Proceedings of the First International Workshop on Sustainable Ultrascale Computing Systems (NESUS 2014) Porto, Portugal

Jesus Carretero, Javier Garcia Blas
Jorge Barbosa, Ricardo Morla
(Editors)

August 27-28, 2014

Data parallel algorithm in finding 2-D site percolation backbones

Biljana Stamatovic*, Roman Trobec+

*University of Donja Gorica, Montenegro biljana.stamatovic@udg.edu.me

+Institut Jozef Stefan, Slovenia roman.trobec@ijs.si

Abstract

A data parallel solution approach formulated with cellular automata is proposed with a potential to become a part of future sustainable computers. It offers extreme parallelism on data-flow principles. If a problem can be formulated with a local and iterative methodology, so that data cell results always depend on neighbouring data items only, the cellular automata could be an efficient solution framework. We have demonstrated experimentally, on a graph-theoretical problem, that the performance of the proposed methodology has a potential to be for two orders of magnitude faster from known solutions.

Keywords Sustainable computing, Data parallel, Data-flow, Cellular automata, Percolation backbone, Cluster

I. Introduction

Increasing data and processing requirements of applications are constantly pushing for further increases in computational capabilities. Today, we have reached a point where computational systems are confronted with physical barriers that limits a significant further increase of system frequencies. Additionally, excessive energy consumption, limited scalability and complex data management are obstacles that have to be solved nowadays to enable further increase of the computing performances.

It seems that massively parallel computing devices on all levels of computing, connected in heterogeneous computing systems, could counterbalance these difficulties. Multicore systems, graphic processing units (GPU), dedicated FPGA accelerators are already widely implemented and investigated options in contemporary high performance computers. Adaptive combinations of control-flow and data-flow approaches seems to be able to offer general purpose and sustainable ultrascale computers that will be able to cope the applications either with demanding processing power or with complex analysis of big data sets.

The cellular automata (CA) can be considered as an alternative way of computation based on local data-flow principles. The concept of CA was first proposed by John von Neumann in 1950s through self-reproducing systems (published later in [1]). The formalization has been improved by various authors [2, 3, 4, 5], emphasizing different theoretical and practical perspectives. Various application areas of CAs, ranging from ecology [6], biology [7], diffusion through soil pores [8], image processing [9], wafer diagnostics [10], sociology [11] etc., have been investigated later.

A CA can be informally represented as a set of regularly and locally connected identical elements. The elements can be only in a finite set of states. The CA evolves in discrete time steps, changing the states of elements according to a local rule, which is the same for all elements. The new state of each element depends on its previous state and on the state of its neighbourhood. The characteristic properties of CAs are therefore locality, discreetness and synchrony.

The CA can be considered as a computing machine in the sense that it is able to transform an input configuration, embedded in its initial state, to an output configuration. The theoretical studies of the computational capabilities of CA [3, 4] have shown that there exists CAs that are equivalent to the Turing machine and therefore can be used as general purpose computers. Our work is relevant to the practical backward approach in the design of CAs [5], where transition rules are searched for that result in CA states that match the physical system.

Many efficient applications have been developed with data-flow approaches and CAs on problems that have been previously solved with local numerical methods [12, 13]. The CAs can compete with classical computers in computational performance and efficient use of energy because of massive parallelism and relatively low system clock. The hardware resources of CAs can be implemented with Systems-on-Chips (SoC) on the wafer level [14], with more general Field programmable logic arrays(FPGA) [12], with emulation on manycore systems or on hybrid architectures mixing all above options.

A CA can be mapped on the physical system with the methodology already established in heterogeneous computing: the initial problem data and possibly transition rules are mapped from the host CPU onto the computing array which implements the CA. CA cells compute in parallel for the required number of computational steps. After a stopping criteria is met the results are read from the CA into the host memory and further analysed or visualized on the CPU. The procedure can be implemented in a loop with an eventual reprogramming of the CA. It is evident that such a data-flow approach could have a striking advantage [15] over the classical crunching machines in the significantly lower consumption of energy per a computing operation, which could contribute to the sustainability of future computers.

