Bull. Nov. Comp. Center, Comp. Science, 36 (2014), 21–31

Cellular automata diffusion models for multicomputer implementation*

Olga Bandman

Abstract. Simulating large-scale phenomena by Cellular Automata (CA) meets the problem of designing CA models that could be efficiently implemented on supercomputers with distributed memory. Since most of large-scale spatially distributed processes contain diffusion as a component which takes a significant part of computational time, the study of coarse-grain parallelization characteristics of CA diffusion are of interest. There is a scope of investigations of CA diffusion models and their modifications. Most of them concern a two-dimensional case, although in large-scale simulation tasks three-dimensional processes are under investigation. Moreover, the known information on CA diffusion is incomplete and scattered among different scientific journals. In this paper, all known data are systematized and supplemented with parameters of three-dimensional CA diffusion. Three parameters of the model are considered to be most important for developing large-scale simulation algorithms: diffusion number of the model, the number of interprocessor exchanges in parallel implementation, and the computational time. Based on the given data analysis some recommendations are made how to make a proper choice of the CA diffusion model.

1. Introduction

Capabilities of CA to simulate nonlinear, non-continuous spatial dynamics, their algorithmic fine granularity, capability of probabilistic and stochastic modes of computation, as well as a good adaptability to supercomputer architecture, make them particularly useful for investigating the behavior of large scale complex systems. The number of developed and investigated CA-models rapidly increases, comprising the study of more and more complicated phenomena in physics, biology, and chemistry [1]. Nowadays there exists a relatively large bank of well-studied CA-models, as well as a developed theory and methodology to construct CA compositions for simulating complex phenomena [2]. Most of them contain CA diffusion as a component, which may take a significant part of computational time in the course of simulation. CA simulation of these phenomena require the usage of high performance computer systems, hence, a new problem arises concerning the parallel implementation efficiency. Its solution cannot anymore rely on the inherent fine-grain algorithmic structure of CA, since the allocation of the

*Supported by Presidium of Russian Academy of Sciences, Project 15.9-2014.

interacting CA onto a parallel processor system requires efficient coarse-grain parallelization. So, both levels of parallelism are to be exploited in developing large-scale CA-models.

Nowadays, the most powerful high performance computing systems are clusters of processing units with distributed memory and the most efficient method of parallelization is domain decomposition. There are some investigations of CA-model implementation on clusters [3–5], which show a good parallelization efficiency for synchronous [4] and formulate problems to be solved for asynchronous CA-models [5]. The latter are intensively used in simulating chemical, biological and social processes. Direct methods of coarse-grain parallelization of asynchronous CA have rather a low efficiency [6]. Hence, equivalent transformations of asynchronous into a synchronized (block-synchronous) mode of CA functioning are proposed [7] and investigated. Thus, now there is a scope of CA diffusion models and their modifications with different characteristics. The fact is, the information about them is not complete and is scattered in different fields in the journals. Moreover, three-dimensional CA-models are not studied at all, although it is the case that is usually needed in large-scale applications. Thus, the complete systematized data concerning existing CA-models, a comparative analysis of their characteristics would be useful for developing CA-models of a wide range of natural phenomena and technological processes. This is the reason to do the work presented in this paper. In the next section, the formalisms used for CA-models presentations are given. In the third section, all known diffusion 2D and 3D CA-models are described formally, and their charcteristics are summarized, some of them, unknown before, being obtained by simulation. The fourth section contains a comparative analysis of data presented from the point of view of multicomputer implementation.

2. A formal representation of a CA-model

A CA is a set of identical computing units, denoted by the pairs (u, x), called *cells*, where $u \in A$ is a *cell state* from the alphabet A, $\boldsymbol{x} \in X$ is a *name*, often being given as a vector $\boldsymbol{x} = (i, j, k)$ from a set of coordinates of a finite b-dimensional ($b = 1, 2, 3$) discrete space with $i = 0, \ldots, I$; $j = 0, \ldots, J$; $k = 0, \ldots, K$. The set of cells $\Omega = \{(u_i, \boldsymbol{x}_i) : \ u \in A, \ \boldsymbol{x} \in X, \ \boldsymbol{x}_i \neq \boldsymbol{x}_j\}$ is referred to as a *cellular array*, and a list of states of cells from Ω — as a CA *global state* $\Omega_A = (u_1, u_2, \ldots, u_{|X|})$. A subset of cells in the neighborhood of any $\boldsymbol{x} \in X$

