A Simple Optimum-Time FSSP Algorithm for Multi-Dimensional Cellular Automata

HIROSHI UMEO
University of Osaka Electro-Communication
umeo@cyt.osakac.ac.jp

KINUO NISHIDE
University of Osaka Electro-Communication
nishide@cyt.osakac.ac.jp

KEISUKE KUBO
University of Osaka Electro-Communication
kubo@cyt.osakac.ac.jp

The firing squad synchronization problem (FSSP) on cellular automata has been studied extensively for more than forty years, and a rich variety of synchronization algorithms have been proposed for not only one-dimensional arrays but two-dimensional arrays. In the present paper, we propose a simple recursive-halving based optimum-time synchronization algorithm that can synchronize any rectangle arrays of size $m \times n$ with a general at one corner in $m+n+\max(m,n)-3$ steps. The algorithm is a natural expansion of the well-known FSSP algorithms proposed by Balzer [1967], Gerken [1987], and Waksman [1966] and it can be easily expanded to three-dimensional arrays, even to multi-dimensional arrays with a general at any position of the array. The algorithm proposed is isotropic concerning the side-lengths of multi-dimensional arrays and its algorithmic correctness is transparent and easily verified.

1 Introduction

We study a synchronization problem that gives a finite-state protocol for synchronizing large-scale cellular automata. The synchronization in cellular automata has been known as a firing squad synchronization problem (FSSP) since its development, in which it was originally proposed by J. Myhill in Moore [1964] to synchronize all parts of self-reproducing cellular automata. The problem has been studied extensively for more than forty years [1-23], and a rich variety of synchronization algorithms have been proposed for not only one-dimensional (1D) arrays but two-dimensional (2D) arrays. The 1D FSSP is described as follows: given a one-dimensional array of n identical cellular automata, including a *general* at one end that is activated at time $t = 0$, we want to design the automata such that, *at some future time*, all the cells will *simultaneously* and, *for the first time*, enter a special *firing* state.

Some questions may arise:

– How can we synchronize multi-dimensional arrays?

– Can we expand those 2D FSSP algorithms proposed so far to 3D arrays, or more generally to multi-dimensional arrays?

– Can we generalize those 2D FSSP algorithms to generalized ones, in which an initial general is located at any position of the array?

– What is a lower bound of time steps needed for synchronizing multi-dimensional arrays with a general at one corner?

– What is a lower bound for the generalized case?

In the present paper, we attempt to answer these questions by proposing a new, simple recursive-halving based synchronization algorithm for 2D rectangle cellular automata. The algorithm can synchronize any 2D rectangle array of size $m \times n$ with a general at one corner in $m + n + \max(m,n) - 3$ steps. An implementation in terms of local transition rules is also given on a 2D cellular automaton, not only for the array with a general at one corner but a generalized case.

The algorithms proposed in this paper are interesting in the following view points.

- The 2D algorithm proposed is isotropic with respect to shape of a given rectangle array, i.e. no need to control the FSSP algorithm for longer-than-wide and wider-than-long input rectangles.
- The algorithm proposed can be easily expanded to 3D arrays, even to multi-dimensional arrays.
- The algorithm is a natural generalization of the well-known 1D FSSP algorithms developed by Waksman [1966], Balzer [1967] and Gerken [1987]. It gives us a new view point of those classical 1D FSSP algorithms based on recursive-halving.
- The algorithm can be expanded to a generalized FSSP solution, where an initial general is at an arbitrary position of a given array.
- The algorithm can be generalized to an optimum-time generalized FSSP solution.

In Section 2 we give a description of the 2D FSSP and review some basic results on 2D FSSP algorithms. Section 3 defines the recursive-halving marking on 1D arrays and gives some preliminary lemmas for the construction of 2D FSSP algorithms. In Sections 4, 5, and 6 we present a new 2D FSSP algorithm based on the recursive-halving marking and several multi-dimensional expansions. Two implementations in terms of 2D cellular automata are also presented for the optimum-time FSSP algorithms. Most of the descriptions of the multi-dimensional FSSP algorithms are based on the 2D FSSP algorithms. Some expanded and generalized theorems for multi-dimensional arrays are given without proofs.

2 Firing Squad Synchronization Problem on Two-Dimensional Arrays

Figure 1 shows a finite two-dimensional (2D) cellular automaton consisting of $m \times n$ cells. Each cell is an identical (except the border cells) finite-state automaton. The array operates in lock-step mode in such a way that the next state of each cell (except border cells) is determined by both its own present state and the present states of its north, south, east and west neighbors. All cells (*soldiers*), except the north-west corner cell (*general*), are initially in the quiescent state at time $t = 0$ with the property that the next state of a quiescent cell with quiescent neighbors is the quiescent state again. At time $t = 0$, the north-west corner cell C_{11} is in the *fire-when-ready* state, which is the initiation signal for the array. The firing squad synchronization problem is to determine a description (state set and next-state function) for cells that ensures all cells enter the *fire* state at exactly the same time and for the first time. The tricky part of the problem is that the same kind of soldier having a fixed number of states must be synchronized, regardless of the size $m \times n$ of the array. The set of states and next state function must be independent of m and n.