The CAs rely on the discreteness, locality, regularity, and synchrony. Their simple definition has several advantages, e.g. CAs are not limited by the number of elements, their evolution is inherently parallel, they have a strong resemblance to many important principles in the nature like: cells that are building blocks for large systems, elementary particles, etc. However, to approach to real problems, some additional properties of CAs are necessary, e.g. global communication (for loading of initial data, implementation of stopping criteria or reading of final results) and non-homogeneity (for an adequate treatment of boundary conditions).

The rest of the work is organised as follows. In the next Section a principal description of the proposed system architecture is described focusing on global operations related to data movements. Section III is an overview of the work related to the identification of infinite clusters and their percolation backbones. This problem can be considered as an example of a large class of demanding tasks that can be solved by parallel iterative operations on local data. In Section IV a new efficient CA algorithm is proposed that can efficiently cope with this graph-theoretical problem. The proposed approach is experimentally tested and evaluated on finding 2-D site percolation backbones. It was shown that the performance of the proposed solution has a potential to be for two orders of magnitude faster than previous known solutions.

II. System architecture

To objectively evaluate a CA algorithm the time needed for data manipulation has also be considered. Fore example, initial data, e.g. pixels of images, sites of percolation array, etc. must be loaded into CA cells. Beside data management capability, each CA cell must comprise also a processing and memory unit that are tailored to algorithm requirements. All cells usually execute the same operations with the same system clock.

We can suppose that a CA array is build from rectangular tiles that incorporate $N = (L \times M)$ CA cells. For simplicity, we suppose that $M = L$. Each cell is connected directly with a small number of nearest neighbours d, e.g. $d = 4$ or $d = 8$ with a simple communication switch built in each CA cell [16].

A set of l routers with radix r is connected to a subset of r CA cells. The radix is here a number of communication ports to the lower level. Each router has also a single, possibly faster, port to the higher level. If the number of CA cells is large, p intermittent levels of routers can be introduced. The routers communicate with a host computer that manage data transfer to lower layer routers, or to CA cells if a router is positioned on the lowest level.

The communication among direct d connected cells is performed in a single communication step (hop). Also the routers can connect two levels of routers in a single hop. In the case of a single level of l routers, the data can be transferred from the host computer to all N CA cells in three hops. For example, if we have a grid of $N = 10.000$ CA cells arranged in a (100×100) grid with $d = 8$ and $r = 36$ then $dr^2 = 10.368$ CA cells can be reached with a single level of r routers, so the whole CA grid can be filled with initial data in three hops.

Note that a certain degree of pipe-lining in the communication is possible that can further decrease the total communication time. The ratio between the communication and calculation time limits the scalability and speed-up of the execution, as usual in all parallel systems.

III. Related work and problem definition

In percolation theory, one of the fundamental task is to find a spanning cluster (termed also infinite cluster) of the same sites that connect two opposite borders of a simulated square domain. Such clusters appear if the probability that a site or CA cell is coloured black is higher than the percolation probability p_c [17]. The recursive Hoshen-Kopelman algorithm [18] is a popular algorithm for the identification of infinite clusters that has been also successfully parallelized [19] on distributed memory computers. Several other approaches to the paralellization of graph algorithms are presented in [20].

The next task, which is computationally more complex and therefore still a bottleneck, is the backbone identification in infinite clusters. Informally, all sites in dead ends or loops that can not contribute to the "transport" of a matter trough the infinite cluster has to be identified and removed to determine the backbone. Tarjan's recursive depth-first-search (DFS) algorithm [21] is well known and often used for the backbone identification. The key to finding the backbone is to recognize the articulation sites of the infinite cluster. A local procedure for recognizing articulation sites along with an improved, almost four times faster, algorithm for the backbone identification was proposed in [22].

An interesting approach for the backbone identification in an infinite cluster, based on the principle of direct electrifying solved by FEM methodology, is presented in [23]. All of the backbone finding algorithms are recursive with a risk of stack overflow in large systems.

As an alternative to the above listed approaches, we propose a CA algorithm for identification of infinite clusters and their backbones in 2-D grids with open boundary conditions. The algorithm relays on local rules and can resolve the problem of stack overflow in large systems. The inherent data-parallel approach of CA can improve efficiency and speed-up of the execution.