$$S(\boldsymbol{x}) = \{(u_0, \boldsymbol{x}), (u_1, \boldsymbol{x} + \boldsymbol{a}_1), \ldots, (u_{n-1}, \boldsymbol{x} + \boldsymbol{a}_{n-1})\} \tag{1}$$

forms a *local configuration* with an underlying template

$$T(\boldsymbol{x}) = \{\boldsymbol{x}, \boldsymbol{x} + \boldsymbol{a}_1, \ldots, \boldsymbol{x} + \boldsymbol{a}_{n-1}\}, \tag{2}$$

where $\boldsymbol{a}_j$ is a shift vector.

The CA functioning is determined by a *local operator* $\Theta(\boldsymbol{x})$

$$\Theta(\boldsymbol{x}) = \Phi(\theta_1(\boldsymbol{x}), \ldots, \theta_n(\boldsymbol{x})) \tag{3}$$

that is a composition of several substitutions of the form

$$\theta(\boldsymbol{x}) : S(\boldsymbol{x}) \to S'(\boldsymbol{x}), \tag{4}$$

where $S(\boldsymbol{x})$ and $S'(\boldsymbol{x})$ differ in the cell states. A substitution $\theta(\boldsymbol{x})$ is applicable to a cell $(u, \boldsymbol{x}) \in \Omega$, if $S(\boldsymbol{x}) \in \Omega$. Application of $\theta(\boldsymbol{x})$ results in replacing the states of cells $(u_j, \boldsymbol{x}) \in S(\boldsymbol{x})$ by

$$u'_j = f_j(u_1, \ldots, u_n), \quad n = |S(\boldsymbol{x})|, \quad j = 0, \ldots, |S'(\boldsymbol{x})|, \tag{5}$$

where $f_j(u_1, \ldots, u_n)$ is a *transition function.*

Local operator (3) determines the order of its substitutions application to any cell [8]. The most frequently used local operators are:

- Φ_S — a sequential application of all substitutions and
- Φ_R — application of one randomly chosen substitution.

Application of $\Theta(\boldsymbol{x})$ to all $\boldsymbol{x} \in X$ comprises a *global operator* $\Theta(X)$. Its application changes $\Omega(t)$ for $\Omega(t+1)$, constituting the t-th *iteration* of the *CA evolution*, the latter being a sequence

$$\Omega(0),\ \Omega(1),\ \ldots,\ \Omega(T).$$

A global operator $\Theta(X)$ is *correct*, i.e. no loss of data can be caused by its execution [9], if the underlying templates of substitutions, which may be applied simultaneously, do not intersect:

$$T'_k(\boldsymbol{x}) \cap T'_m(\boldsymbol{y}) = \emptyset, \quad \forall \boldsymbol{x}, \boldsymbol{y} \in X, \quad \forall k, m \in \{1, \ldots, n\}, \quad n = |\Theta(\boldsymbol{x})|. \tag{6}$$

There are several modes of the global transition execution, the main of them being as follows:

Synchronous mode implies the following sequence of actions:

1. For all $(u, \boldsymbol{x}) \in \Omega(t)$ new states are computed by (5);
2. In all cells $(u, \boldsymbol{x}) \in \Omega(t)$ the states $u(\boldsymbol{x})$ are replaced by $u'(\boldsymbol{x})$.

There are two ways to satisfy correctness condition (6) in the synchronous mode:

1. Substitutions (4) of $\Theta(\boldsymbol{x})$ should have a single cell base, i.e.

$$|S'(\boldsymbol{x})| = 1 \quad \forall \theta_i \in \Theta(\boldsymbol{x}); \tag{7}$$

2. The naming set should be partitioned into $|X|/|T| = m$ non intersecting subsets $X = X_1 \cup \ldots \cup X_l \cup \ldots \cup X_m$ in such a way, that $T(\boldsymbol{x}) \cap T(\boldsymbol{y}) = \emptyset$ for any pair $(\boldsymbol{x}, \boldsymbol{y})$ belonging to one and the same X_l, application of $\Theta(\boldsymbol{x})$ should be successively accomplished at m synchronous stages.