The problem was first solved by J. McCarthy and M. Minsky who presented a $3n$-step algorithm for 1D cellular array of length n. In 1962, the first optimum-time, i.e. $(2n-2)$-step, synchronization algorithm was presented by Goto [1962], with each cell having several thousands of states. Waksman [1966] presented a 16-state optimum-time synchronization algorithm. Afterward, Balzer [1967] and Gerken [1987] developed an eight-state algorithm and a seven-state synchronization algorithm, respectively, thus decreasing the number of states required for the synchronization. Mazoyer [1987] developed

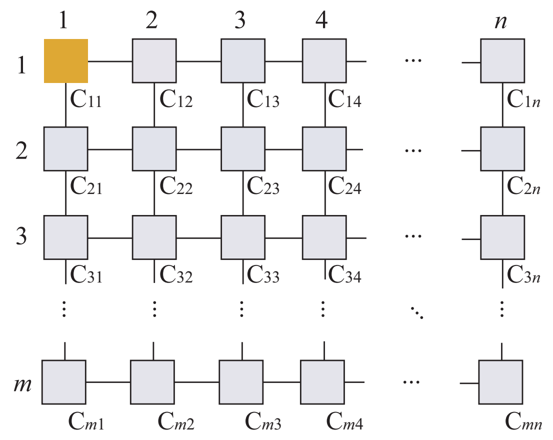

Figure 1: A two-dimensional (2D) cellular automaton.

a six-state synchronization algorithm which, at present, is the algorithm having the fewest states for 1D arrays.

On the other hand, several synchronization algorithms on 2D arrays have been proposed by Beyer [1969], Grasselli [1975], Shinahr [1974], Szwerinski [1982], Schmid [2003], Schmid and Worsch [2004], Umeo, Maeda, Hisaoka and Teraoka [2006], and Umeo and Uchino [2008]. It has been shown independently by Beyer [1969] and Shinahr [1974] that there exists no 2D cellular automaton that can synchronize any 2D array of size $m \times n$ in less than $m+n+\max(m,n)-3$ steps. In addition they first proposed an optimum-time synchronization algorithm that can synchronize any 2D array of size $m \times n$ in optimum $m+n+\max(m,n)-3$ steps. Shinahr [1974] gave a 28-state implementation. Umeo, Hisaoka and Akiguchi [2005] presented a new 12-state synchronization algorithm operating in optimum-step, realizing a smallest solution to the rectangle synchronization problem at present.

As for the time optimality of the 2D FSSP algorithms, the following theorems have been shown.

Theorem 1 Beyer [1969], Shinahr [1974] There exists no cellular automaton that can synchronize any 2D array of size $m \times n$ in less than $m+n+\max(m,n)-3$ steps, where the general is located at one corner of the array.

Theorem 2 Shinahr [1974], Umeo, Hisaoka, and Akiguchi [2005] There exists a cellular automaton that can synchronize any 2D array of size $m \times n$ at exactly $m+n+\max(m,n)-3$ steps, where the general is located at one corner of the array.

3 Recursive-Halving Marking

In this section, we develop a marking schema for 1D arrays referred to as *recursive-halving marking*. The marking schema prints a special mark on cells in a cellular space defined by the recursive-halving marking. The marking itself is based on a 1D FSSP synchronization algorithm. It will be effectively used for constructing multi-dimensional FSSP algorithms operating in optimum-time.

Let S be a 1D cellular space consisting of cells C_i, C_{i+1}, ..., C_j, denoted by $[i...j]$, where $j > i$. Let $|S|$ denote the number of cells in S, that is $|S| = j-i+1$. A center cell(s) C_x of S is defined by

$$x = \begin{cases} (i+j)/2 & |S|\text{: odd} \\ (i+j-1)/2, (i+j+1)/2 & |S|\text{: even.} \end{cases} \tag{1}$$

The recursive-halving marking for a given cellular space S = [1...n] is defined as follows:

Recursive-Halving Marking: RHM

```
Algorithm RHM($S$)
begin
    if $|S| \geq 2$ then
        if $|S|$ is odd then
            mark a center cell $C_x$ in $S$;
            $S_L$:= [1...$x$]; $S_R$:= [$x$...$n$];
            $\mathbf{RHM}_{\mathrm{L}}(S_L)$; $\mathbf{RHM}_{\mathrm{R}}(S_R)$;
        else
            mark center cells $C_x$ and $C_{x+1}$ in $S$;
            $S_L$:= [1...$x$]; $S_R$:= [$x+1$...$n$];
            $\mathbf{RHM}_{\mathrm{L}}(S_L)$; $\mathbf{RHM}_{\mathrm{R}}(S_R)$;
end
```

Left-Side Recursive-Halving Marking: $\mathrm{RHM_L}$

```
Algorithm $\mathbf{RHM}_{\mathrm{L}}(S)$
begin
    while $|S| > 2$ do
        if $|S|$ is odd then
            mark a center cell $C_x$ in $S$;
            $S_L$:= [1...$x$]; $\mathbf{RHM}_{\mathrm{L}}(S_L)$;
        else
            mark center cells $C_x$ and $C_{x+1}$ in $S$;
            $S_L$:= [1...$x$]; $\mathbf{RHM}_{\mathrm{L}}(S_L)$;
end
```

Right-Side Recursive-Halving Marking: $\mathrm{RHM_R}$

```
Algorithm $\mathbf{RHM}_{\mathrm{R}}(S)$
begin
    while $|S| > 2$ do
        if $|S|$ is odd then
            mark a center cell $C_x$ in $S$;
            $S_R$:= [$x$...$n$]; $\mathbf{RHM}_{\mathrm{R}}(S_R)$;
        else
            mark center cells $C_x$ and $C_{x+1}$ in $S$;
            $S_R$:= [$x+1$...$n$]; $\mathbf{RHM}_{\mathrm{R}}(S_R)$;
end
```

For example, we consider a cellular space S = [1...15] consisting of 15 cells. The first center cell is C_8, then the second one is C_4, C_5 and C_{11}, C_{12}, and the last one is C_2, C_3, C_{13}, C_{14}, respectively. In case S = [1...17], we get C_9, C_5, C_{13}, C_3, C_{15}, and C_2, C_{16} after four iterations.