We consider a CA as a two dimensional lattice network of unite squares (cells) whose centres are in integer lattice (grid). For simplicity, we suppose that the grid has $N = L^2$ cells (i, j) with positions determined by indices $i, j = 0, ..., L-1$ in x and y directions, where $L \geq 3$. Each cell can exist in a finite number of states. Cells of the lattice network change their states in discrete moments in time. Cell's next state is defined by *local transition function*, which manages with altering the states of each cell, based on the present cell state and states of neighbourhood's cells. We use Moore neighbourhood with twenty-four neighbours. A cell has four nearest neighbours (*nn*) and four next-nearest neighbours (*nnn*) and sixteen not next - nearest neighbours. Two cells (i_1, j_1) and (i_2, j_2) are *nn*-neighbours if $|i_1 - i_2| + |j_1 - j_2| \leq 1$, *nnn*-neighbours if $(i_1 - i_2)^2 + (j_1 - j_2)^2 = 2$ and *nnnn*-neighbours if $2 < (i_1 - i_2)^2 + (j_1 - j_2)^2 \leq 4$.

For a CA A applying the local transition function φ_A to all cells of a configuration $Conf$ simultaneously, we get the sequence of configurations $conf_A(t, Conf)$, where $t = 0, 1, 2, ...$ is a *time step*.

The initial configuration is represented by a 2-D grid of square cells and each cell can exist in two different states, white or black. On the top and bottom boundaries of the grid are black cells only while the left and right boundaries are white. All remaining cells of the grid, are coloured black with probability p and white with probability $1 - p$. These probabilities are independent for each cell. Two examples of grids with $p = 0.5 < p_c$ and $p = 0.6 > p_c$ are shown in Figures 2 and 1, respectively. We can see that an infinite cluster appears in the example from Figure 1.

Two cells (i_1, j_1) and (i_n, j_n), which are in the same state are *nn*-connected if there exists a sequence of cells (i_k, j_k), $2 \leq k \leq n$ which are in the same state, such that each pair (i_{k-1}, j_{k-1}) and (i_k, j_k) are *nn*-neighbours. Similarly, they are *nnn*-connected if the sequence of

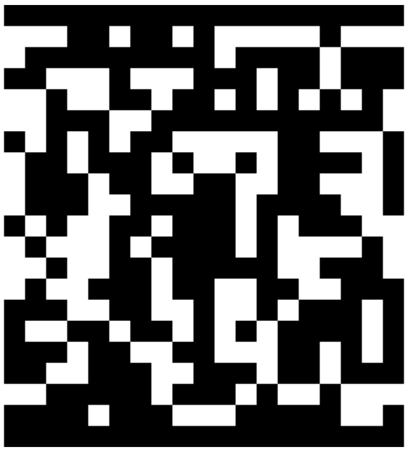
Figure 1: Initial grid configuration generated with $p = 0.5$ *without infinite clusters.*

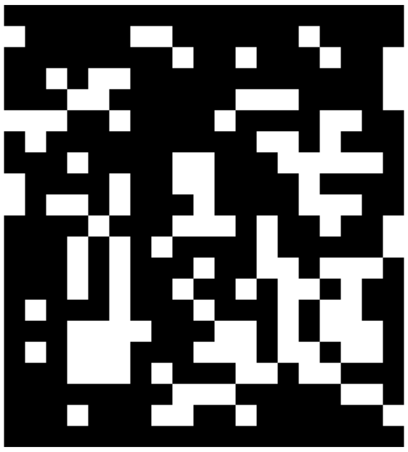
Figure 2: Initial grid configuration generated with $p = 0.6$ *with a single infinite cluster.*

the cells (i_k, j_k), $2 \leq k \leq n$ which are in the same state, such that each pair (i_{k-1}, j_{k-1}) and (i_k, j_k) are *nn*- or *nnn*-neighbours.

A black *nn*-cluster is a group of black cells which are *nn*-connected. The *infinite cluster* is a black cluster which spans from the top to the bottom row of a grid. The *backbone* is a subset of the infinite cluster, which cells are *nn*-connected with the top or bottom row cells with at least two disjoint chains. A cell is an *articulation cell* of an infinite cluster if removing the cell (i.e. changing its state to white) splits the infinite cluster into two or more parts, at least one part being connected to neither the top nor the bottom row.

In the same way, an *nnn*-cluster is a group of cells in the same state that are *nnn*-connected. In particular, white *nnn*-cluster is a group of white cells that are *nnn*-connected.