Asynchronous mode implies the following procedure of local operator application:

1. A cell $(u, \boldsymbol{x}) \in \Omega$ is chosen with probability $p = 1/|X|$;
2. $\Theta(\boldsymbol{x})$ is applied to a chosen cell and the base cells states $(u, \boldsymbol{x}) \in S(\boldsymbol{x})$ are immediately replaced by the corresponding $(u', \boldsymbol{x}) \in S'(\boldsymbol{x})$.

By condition, it is accepted that $|X|$ repetitions of this algorithm comprise an iteration. Such an agreement is helpful, because it is in accord with the synchronous mode and with a definition of a *step* in asynchronous CA, widely used in the simulation surface chemical reactions, and called by chemists *kinetic Monte Carlo method* [10].

Ordered asynchronous mode is a modification of the asynchronous mode, when $\Theta(\boldsymbol{x})$ is applied sequentially to the ordered cell set.

Due to the fact that in asynchronous CA a local operator is applied to $\boldsymbol{x} \in X$ sequentially, condition (6) is always satisfied. The problem of correct computation of asynchronous CA arises only if the functioning of the CA is executed in parallel on several processors.

Block-synchronous mode [7] is introduced for providing efficiency of asynchronous CA in parallel implementation. It is obtained by transformation of asynchronous mode into a sequence of synchronous computations, which makes the parallel computation to satisfy correctness condition (6).

Let $T(\boldsymbol{x})$ be the underlying template of $\Theta(\boldsymbol{x})$, and $B(\boldsymbol{x}) \supseteq T(\boldsymbol{x})$, then the block-synchronous transformation is as follows:

1. X is decomposed into $m = |B(\boldsymbol{x})|$ non-intersecting subsets $X = X_0 \cup, \ldots, X_l, \ldots, \cup X_m$ in such a way, that $\forall j \, \forall l \, \forall \boldsymbol{x}$:

$$|B_j(\boldsymbol{x}) \cap X_l| \leq 1. \tag{8}$$

2. At each t-th iteration a local operator $\Theta(\boldsymbol{x})$ is applied to $\Omega(t)$ at m successive stages, at each l-th stage — to all $\boldsymbol{x} \in X_l$ synchronously, $l = 0, \ldots, m$.
3. Data exchange between the adjacent subdomains of X is performed after each stage. Such a mode of functioning is further called *multistage synchronous mode*.

The mode of functioning is an essential parameter of a global operator, i.e. if two CA differ only in functioning modes, their evolutions may still be quite different. Thus, a global operator should be indexed by the mode of functioning, which is denoted as Θ_ρ, where $\rho \in \{\alpha, \beta, \sigma, \omega\}$, α standing for asynchronous mode, β — for block-synchronous, σ — for synchronous, and ω — for ordered asynchronous.

3. The CA models of diffusion

Diffusion CA belong to a large class of CA models where all operations may be interpreted as movements of abstract particles, represented as "ones" in CA with Boolean alphabet. In such actions, several neighboring cells have to change their states simultaneously. Hence, to satisfy correctness condition (6), asynchronous CA or synchronous multi-stage CA should be used. If a model is intended to be implemented on a multiprocessor system, asynchronous CA should be transformed into its block synchronous multistage version. Obviously, the less is the number of stages the higher is parallelization efficiency

Synchronous CA with Margolus' neighborhood. Among the known CA-models of diffusion, the most efficient parallel implementation seems to be the CA with Margolus' neighborhood [11, 12], which is a two-stage synchronous CA. Its functioning is performed according to the following algorithm:

1. Two subsets of cells are formed in $\Omega(t)$:

$$\Omega_0(t) = \{(u, \boldsymbol{x}) : \ i \bmod 2 = 0, \ j \bmod 2 = 0, \ k \bmod 2 = 0\},$$
$$\Omega_1(t) = \{(u, \boldsymbol{x}) : \ i \bmod 2 = 1, \ j \bmod 2 = 1, \ k \bmod 2 = 1\}.$$