Figure 2 (left) shows a space-time diagram for the marking. At time $t = 0$, the leftmost cell C_1 generates an infinite set of signals $w_1, w_2, ..., w_k$, .., each propagating in the right direction at $1/(2^k - 1)$ speed,

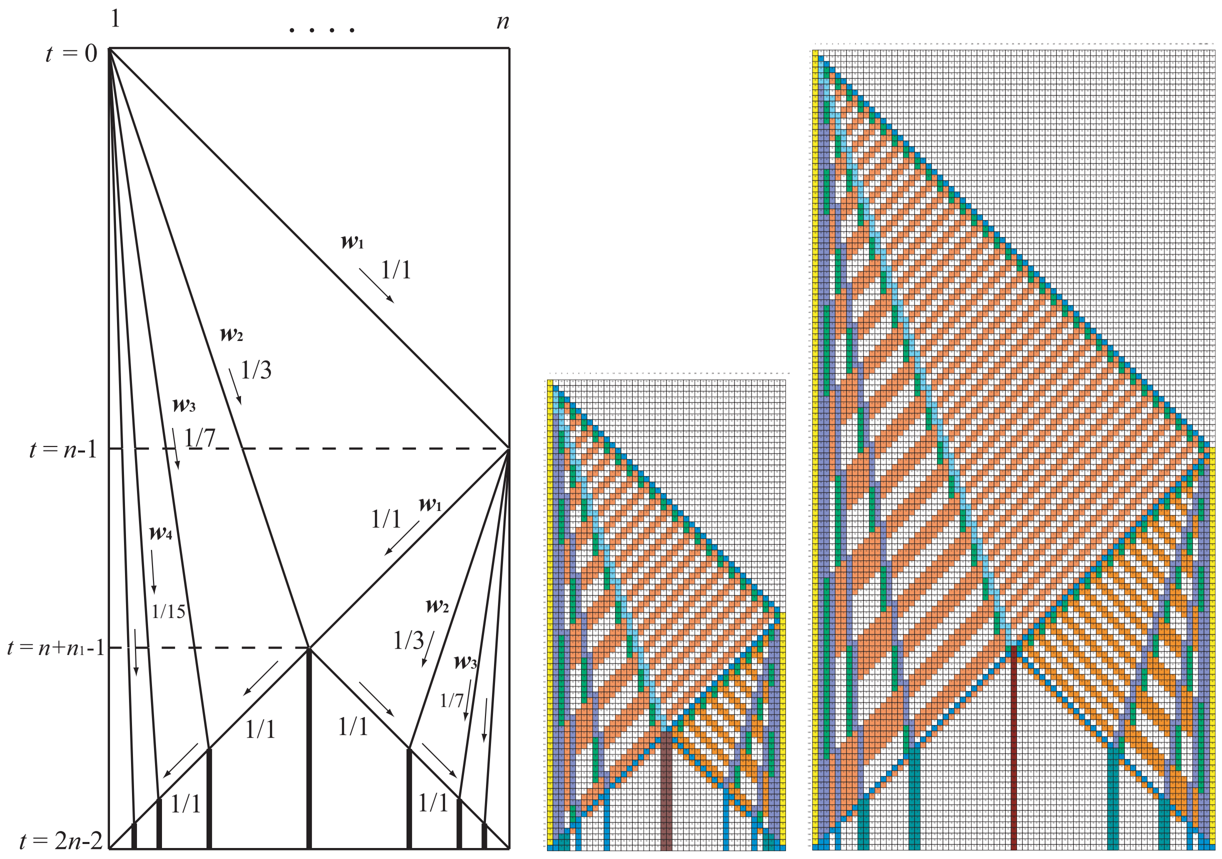

Figure 2: Space-time diagram for recursive-halving marking on 1D array of length n (left) and some snapshots for the marking on 42 (middle) and 71 (right) cells, respectively.

where $k = 1,2,3,...,$. The $1/1$-speed signal w_1 arrives at C_n at time $t = n-1$. Then, the rightmost cell C_n also emits an infinite set of signals $w_1, w_2, ..., w_k$, .., each propagating in the left direction at $1/(2^k-1)$ speed, where $k = 1,2,3,...,$. The readers can find that each crossing of two signals, shown in Fig. 2 (left), enables the marking at middle points defined by the recursive-halving. A finite state realization for generating the infinite set of signals above is a well-known technique employed in Balzer [1967], Gerken [1987], and Waksman [1966] for the implementations of the optimum-time synchronization algorithms on 1D arrays.

We have developed a simple implementation of the recursive-halving marking on a 13-state, 314-rule cellular automaton. In Fig. 2 (middle and right) we present several snapshots for the marking on 42 and 71 cells, respectively. Thus we have:

Lemma 3 There exists a 1D 13-state, 314-rule cellular automaton that can print the recursive-halving marking in any cellular space of length n in $2n-2$ steps.