The proposed CA algorithm for identification of infinite cluster and its backbone is implemented in four steps. In the Step 1, white *nnn*-clusters are labelled with different colors. After finishing this task the algorithm identified the cells that belong to the infinite cluster. In Step 2, the algorithm recognizes articulation cells of the infinite cluster. Some of them are permanently removed, but some of them are marked by white color. In Step 3, the parts of backbone are labelled. Finally, in Step 4, some of previously marked white cells become a part of the backbone and the backbone is identified.

IV. Solution algorithm

Let $i, j \in \{0, 1, 2, ..., L-1\}$ and $t \in N, t \geq 0$. Denote by $c_{(i,j)}(t)$ the state of a cell (i, j) in time step t and by $c\,(t)$ an argument of a local transition function, which is an ordered collections of *nn*- , *nnn*- and *nnnn*- neighbours cell's states in time step t.

We use the terminology of colors. A cell is in state $'m'$ if it is coloured by color m. 0-color is white, 1-color is black, m-color is a color with code m, $m > 1$, e.g. in RGB implementation.

Let C_0 be an initial configuration. We will define five CAs A_i by their local transition functions φ_{A_i}, $i = 1, 2, ..., 5$. For a CA A_i applying the local transition function φ_{A_i} to all cells on a configuration $Conf$ simultaneously, we get the sequence of configurations $conf_{A_i}(t, Conf)$, $i \in \{1, 2, ..., 5\}$, where $t = 0, 1, 2, ...$ is a time step.

Step 1: CA A_1 will change states of some white cells in each white *nnn*-cluster. These new states will be all different because their colors are related to their positions. Let a and b are the lowest left and lowest right cells on the left and right boundaries of the grid in initial configuration C_0. Let $LR = \{a, b\} \bigcup$
$\left\{(i,j) \in C_0 | c_{(i,j)}(0) = 0 \wedge c_{(i+1,j-1)}(0) = c_{(i+1,j)}(0) = c_{(i,j-1)}(0) = 1\right\}$.
Then $\varphi_{A_1}(c(t)) = c^1_{(i,j)}(t+1)$ where:

$$c^1_{(i,j)}\,(t+1) = \begin{cases} i * L + j + 2, \ t = 0 \wedge (i,j) \in LR \\ c_{(i,j)}\,(t)\,, \text{ otherwise.} \end{cases}$$

and $t = 0, 1, 2, ...$ is a time step.

Colour $i * L + j + 2$ of a cell (i, j) depends on the position of a cell in the grid. Hence, the CA A_1 changes the state of a cell in a new unique state, different from all other cells. Also, lowest left cells of any white *nn*- and *nnn*-cluster has black *nnn*-neighbours $(i+1, j-1), (i+1, j), (i, j-1)$. Hence, in every white *nnn*-cluster at least one cell will have its unique color, different from white. Note, for $t > 1$ the CA A_1 will remain idle.

CA A_2 will colour every white *nnn*-cluster with unique color, different from black and white. Let $Cols_{(i,j)}(t) = \left\{c_{(k,l)}(t) | k \in \{i-1, i, i+1\}, l \in \{j-1, j, j+1\}\right\}$ be the set of all colors of *nn*- and *nnn*-neighbours of a cell (i, j). Then $\varphi_{A_2}(c(t)) = c^2_{(i,j)}(t+1)$ is defined by:

$$c^2_{(i,j)}\,(t+1) = \begin{cases} max(Cols_{(i,j)}(t)), \ c_{(i,j)}(t) \neq 1 \wedge max(Cols_{(i,j)}(t)) > 1 \\ c_{(i,j)}\,(t)\,, \text{ otherwise.} \end{cases}$$

where $t = 0, 1, 2, ...$ For the CA A_2, after some time step, no cell will be changed. In implementation of the CA A_2 a global variable is used to identify this.

If cells a and b have the same colors then left and right boundaries are in same nnn- cluster. Hence, if cells a and b have the same colors initial configuration doesn't have an infinite cluster.

Step 2: CAs A_i, $i = 3, 4, 5$ will implemented the part of the algorithm from [22].