   Each subset $\Omega_l(t)$, $l = 1, 2$, generates partitioning B_d of $\Omega(t)$ into $|\Omega(t)|/m$, $m = 2^d$, square ($d = 2$) or cubic ($d = 3$) blocks $B_d(\boldsymbol{x}) = \{\boldsymbol{x} + \boldsymbol{a}_0, \boldsymbol{x} + \boldsymbol{a}_1, \ldots, \boldsymbol{x} + \boldsymbol{a}_m\}$, where shift vectors for $d = 2$ are

$$\boldsymbol{a}_0 = (0, 0), \quad \boldsymbol{a}_1 = (0, 1), \quad \boldsymbol{a}_2 = (1, 1), \quad \boldsymbol{a}_3 = (1, 0), \tag{9}$$

and for $d = 3$

$$\begin{array}{llll} \boldsymbol{a}_0 = (0,0,0), & \boldsymbol{a}_1 = (0,1,0), & \boldsymbol{a}_2 = (1,1,0), & \boldsymbol{a}_3 = (1,0,0), \\ \boldsymbol{a}_4 = (0,0,1), & \boldsymbol{a}_5 = (0,1,1), & \boldsymbol{a}_6 = (1,1,1), & \boldsymbol{a}_7 = (1,0,1). \end{array} \tag{10}$$

2. Each t-th iteration is divided into 2 stages: $t = (t_0, t_1)$. At the j-th stage, a local operator Θ_d is applied to all $\boldsymbol{x} \in \Omega_j(t)$.
3. The substitutions of Θ_d are constructed based on the following pattern:

$$\begin{array}{rl} \theta : & \{(u_0, \boldsymbol{y}_0)(u_1, \boldsymbol{y}_1)(u_2, \boldsymbol{y}_2)(u_3, \boldsymbol{y}_3)\} \xrightarrow{p} \\ & \{(u_1, \boldsymbol{y}_0)(u_2, \boldsymbol{y}_1)(u_3\boldsymbol{y}_2)(u_0, \boldsymbol{y}_3)\}, \\ \theta' : & \{(u_0, \boldsymbol{y}_0)(u_1, \boldsymbol{y}_1)(u_2, \boldsymbol{y}_2)(u_3, \boldsymbol{y}_3)\} \xrightarrow{1-p} \\ & \{(u_3, \boldsymbol{y}_0)(u_0, \boldsymbol{y}_1)(u_1, \boldsymbol{y}_2)(u_2, \boldsymbol{y}_3)\}. \end{array} \tag{11}$$

For the two-dimensional case $\Theta_2 = \Phi_R(\theta_0, \theta'_0)$, the substitutions being obtained from θ, θ' (11) by setting $\boldsymbol{y}_q$ equal to $\boldsymbol{x} + \boldsymbol{a}_q$, $q = 0, 1, 2, 3$, $\boldsymbol{a}$-from (9).

For the three-dimensional case $\Theta_3 = \Phi_R(\theta_0, \theta'_0 \ldots, \theta_5, \theta'_5)$, substitutions θ_l, θ'_l being obtained by replacing $\boldsymbol{y}_q$ by $\boldsymbol{x} + \boldsymbol{a}_q$, $\boldsymbol{a}$ from (10), and also by replacing $(u_q, \boldsymbol{y}_q)$ by a pair of the adjacent cells according to the following formulas:

$$\begin{array}{llll} (u_q, \boldsymbol{y}_0) = \{(u_0, \boldsymbol{x}+\boldsymbol{a}_0), (u_1, \boldsymbol{x}+\boldsymbol{a}_1)\}, & (u_q, \boldsymbol{y}_1) = \{(u_4, \boldsymbol{x}+\boldsymbol{a}_4), (u_5, \boldsymbol{x}+\boldsymbol{a}_5)\}, \\ (u_q, \boldsymbol{y}_2) = \{(u_6, \boldsymbol{x}+\boldsymbol{a}_6), (u_7, \boldsymbol{x}+\boldsymbol{a}_7)\}, & (u_q, \boldsymbol{y}_3) = \{(u_2, \boldsymbol{x}+\boldsymbol{a}_2), (u_3, \boldsymbol{x}+\boldsymbol{a}_3)\}, \\ (u_q, \boldsymbol{y}_4) = \{(u_0, \boldsymbol{x}+\boldsymbol{a}_0), (u_3, \boldsymbol{x}+\boldsymbol{a}_3)\}, & (u_q, \boldsymbol{y}_5) = \{(u_1, \boldsymbol{x}+\boldsymbol{a}_1), (u_2, \boldsymbol{x}+\boldsymbol{a}_2)\}, \\ (u_q, \boldsymbol{y}_6) = \{(u_5, \boldsymbol{x}+\boldsymbol{a}_5), (u_6, \boldsymbol{x}+\boldsymbol{a}_6)\}, & (u_q, \boldsymbol{y}_7) = \{(u_4, \boldsymbol{x}+\boldsymbol{a}_4), (u_7, \boldsymbol{x}+\boldsymbol{a}_7)\}, \\ (u_q, \boldsymbol{y}_8) = \{(u_3, \boldsymbol{x}+\boldsymbol{a}_3), (u_7, \boldsymbol{x}+\boldsymbol{a}_7)\}, & (u_q, \boldsymbol{y}_9) = \{(u_0, \boldsymbol{x}+\boldsymbol{a}_0), (u_4, \boldsymbol{x}+\boldsymbol{a}_4)\}, \\ (u_q, \boldsymbol{y}_{10}) = \{(u_1, \boldsymbol{x}+\boldsymbol{a}_1), (u_5, \boldsymbol{x}+\boldsymbol{a}_5)\}, & (u_q, \boldsymbol{y}_{11}) = \{(u_2, \boldsymbol{x}+\boldsymbol{a}_2), (u_6, \boldsymbol{x}+\boldsymbol{a}_6)\}. \end{array}$$

In other words, the 2D diffusion algorithm is as follows: at the j-th stage ($j = 1, 2$) each 2×2 block of Ω_j rotates its cell states clockwise (θ_0) or counterclockwise (θ'_0) depending on a random number and a given probability relation (Figure 1). In the 3D diffusion algorithm, at each j-th stage three similar operations are executed, each rotating four pairs of the adjacent cell states around the axis that is parallel to the segment between the pair cells centers (Figure 2). In both cases for the isotropic diffusion the substitutions in $\Theta_d(\boldsymbol{x})$ are chosen with equal probability.

Binate diffusion CA model has a local operator, consisting of a single substitution defined on a pair of the adjacent cells (thereafter the name of a model). Although it is the most simple operator, it requires $2d$ subsets to be formed out of the cellular array, and, hence, $2d$ interprocessor data exchanges are needed to be done per iteration.

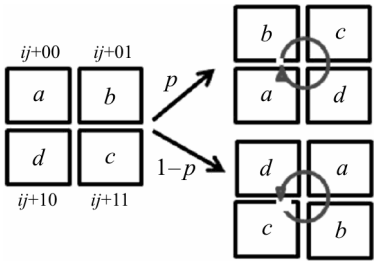

Figure 1. Graphical representation of a local operator $\Theta_2(\boldsymbol{x})$

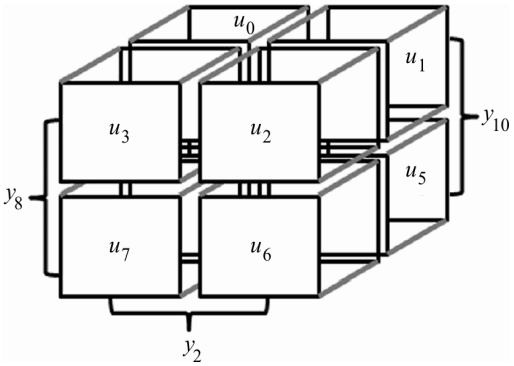

Figure 2. 3D block of $B_3(\boldsymbol{x})$

The subsets of cells $\Omega_l(t)$ that are processed at each lth stage, $l = 0, \ldots, 2d$, are as follows:

$$\Omega_0(t) = \{(u,(i,j,k)) :\ i \bmod 2 = 0,\ j = 0,\ldots,J,\ k = 0,\ldots,K\},$$
$$\Omega_1(t) = \{(u,(i,j,k)) :\ i \bmod 2 = 1,\ j = 0,\ldots,J,\ k = 0,\ldots,K\},$$
$$\Omega_2(t) = \{(u,(i,j,k)) :\ j \bmod 2 = 0,\ i = 0,\ldots,I,\ k = 0,\ldots,K\},$$
$$\Omega_3(t) = \{(u,(i,j,k)) :\ j \bmod 2 = 1,\ i = 0,\ldots,I,\ k = 0,\ldots,K\},$$
$$\Omega_4(t) = \{(u,(i,j,k)) :\ k \bmod 2 = 0,\ i = 0,\ldots,I,\ j = 0,\ldots,J\},$$
$$\Omega_5(t) = \{(u,(i,j,k)) :\ k \bmod 2 = 1,\ i = 0,\ldots,I,\ j = 0,\ldots,J\},$$