An optimum-time complexity $2n-2$ needed for synchronizing cellular space of length n in the classical WBG-type (Waksman [1966], Balzer [1967], and Gerken [1987]) FSSP algorithms can be interpreted as follows: Let S be a cellular space of length $n = 2n_1+1$, where $n_1 \geq 1$. The first center mark in S is printed on cell C_{n_1+1} at time $t_{1D-\text{center}} = 3n_1$. Additional n_1 steps are required for the markings thereafter, yielding a final synchronization at time $t_{1D-opt} = 3n_1 + n_1 = 4n_1 = 2n - 2$. In the case $n = 2n_1$, where $n_1 \geq 1$, the first center mark is printed simultaneously on cells C_{n_1} and C_{n_1+1} at time $t_{1D-center} = 3n_1 - 1$. Additional $n_1 - 1$ steps are required for the marking and synchronization thereafter, yielding the final synchronization at time $t_{1D-opt} = 3n_1 - 1 + n_1 - 1 = 4n_1 - 2 = 2n - 2$.

$$t_{1D-center} = \begin{cases} 3n_1 & |S| = 2n_1 + 1, \\ 3n_1 - 1 & |S| = 2n_1. \end{cases} \tag{2}$$

Thus, additional $t_{1D-sync}$ steps are required for the synchronization for a cellular space with the recursive-halving marks:

$$t_{1D-sync} = \begin{cases} n_1 & |S| = 2n_1 + 1, \\ n_1 - 1 & |S| = 2n_1. \end{cases} \tag{3}$$

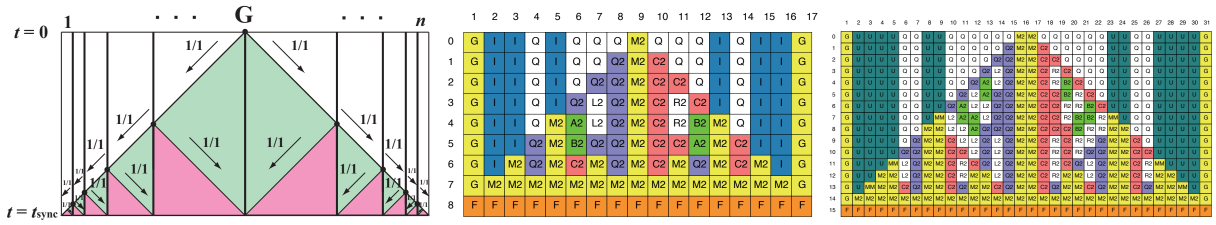

Figure 3: Space-time diagram for synchronizing a cellular space with recursive-halving marking (left) and some snapshots for the synchronization on 17 (middle) and 32 (right) cells, respectively.

In this way, it can be easily seen that any cellular space of length n with the recursive-halving marking initially with a general on a center cell or two generals on adjacent center cells can be synchronized in $\lceil n/2 \rceil - 1$ optimum-steps. In Fig. 3, we illustrate a space-time diagram for synchronizing a cellular space with recursive-halving marking (left) and some snapshots for the synchronization on 17 (middle) and 32 (right) cells, respectively. Thus we have:

Lemma 4 Any 1D cellular space S of length n with the recursive-halving marking initially with a general(s) on a center cell(s) in S can be synchronized in $\lceil n/2 \rceil - 1$ optimum-steps.

As was seen, the first marking of center cell(s) plays an important role. We print a special mark for the first center cell(s) of a given cellular space. On the other hand, for the center cells generated thereafter are marked with a different symbol from the first one.

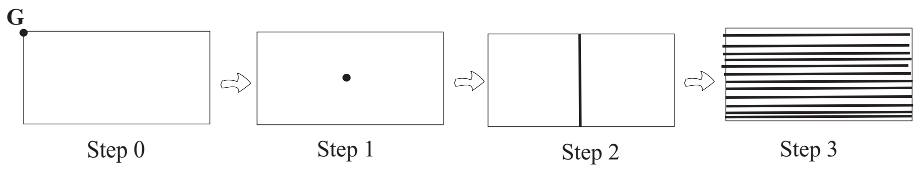

Figure 4: Synchronization schema for 2D cellular automaton.

4 An Optimum-Time 2D FSSP Algorithm $\mathscr{A}_1$

4.1 Overview of the Algorithm $\mathscr{A}_1$

We assume that an initial general G is on the north-west corner cell C_{11} of a given array of size $m \times n$. The algorithm consists of three phases: a marking phase, a pre-synchronization phase and a final synchronization phase. An overview of the 2D synchronization algorithm $\mathscr{A}_1$ is as follows:

Step 1. Start the recursive-halving marking for cells on each row and column, **find** a *center cell(s) of the given array*, and **generate** a new general(s) on the center cell(s). Note that a crossing(s) of the center column(s) with the center row(s) is a center cell(s) of the array.
Step 2. Pre-synchronize the center column(s) using Lemma 4, which is initiated by the general in Step 1. Every cell on the center column(s) acts as a general at the following Step 3.
Step 3. Synchronize each row using Lemma 4, initiated by the general generated in Step 2. This yields the final synchronization of the array.