CA A_3 will locally identify articulation cells and color them with white color, in the first time step. In the second time step, the CA will label some nn-neighbours of white cells, because some of the white cells may belong to the backbone. Note, that after *Step 1* no cell remains white. Definition of CA A_3 is obtained from discussion and the corollary in [22].:

Corollary 1 *Let a be a cell of the black infinite cluster, and let G_a be a set of a and its nn and nnn cells then a is an articulation cell if and only if there are in G_a at least two white cells, referred to as b and c, that belong to the same nnn-cluster but cannot be connected by the white cells from G_a.*

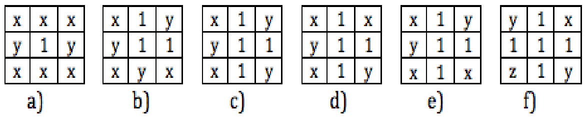

Figure 3: Cases of articulation cells: y is any color different from black and x and z are any colors with a restriction that z must be different from y.

In figure 3 some cases of G_a are shown. The set of the shown cases together with their rotated configurations for 90, 180, 270 degrees, are denoted by Ar. Let Tag is the set of black nn-neighbours of white cells which are tagged as contact couple. For example, for central cell (i, j) in figure 3 d) the contact couple is $\{(i+1, j), (i, j+1)\}$, in figure 3 e) the contact couple is $\{(i+1, j), (i, j-1)\}$ and in figure 3 f) contact couples are $\{(i, j+1), (i+1, j)\}$ and $\{(i-1, j), (i, j-1)\}$. In implementation we will have twelve cases for the tagging and we will use $nnnn$-neighbours. Here, we only use one color $tag = L^2 + 2$, because of a simplified explanation. Now, we define the local transition function $\varphi_{A_3}(c(t)) = c^3_{(i,j)}(t+1)$ by:

$$c^3_{(i,j)}(t+1) = \begin{cases} y,\ t = 0 \wedge (i,j) \in Ar\, a), b), c \\ 0, t = 0 \wedge (i,j) \in Ar\, d), e), f) \\ tag, t = 1 \wedge (i,j) \in Tag \\ c_{(i,j)}(t),\ \text{otherwise.} \end{cases}$$

where $t = 0, 1, 2, \ldots$ is a time step and y is from figure 3 a), b), c) . Note, for $t > 2$ the CA A_3 will remain idle.

Step 3: CA A_4 will color black cells from the infinite cluster that are in its backbone, except some articulation cells.

Let $m = L^2 + 3$. The local transition function φ_{A_4} will label a part of backbone by a color m. Let $TB = \left\{(i,j) | c_{(i,j)}(t) = 1 \wedge (j = 0 \vee j = L-1)\right\}$ be the set of top and bottom boundary black cells. Let Nn be a set of black and tag cells whose at least one of its nn-neighbours is in state m. Let $Ntag$ be a set of tag cells which have one white nn-neighbour and one different tagged nnn-neighbour or two white nn-neighbours and two m nnn-neighbours. Now, the local transition function $\varphi_{A_4}(c(t)) = c^4_{(i,j)}(t+1)$ is defined by:

$$c^4_{(i,j)}(t+1) = \begin{cases} m,\ (i,j) \in TB \bigcup Nn \bigcup Ntag \\ c_{(i,j)}(t),\ \text{otherwise.} \end{cases}$$

where $t = 0, 1, 2, \ldots$ is a time step. For the CA A_4, after some time step, no cell will be changed. In implementation of the CA A_4 a global variable is used to identify this.

Step 4: Here, we will define CA A_5 who decide which white cell from *Step3* is in the backbone. We will use the fact "a white cell should be restored to be a cell of the backbone if either (i) it has at least two nn-neighbours belonging to the infinite cluster, or (ii) it has an nn-neighbour of the infinite cluster and a white nn cell which are nn-connected by another cell of the infinite cluster" from paper [22].

Let B be a set of white cells with two or more m nn-neighbours and cells with an m nn-neighbour and a white nn-neighbour which is nn-connected by another m cell. The local transition functions $\varphi_{A_5} = c^5_{(i,j)}(t+1)$ is defined by:

$$c^5_{(i,j)}(t+1) = \begin{cases} m,\ (i,j) \in B \\ c_{(i,j)}(t),\ \text{otherwise} \end{cases}$$

where $t = 0, 1, 2, \ldots$ is a time step. Note, for implementation of the CA A_5 we will use $nnnn$ - neighbours.