A local operator $\Theta(\boldsymbol{x})$ consists of a single substitution

$$\theta(\boldsymbol{x}) : \{(u_0,\boldsymbol{x}),(u_l,\boldsymbol{x}+\boldsymbol{a}_l\} \to \{(u_l,\boldsymbol{x}),(u_0,\boldsymbol{x}+\boldsymbol{a}_l\}, \tag{12}$$

where for $d = 2$: $l = i, j$, $\boldsymbol{a}_i = (01)$, $\boldsymbol{a}_j = (10)$, for $d = 3$: $l = i, j, k$, $\boldsymbol{a}_i = (001)$, $\boldsymbol{a}_j = (010)$, $\boldsymbol{a}_k = (100)$. At each l-th stage, $\theta(\boldsymbol{x})$ is applied to a randomly chosen $\boldsymbol{x} \in \Omega_l(t)$

Naive diffusion CA model. The well-known *naive* diffusion CA [11] in its classic asynchronous case is defined by a single substitution local operator

$$\theta(\boldsymbol{x}) : \{(u_0,\boldsymbol{x}+\boldsymbol{a}_0),\ldots,(u_l,\boldsymbol{x}+\boldsymbol{a}_l),\ldots,(u_d,\boldsymbol{x}+\boldsymbol{a}_d)\} \to$$
$$\{(u_l,\boldsymbol{x}+\boldsymbol{a}_0),\ldots,(u_0,\boldsymbol{x}+\boldsymbol{a}_l),\ldots,(u_d,\boldsymbol{x}+\boldsymbol{a}_d)\}, \tag{13}$$

where l is a randomly chosen number from $\{1,\ldots,2d\}$.

A block synchronous version of the naive diffusion has $2d+1$ stages, being applied to the following subsets. For a two-dimensional case, $l = 0,\ldots,4$, $\boldsymbol{x} = (i,j)$, $\boldsymbol{a}_l : (a_l, b_l)$,

$$B_l = \{(i,j) : (i \bmod 5 = a_l,\ j \bmod 5 = b_l) \vee (i \bmod 5 = a_l + 1,\ j \bmod 5 = b_l + 2)\}, \quad (14)$$

where $\boldsymbol{a}_0 = (0,0)$, $\boldsymbol{a}_1 = (1,0)$, $\boldsymbol{a}_2 = (0,1)$, $\boldsymbol{a}_3 = (4,0)$, $\boldsymbol{a}_4 = (0,4)$.

For a three-dimensional case, $l = 0,\ldots,6$, $\boldsymbol{x} = (i,j,k)$, $\boldsymbol{a}_l = (a_l, b_l, c_l)$,

$$B_l = \{(i,j,k) : (i \bmod 5 = a_l,\ j \bmod 5 = b_l, k \bmod 5 = c_l) \vee (i \bmod 5 = a_l + 1,\ j \bmod 5 = b_l + 1,\ k \bmod 5 = c_l + 1) \vee (i \bmod 5 = a_l + 1,\ j \bmod 5 = b_l + 2,\ k \bmod 5 = c_l)\}, \quad (15)$$

where $\boldsymbol{a}_0 = (0,0,0)$, $\boldsymbol{a}_1 = (1,0,0)$, $\boldsymbol{a}_2 = (0,1,0)$, $\boldsymbol{a}_3 = (0,0,1)$, $\boldsymbol{a}_4 = (4,0,0)$, $\boldsymbol{a}_5 = (0,4,0)$, $\boldsymbol{a}_6 = (0,0,4)$.