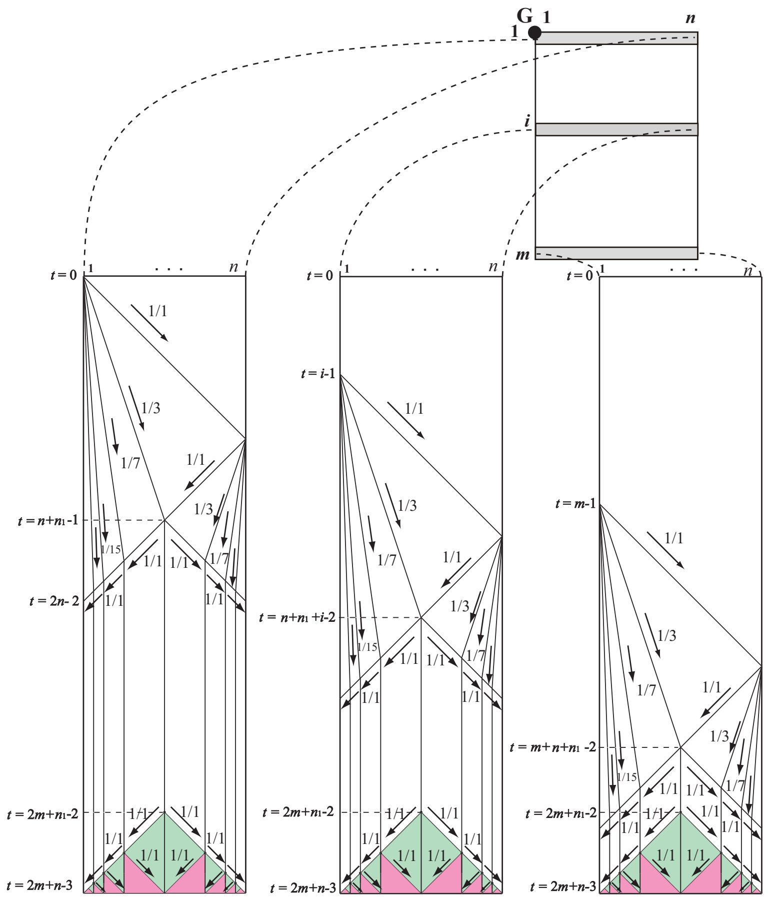

Figure 5: Space-time diagrams for the synchronization algorithm on the 1st, ith, and mth rows of a longer-than-wide rectangle array of size $m \times n$, respectively.

Figure 4 illustrates the synchronization schema for 2D cellular automaton. We assume that $m =$

$2m_1+1, n = 2n_1+1$, where $m_1, n_1 \geq 1$. The algorithm operates as follows:

1. At time $t = 0$ an initial general on the north-west corner emits a $1/1$-speed signal along the first row and column to print recursive-halving marks. Once a center mark is printed, it is copied to the adjacent row and column. At time $t = 3m_1+n_1$, a center mark of the center column of the array is marked, and the center mark of the center row is marked at time $t = 3n_1+m_1$. The center of the array is marked at time $t = t_{2\text{D}-\text{center}} = \max(3m_1+n_1, 3n_1+m_1)$.
2. Using Lemma 4, the center column will be synchronized with a tentative pre-firing state at time $t = t_{2\text{D}-\text{center}} + m_1$.
3. Once the center column could be synchronized with the pre-firing state, then the cell C_{i,n_1+1} initiates the synchronization for the ith row for each i such that $1 \leq i \leq m$. Using Lemma 4, for any $i, 1 \leq i \leq m$, the ith row will be synchronized at time $t = t_{2\text{D}-\text{center}} + m_1 + n_1 = \max(3m_1 + n_1, 3n_1+m_1) + m_1 + n_1 = \max(2m_1+1, 2n_1+1) + 2m_1 + 2n_1 - 1 = m+n+\max(m,n) - 3$. Thus, the array can be synchronized at time $t = m+n+\max(m,n)-3$ in optimum-steps.

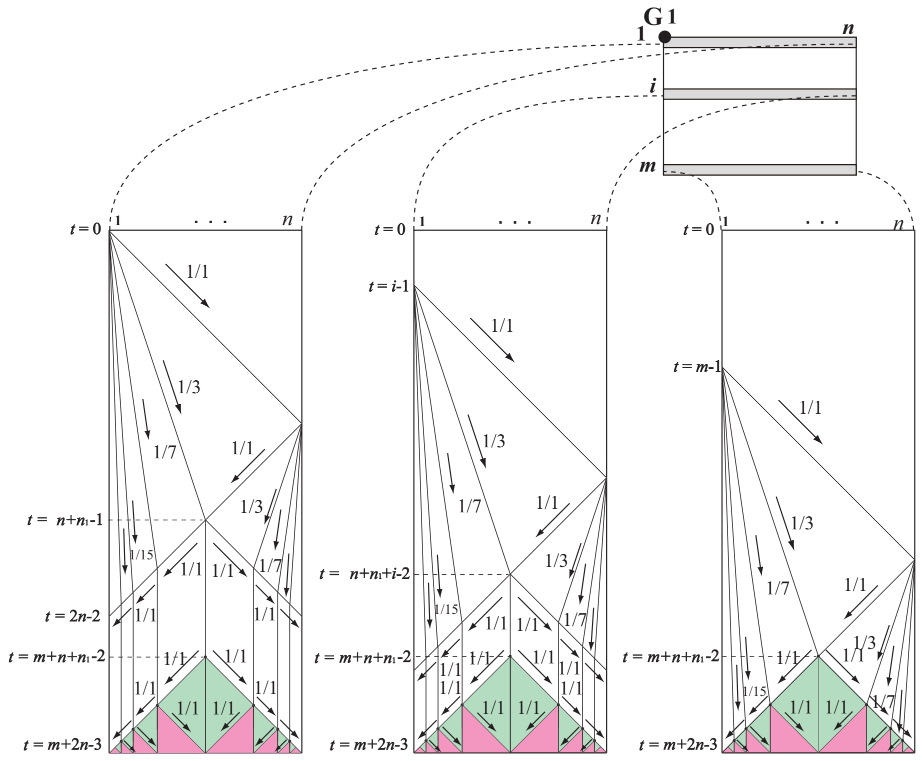

Figure 6: Space-diagrams of the synchronization algorithm on the 1st, ith, and mth rows of a wider-than-long rectangle array of size $m \times n$, respectively.