Using the described CAs in a loop we can form algorithm 1 for the identification of infinite clusters and their backbones.

```
Data: Initial configuration $C_0$
Result: If the infinite cluster does not exist then the algorithm
        stops else the obtained set of $m$ cells is backbone.
$\varphi_{A_1}$;
while exist cell which change its state do
    $\varphi_{A_2}$;
end
if state of lowest left cell == state of lowest right cell then
    stop;            //no infinite cluster
else
    $\varphi_{A_3}$;    // articulation cells become white
    $\varphi_{A_3}$;    // $nn$-neighbours some of white cells are tagged
    Let $m$ be an unique color;
    while exist cell which change its state do
        $\varphi_{A_4}$;
    end
    $\varphi_{A_5}$;    // some articulation cells are in backbone
end
```

Algorithm 1: Algorithm for identification of an infinite cluster and its backbone.

V. Experimental results and discussion

The implementation of the algorithm is made in NetLogo 5.0.4. The algorithm was extensively evaluated on various test cases for different size of grids and percolation probabilities, i.e. densities of black cells in initial configuration. In Figures 4 and 5 the final configurations are shown after running the Algorithm 1 on initial configurations from Figures 1 and 2, respectively.

It is evident from figure 4 that there is no infinite cluster in the initial configuration from figure 1. However, figure 5 indicates a backbone of an infinite cluster, which is obviously present on the initial configuration from figure 2.

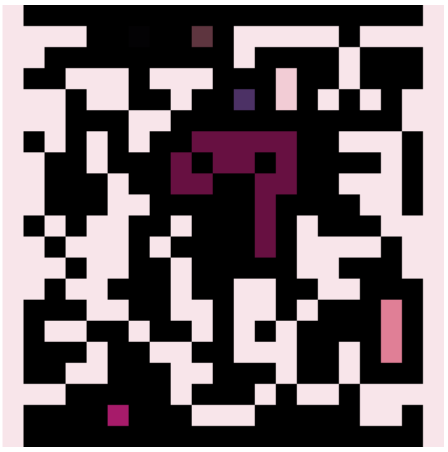

Figure 4: Final result after the application of proposed algorithm on the initial configuration from figure 1.

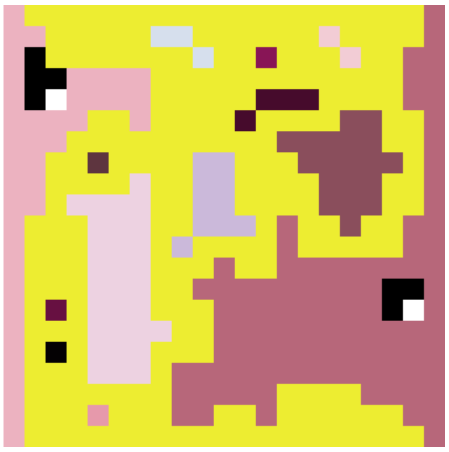

Figure 5: Yellow backbone for initial configuration from figure 2.

The preliminary numerical results for different sizes of grids and probability of black cells $p = 0.592745$ are shown in Table 1. We calculated the average time steps required for 100 initial configurations of each for grid size. The slope in the number of time steps can be estimated as $\Delta L^2/\Delta M$ and is significantly smaller as in [22]. However, the current implementation of the algorithm with NetLogo is limited with grid sizes. The results still indicate that the proposed algorithm can be faster than known algorithms for the backbone identification. For definite confirmation we need a data-flow implementation of the algorithm, which is a part of our future investigation.

Advantages of the proposed approach are in using data-flow approaches in *Step 1* and *Step 3* [24] while the well known algorithms for labelling components are based on DFS approach, which is recursive and hard to parallelize [25]. The proposed CA definition of the proposed data parallel algorithm has several advantages, e.g. it is not limited by the number of cells, its evolution is inherently parallel, and it has a strong resemblance to the important approaches in the nature like principles of cells or elementary particles.