There are many other block-synchronous versions of the naive diffusion (13) that are obtained by taking B_l having $m \times m$ redundant size, i.e. $B_l(\boldsymbol{x}) \supset T(\boldsymbol{x})$, with $m = 3,5,7,\ldots$ Obviously, they are executed on m^d subsets of the form

$$B_l = \{(i,j,k) :\ i \bmod m = a_l,\ j \bmod m = b_l,\ k \bmod m = c_l\}.$$

The diffusion CA belong to a large class of CA models where all operations may be interpreted as movements of abstract particles, represented as "ones" in CA with Boolean alphabet. When simulating real natural phenomena, where all values are assessed in real numbers, we come up against a new problem of transforming spatial functions into Boolean array and vice versa. The problem has been solved by using two operators: *discretization* and *averaging* [13]. The operator of averaging summarizes the state values in the vicinity of any $\boldsymbol{x} \in X$ and divides the sum by the number of cells in this area. If the area is small, the averaging results in a non-smooth function, i.e. having a big "automata noise". Averaging over a large area distorts the diffusion effect [14]. When averaging is used only at the end of simulation, the distortion is insignificant, but being used at each iteration, the automata noise becomes harmful. In such cases the following diffusion CA-model is helpful.

Diffusion CA-model with integer alphabet is proposed in [15], where it is called "the multi-particle diffusion model". Is shown there that using the integer alphabet instead of Boolean one allows one to decrease the automata noise, and, hence, to eliminate the process of averaging. Moreover, the model effectively combines with CA-models that have integer alphabets [16].

The main idea is to represent the cell states as a sum of two integers $u = \gamma u + (1-\gamma)u$, $\gamma \in (0,1]$, only one of them being transferred into the neighboring cell.

Such a modification may be applied to any asynchronous diffusion CA as follows:

1. The alphabet of a CA is a finite set of non-negative integers: $A = \{0, 1, \ldots, M\}$, M being chosen according to the requirement of averaging accuracy, or compatibility with other simulation system component.
2. Each state $u \in A$ is represented as a sum

$$u = u' + u'', \tag{16}$$

where $u' = \lfloor u \rfloor \gamma + \delta$, $u'' = u - u'$, $\delta = 1$ with probability $p = (u - \lfloor u \rfloor)\gamma$, and $\delta = 0$, otherwise.

3. Each substitution in local operator (3), which simulates the exchange of states between the neighboring cells is replaced by

$$\begin{aligned} \theta(\boldsymbol{x}) : \{(u'_0 + u''_0), \boldsymbol{x}), (u'_l + u''_l), \boldsymbol{x} + \boldsymbol{a}_l\} \rightarrow \\ \{(u''_0 + u'_l, \boldsymbol{x}) + (u''_l + u'_0, \boldsymbol{x} + \boldsymbol{a}_l)\}, \end{aligned} \tag{17}$$

The above principle of constructing the integer version Int-CA may be applied to any asynchronous diffusion CA model. The obtained Int-CA would differ from its Boolean counterpart Bool-CA $\Theta_{\mathrm{Bool-CA}}(\boldsymbol{x})$ in its substitutions, being constructed according to (16) or (17). The diffusion number of integer CA $D(\text{Int-CA}) = \gamma D(\text{Bool-CA})$. In the multi-processor implementation, block-synchronous transformations of integer CA should be used, being constructed in a similar way as their Boolean counterparts and having the same numbers of exchanges per iteration.

4. The main characteristics of CA diffusion models

Three characteristics of the diffusion CA models are important for designing large-sale simulation programs, including CA as a component: diffusivity, the number of synchronous stages, the time of computation. *Diffusivity* is a characteristic that may be expressed in terms of physical parameters of the process simulated by the CA evolution, hence, it is called a *CA invariant* [17]. The invariant is a dimensionless characteristic of the process that does not depend on its mathematical representation allowing one to determine the scaling coefficients between model parameters ant their physical counterparts. Since the CA evolution simulates spatial dynamics, its invariant should represent the relation between spatial and temporal scales. In the case of diffusivity, such a relation is given by the formula

$$D = \frac{c_d \tau}{h^2},$$

where c_d (m^2/s) is a diffusion coefficient, τ (s) is iteration time, and h (m) is the linear size of a cell. Diffusion numbers (D) and the number of synchronous stages (#) of the above CA-models are summarized in the table:

d	Marg-N		Binate		Naive-3		Naive-m	
	D	#	D	#	D	#	D	#
2	1	2	0.75	4	0.5	9	0.5	m^2
3	1	2	0.75	6	0.33	27	0.5	m^3

5. Conclusion

The CA diffusion models are considered and analyzed from the point of view of using them in large-scale complex tasks to be implemented on supercomputers. The models are characterized by their diffusion numbers and the numbers of synchronous stages equal to the number of interprocessor exchanges per iteration. The information about these two characteristics is of great importance when developing a parallel algorithm for simulating complex phenomena by CA.