Note that the signal propagation for the recursive-halving marking and the wake-up signal for the synchronization are made by the same 1/1 speed. Thus, the synchronization can be performed successfully for each column and the array can be synchronized in optimum steps. Figures 5 and 6 illustrate a space-time diagram for the recursive-halving marking and the synchronization operations on the 1st, ith, and mth row of a longer-than-wide and wider-than-long array, respectively. One can see that

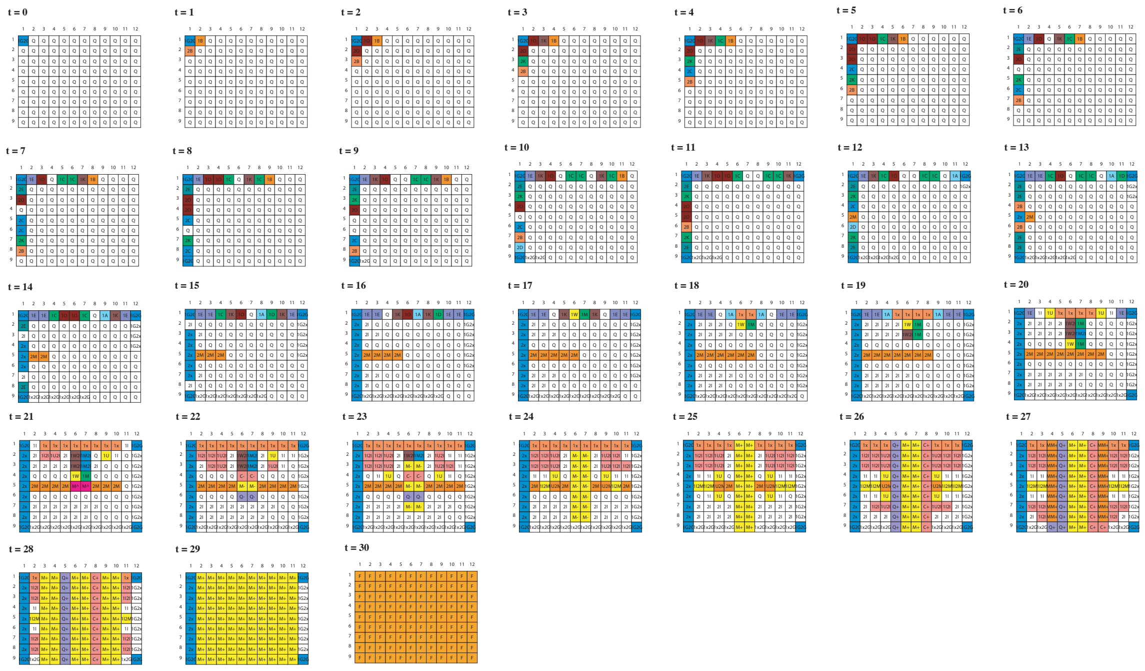

Figure 7: Snapshots for synchronization on a 9×12 array.

each marking operation has been finished before the arrival of the first wake-up signal for the synchronization. The algorithm operates in optimum-steps in a similar way for the rectangles such as case 1: $m = 2m_1 + 1, n = 2n_1$, case 2: $m = 2m_1, n = 2n_1 + 1$, and case 3: $m = 2m_1, n = 2n_1$.

Thus, we can establish the following theorem.

Theorem 5 The synchronization algorithm $\mathscr{A}_1$ can synchronize any $m \times n$ rectangular array in optimum $m + n + \max(m,n) - 3$ steps.

We have implemented the algorithm $\mathscr{A}_1$ on a 2D cellular automaton having 60 states and 13633 local rules. In Figures 7 and 8 we present some snapshots of the synchronization processes of the algorithm on 9×12 and 12×9 arrays, respectively.

5 Expansion to Multi-Dimensional Arrays

5.1 Three-Dimensional Arrays

In this section, we show that there exists no algorithm that can synchronize any 3D array of size $m \times n \times \ell$ with a general at an arbitrary corner in less than $m + n + \ell + max(m,n,\ell) - 4$ steps.

Theorem 6 The minimum time in which the firing squad synchronization could occur is no earlier than $m + n + \ell + max(m,n,\ell) - 4$ for any 3D array of size $m \times n \times \ell$ with a general at an arbitrary corner cell.

Proof. The proof is made by contradiction. Without loss of generality, we assume that $\ell \leq m \leq n$. It is assumed that there is a cellular automaton $\mathscr{M}$ that can synchronize an array of size $m_0 \times n_0 \times \ell_0$ $(\ell_0 \leq m_0 \leq n_0)$ at step $t = t_0$ such that:

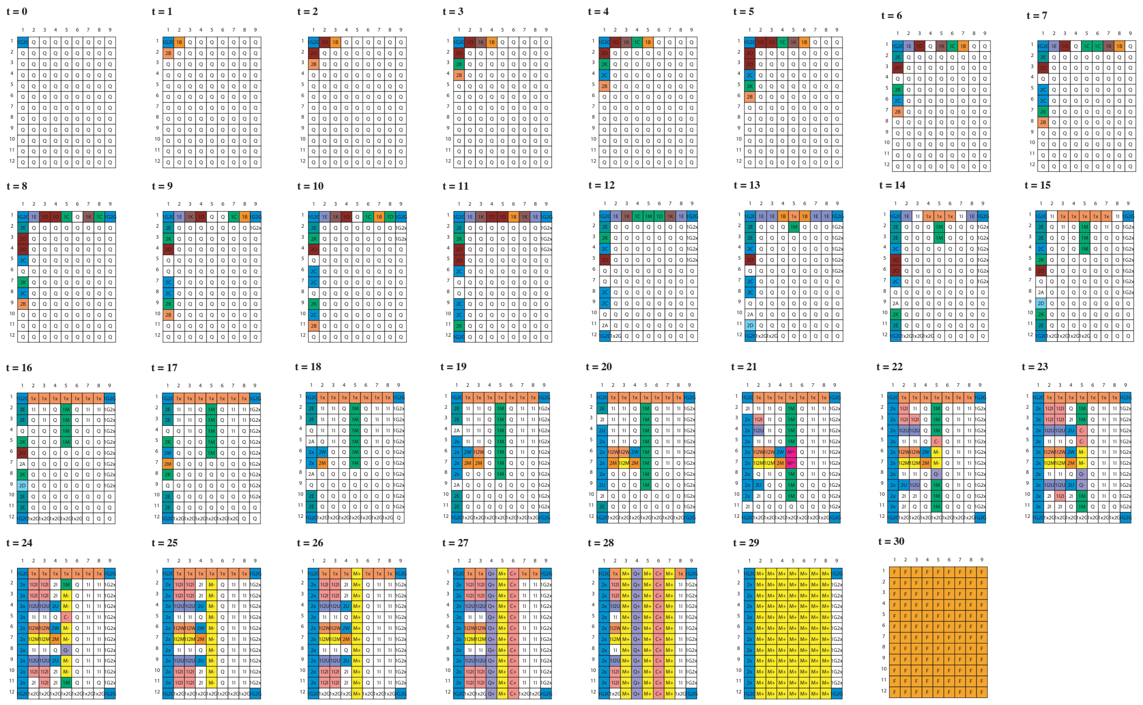

Figure 8: Snapshots for synchronization on a 12×9 array.

$$t_0 < m_0 + 2n_0 + \ell_0 - 4 \tag{4}$$

Now consider the state of cell $C_{m_0\ 1\ \ell_0}$ at time $t = t_0$. Let i and k be any integer such that $1 \le i \le m_0$ and $1 \le k \le \ell_0$. Consider the signal propagation from the cell $C_{1\ 1\ 1}$ to $C_{m_0\ 1\ \ell_0}$ via any cell $C_{i\ n_0\ k}$. It takes:

$$\begin{aligned}(i-1)+(n_0-1)+(k-1)+(m_0-i)+(n_0-1)+(\ell_0-k)\\ = m_0+2n_0+\ell_0-4\end{aligned} \tag{5}$$

steps for the signal to travel from $C_{1\ 1\ 1}$ to $C_{m_0\ 1\ \ell_0}$ via any cell $C_{i\ n_0\ k}$. The state of the cell $C_{m_0\ 1\ \ell_0}$ at step $t = t_0$ entered the final firing state unaffected by any cells on the plane $\{C_{i\ n_0\ k} | 1 \le i \le m_0, 1 \le k \le \ell_0$ $\}$. Therefore, if another three-dimensional array of size $m_0 \times n_0 \times \ell_0$ was added to the right side of the original array (that is, the new array is of size $m_0 \times 2n_0 \times \ell_0$), the cell $C_{m_0\ 1\ \ell_0}$ would still enter the final firing state at step $t = t_0$. This is because the cell structure $\mathscr{M}$ is fixed, cell operation is deterministic and nothing has changed as far as the cell $C_{m_0\ 1\ \ell_0}$ is concerned. Since $t_0 < m_0 + 2n_0 + \ell_0 - 4$, the cell $C_{m_0\ 1\ \ell_0}$ will still be in a quiescent state at time $t = t_0$. Therefore the cell structure does not represent a solution and this is a contradiction. In a similar way, the argument carries over in the cases such as $\ell \le n \le m, n \le \ell \le m$, ..., and so forth.

□

The synchronization algorithm $\mathscr{A}_1$ for 2D arrays can be easily expanded to 3D arrays. See Figure 9 which illustrates the synchronization schema for 3D cellular automaton. Thus, we have:

Theorem 7 There exists an optimum-time synchronization algorithm $\mathscr{A}_2$ that can synchronize any three-dimensional array of size $n_1 \times n_2 \times n_3$ with a general at $C_{1,1,1}$ in optimum $n_1 + n_2 + n_3 + \max(n_1, n_2, n_3) - 4$ steps.

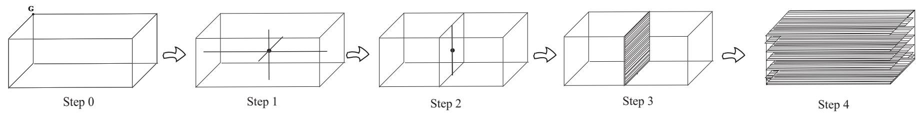

Figure 9: Synchronization scheme for a three-dimensional cellular automaton.

5.2 Multi-Dimensional Arrays

A kD FSSP algorithm $\mathscr{A}_3$ is sketched as follows:

Step 1. Start the recursive-halving marking on cells along each dimension, **find** a *center cell(s) of the given array*, **generate** a general(s) on the center cell(s), and **pre-synchronize** the center point(s): zero-dimensional sub-array of the array.
Step 2. Pre-synchronize a 1D sub-array along the 1st dimension containing the pre-synchronized center cell.
Step 3. - Step k. For $j = 2$ to $k-1$, by increasing the number of dimensions, **pre-synchronize** a jD sub-array containing the pre-synchronized $(j-1)$D sub-array.
Step $k+1$. Synchronize the kD array. This yields the final synchronization of the given array.

Theorems 6 and 7 can be expanded to the kD arrays.

Theorem 9 There exists no cellular automaton that can synchronize any kD array of size $n_1 \times n_2 \times ... \times n_k$ with a general at $C_{1,1,...,1}$ in less than $\sum_{i=1}^{k} n_i + \max(n_1, n_2, ..., n_k) - k - 1$ steps

Theorem 10 There exists an optimum-time synchronization algorithm $\mathscr{A}_3$ that can synchronize any kD array of size $n_1 \times n_2 \times ... \times n_k$ with a general at $C_{1,1,...,1}$ in optimum $\sum_{i=1}^{k} n_i + \max(n_1, n_2, ..., n_k) - k - 1$ steps.