Drawbacks of the algorithm are in using global variables for stopping CA's work. Lack of global communication, implies problems related to global synchronization, data manipulation and inability for calculation of complex mathematical operations, however, these difficulties can be resolved by dedicated hardware resources. The heterogeneous computing, supported today with data-flow approaches, FPGAs, SoCs, GPUs and manycore systems, are promising platforms for the implementation of the efficient CA based algorithms.

Dimesions of grid (L)	Mean time steps (M)
21x21	26.3
41x41	49.61
61x61	58.81
81x81	73.12
101x101	81.59
121x121	85.49
141x141	93.74
161x161	91.38
181x181	95.6
201x201	97.86

Table 1: Average number of time steps M as a function of grid size L.

References

[1] J. Neumann, Theory of Self-reproducing Automata, University of Illinois PressNeumann, 1966.

[2] J. Thatcher, Universality in the von neumann cellular model, Tech. rep., University of Michigan,. 03105-30-T (1964).

[3] E. Codd, Cellular Automata, Academic Press, NewYork, 1968.

[4] E. Burks, Essays on Cellular Automata, University of Illinois Press, 1966.

[5] S. Wolfram, Theory and Applications of Cellular Automata, World Scientific Publication, Singapore, 1986.

[6] P. Hogeweg, Cellular automata as a paradigm for ecological modeling, Appl. Math. Comput. 27 (1988) 81Ű100.

[7] G. Ermentrout, L. Edelstein-Keshet, Cellular automata approaches to biological modeling, J. Theor. Biol. 160 (1993) 97–133.

[8] G. Horgan, B. Ball, Simulating diffusion in a boolean model of soil pores, European Journal of Soil Science 45 (1994) 483–491.

[9] T. Ikenaga, T. Ogura, A DTCNN universal machine based on highly parallel 2-D cellular automata CAM2, IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications 45 (1998) 538–546.

[10] R. Trobec, I. Jerebic, Local diagnosis in massively parallel systems, Parallel Computing 23 (1997) 721–731.

[11] J. Epstein, Generative Social Science, Princeton University Press, 2006.

[12] E. Motuk, R. Woods, S. Bilbao, Implementation of finite difference schemes for the wave equation on FPGA, in: ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings, Vol. III, 2005, pp. III237–III240.

[13] G. Kosec, P. Zinterhof, Local strong form meshless method on multiple graphics processing units, Computer Modeling in Engineering and Sciences 91 (2013) 377–396.

[14] R. Trobec, Evaluation of d-mesh interconnect for SoC, in: Proceedings of the International Conference on Parallel Processing Workshops, 2009, pp. 507–512.

[15] M. J. Flynn, O. Mencer, V. Milutinovic, G. Rakocevic, P. Stenstrom, R. Trobec, M. Valero, Moving from petaflops to petadata, Commun. ACM 56 (5) (2013) 39–42.

[16] R. Trobec, Two-dimensional regular d-meshes, Parallel Computing 26 (13-14) (2000) 1945–1953.

[17] D. Stauffer, A. Aharony, Introduction to Percolation Theory, Taylor and Francis, London, 1992.

[18] J. Hoshen, R. Kopelman, Percolation and cluster distribution: I. cluster multiple labeling technique and critical concentration algorithm, Phys. Rev. B 14 (1976) 34–38.

[19] J. Teuler, J. Gimel, Direct parallel implementation of the hoshen-kopelman algorithm for distributed memory architectures, Computer Physics Communications 130 (2000) 118–129.

[20] S. Hong, T. Oguntebi, K. Olukotun, Efficient parallel graph exploration on multi-core cpu and gpu, in: Parallel Architectures and Compilation Techniques (PACT), 2011 International Conference on, 2011, pp. 78–88.

[21] T. Robert, Depth-first search and linear graph algorithms, SIAM J. Comput. 1 (1972) 146–160.

[22] W.-G. Yin, R. Tao, Algorithm for finding two-dimensional site percolation backbones, Physica B 279 (2000) 84–86.

[23] C. Li, T.-W. Chou, A direct electrifying algorithm for backbone identification, J. Phys. A: Math. Theor. 40 (2007) 14679–14686.

[24] L. Biehl, Forest fires, oil spills, and fractal geometry, Mathematics teacher 92 (1999) 128.

[25] J. H. Reif, Depth-first search is inherently sequential, Information Processing Letters 20 (1985) 229–234.