References

[1] Simulating Complex Systems by Cellular Automata. Understanding Complex Systems / A.G. Hoekstra, et al., eds. — Berlin: Springer, 2010.

[2] Bandman O. Cellular Automata Composition Techniques for Spatial Dynamics Simulation // Simulating Complex Systems by Cellular Automata. Understanding Complex Systems / A.G. Hoekstra et al. eds. — Berlin: Springer, 2010. — P. 81–115.

[3] Bandman O. Using cellular automata for porous media simulation // J. Supercomputing. — 2011. — Vol. 57, No. 2. — P. 121–131.

[4] Medvedev Yu. Dynamic load balancing for lattice gas simulations on a cluster // Proc. Int. Conf. on Parallel Computing Technologies, PaCT-2011 / V. Malyshkin, ed. — 2011. — P. 181–191. — (LNCS; 6873).

[5] Bandman O. Implementation of large-scale cellular automata models on multicore computers and clusters // High Performance Computing and Simulation (HPCS), 2013 Int. Conf. / IEEE Conference Publications. — 2013. — P. 304–310.

[6] Kalgin K. Parallel implementation of asynchronous cellular automama on 32-core computer // Siberian J. Num. Math. — Novosibirsk: SBRAS Press, 2012. — Vol. 15, No. 1. — P. 55–65.

[7] Bandman O. Parallel simulation of asynchronous cellular automata evolution // Proc. 7th Int. Conf. on Cellular Automata for Research and Industry, ACRI 2006 / S. El Yacoubi, B. Chopard, S. Bandini, eds. — 2006. — P. 41–48. — (LNCS; 4173).

[8] Achasova S., Bandman O., Markova V., Piskunov S. Parallel Substitution Algorithm. Theory and Application. — Singapore: World Scientific, 1994.

[9] Achasova S., Bandman O. Correctness of Parallel Processes. — Novosibirsk: Nauka, 1999 (In Russian).

[10] Jansen A.P.J. An Introduction to Monte-Carlo Simulation of Surface Reactions. — arXiv:cond-mat/0303028v1[stst-mech], 2003.

[11] Toffolli T., Margolus N. Cellular Automata Machines: A New Environment for Modeling. — USA: MIT Press, 1987.

[12] Malinetsky G.G., Stepantzov M.E. Simulating diffusion processes by cellular automata with Margolus' neighborhood // J. Comp. Math. and Math. Physics. — 1998. — Vol. 36, No. 6. — P. 1017–1021.

[13] Bandman O. Algebraic Properties of Cellular Automata : the Basis for Composition Technique // Proc. 6th Int. Conf. on Cellular Automata for Research and Industry, ACRI 2004 / P.M.A. Sloot, B.Chopard, A.G. Hoekstra, eds. — 2004. — P. 688–697. — (LNCS; 3305).

[14] [14] Weimar J. Cellular Automata for Reaction Diffusion Systems // Parallel Computing. — 1997. — Vol. 23, No. 11. — P. 1699–1715.

[15] Medvedev Yu.G. Multiparticle Cellular Automata for Diffusion Simulation // MTPP Proc. — Berlin: Springer, 2011. — P. 204–211. — (LNCS; 6083).

[16] Afanasiev I. The CA-model of populations' dynamics of organisms living in Baikal // Bull. Novosibirsk Comp. Center. Ser. Computer Science. — Novosibirsk, 2012. — Iss. 33. — P. 11–23.

[17] Bandman O. The concept of invariants in reaction-diffusion cellular-automata // Bull. Novosibirsk Comp. Center. Ser. Computer Science. — Novosibirsk, 2012. — Iss. 33. — P. 24–36.