1997.
4. J.J. Arulanandham, "Implementing Bead – Sort with P Systems", *Unconventional Models of Computation 2002 (C.S. Calude, M.J. Dinneen, F. Peper Eds.)*, LNCS vol. 2509, pp. 115-125, 2002.
5. K.E. Batcher, "Sorting networks and their applications", *Proc. AFIPS Spring Joint Comput. Conf.,* vol. 32, pp. 307-314, Apr. 1968.
6. G. Bilardi, "Merging and Sorting Networks with the Topology of the Omega Network", *IEEE Transactions on Computers,* vol. 38, no. 10, pp. 1396-1403, Oct. 1989.
7. D.E. Culler, R.M. Karp, D.A. Patterson, A. Sahay, K.E. Schauser, E. Santos, R. Subramonian, and T. von Eicken, "LogP: Towards a Realistic Model of Parallel Computation", *Proc. Fourth ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming,* pp.1-12, May 1993.
8. M. Dowd, Y. Perl, M. Saks, L. Rudolph, "The balanced sorting network", *Proc. Second annual ACM symp. on Principles of distributed computing,* pp. 161-172, 1983.
9. M. Dowd, Y. Perl, M. Saks, L. Rudolph, "The periodic balanced sorting network", *JACM,* vol. 36. no. 4, pp. 738-757, 1989.
10. M. Ionescu, Gh. Păun, T. Yokomori: Spiking neural P systems. *Fundamenta Informaticae*, 71, 2-3 (2006), 279–308.
11. M. Ionescu, D. Sburlan, "Some Applications of spiking neural P systems", *Preproceedings of Membrane Computing, International Workshop - WMC8*, Thessaloniki, Greece, pp. 383-394, 2007.
12. M.F. Ionescu, "Optimizing Parallel Bitonic Sort", *Tech. Report TRCS96-14,* Dept. of Comp. Sci., Univ. of California, Santa Barbara, July 1996.
13. M.F. Ionescu, K.E. Schauser, "Optimizing parallel bitonic sort", *Proc. 11th Int'l Parallel Processing Symp.,* pp. 303-309, 1997.
14. D.E. Knuth, *The art of computer programming*, volume 3: sorting and searching, second ed. Redwood City, CA: Addison Wesley Longman, 1998.
15. C. Kruskal, L. Rudolph, M. Snir. "A complexity theory of efcient parallel algorithms", *Theoretical Computer Science,* vol.71, no.1, pp. 95 - 132, Mar. 1990.
16. J.D. Lee, K.E. Batcher, "Minimizing Communication in the Bitonic Sort", *IEEE Trans. on Parallel and Distributed Systems,* vol. 11, no. 5, pp. 459-474, May 2000.
17. F. Leighton, "Tight Bounds on the Complexity of Parallel Sorting," *IEEE Trans. Computers,* vol. 34, no. 4, pp. 344-354, Apr. 1985.
18. M.S. Paterson, "Improved Sorting Networks with $O(\log N)$ Depth," *Algorithmica,* vol. 5, pp. 75-92, 1990.
19. L. Rudolph, "A robust sorting network", *IEEE Trans. Comput.* C-32,4, pp. 326-335, 1985.