6 Generalization as to General's Position

6.1 Generalized FSSP on 1D Arrays

The recursive-halving marking scheme on 1D array can be easily expanded to the generalized case where the initial general is located at any position of the array. Figure 10 illustrates a space-time diagram for the recursive-halving marking on 1D array of length n with a general on $C_k, 1 \le k \le n$. The marking is based on the generalized FSSP algorithm proposed by Moore and Langdon [1968].

We have seen the following theorems for the 1D generalized case.

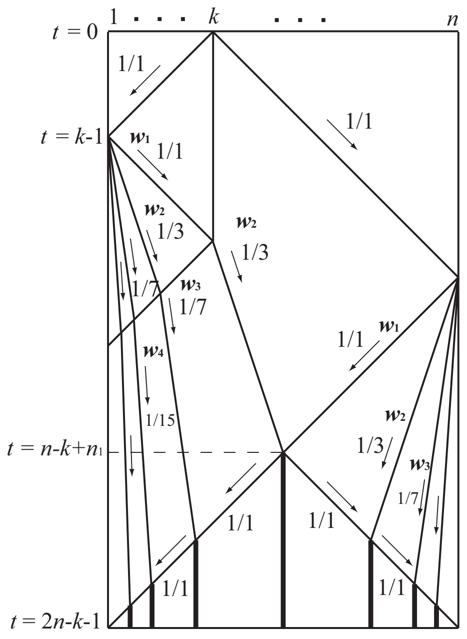

Figure 10: Space-time diagram for recursive-halving marking on 1D array of length n with a general at any position.

Theorem 11 Moore and Langdon [1968] The minimum time in which the generalized firing squad synchronization could occur is $n-2+max(k,n-k+1)$ steps, where the general is located on the kth cell from left end.

Theorem 12 Moore and Langdon [1968] There exists a 17-state cellular automaton that can synchronize any one-dimensional array of length n in optimum $n-2+max(k,n-k+1)$ steps, where the general is located on the kth cell from left end.

An optimum-time complexity $n-2+max(k,n-k+1)$ needed for synchronizing cellular space of length n in the classical ML-type (Moore and Langdon [1968]) generalized FSSP algorithm can be interpreted as follows: Let S be a cellular space of length $n = 2n_1+1$, where $n_1 \geq 1$. The first center mark in S is printed on cell C_{n_1+1} at time $t_{\mathrm{1Dg-center}} = 3n_1 - \min(k-1, n-k)$. Additional n_1 steps are required for the markings thereafter, yielding a final synchronization at time $t_{\mathrm{1Dg-opt}} = 3n_1 - \min(k-1, n-k) + n_1 = 4n_1 - \min(k-1, n-k) = 2n-2-\min(k-1, n-k) = n-2+\max(k, n-k+1)$.

In the case $n = 2n_1$, where $n_1 \geq 1$, the first center mark is printed simultaneously on cells C_{n_1} and C_{n_1+1} at time $t_{\mathrm{1Dg-center}} = 3n_1 - 1 - \min(k-1, n-k)$. Note that two cells C_{n_1} and C_{n_1+1} are pre-synchronized at time $t_{\mathrm{1Dg-center}} = 3n_1 - 1 - \min(k-1, n-k)$. Additional $n_1 - 1$ steps are required for the marking thereafter, yielding the final synchronization at time $t_{\mathrm{1Dg-opt}} = 3n_1 - 1 - \min(k-1, n-k) + n_1 - 1 = 4n_1 - 2 - \min(k-1, n-k) = 2n-2-\min(k-1, n-k) = n-2+\max(k, n-k+1)$.

$$t_{\mathrm{1Dg-center}} = \begin{cases} 3n_1 - \min(k-1, n-k) & |S| = 2n_1+1, \\ 3n_1 - 1 - \min(k-1, n-k) & |S| = 2n_1. \end{cases} \tag{6}$$

In addition, $t_{\mathrm{1Dg-sync}}$ steps are required for the synchronization for a cellular space with the recursive-halving marks:

$$t_{1\mathrm{Dg-sync}} = \begin{cases} n_1 & |S| = 2n_1 + 1, \\ n_1 - 1 & |S| = 2n_1. \end{cases} \tag{7}$$

6.2 Generalized FSSP on 2D Arrays

By a similar method employed in Section 4, we can develop the following theorem for the generalized case.

Theorem 13 There exists no 2D cellular automaton that can synchronize any 2D array of size $m \times n$ with an initial general on $\mathrm{C}_{r,s}$ in less than $m+n+\max(m,n)-\min(r,m-r+1)-\min(s,n-s+1)-1$ steps, where $1 \le r \le m, 1 \le s \le n$.

Now we are going to present a generalized optimum-time FSSP Algorithm $\mathscr{A}_4$ for 2D arrays. We assume that an initial general G is on the cell $\mathrm{C}_{r,s}$ of a given array of size $m \times n$, where $1 \le r \le m, 1 \le s \le n$. The algorithm consists of three phases: a marking phase, a pre-synchronization phase and a final synchronization phase. An overview of the 2-D synchronization algorithm $\mathscr{A}_4$ is as follows:

Step 1. Start the recursive-halving marking for cells on each row and column, **find** a *center cell(s) of the given array*, and **generate** a new general(s) on the center cell(s). Note that a crossing(s) of the center column(s) with the center row(s) is a center cell(s) of the array.
Step 2. Pre-synchronize the center column(s) using Lemma 4, which is initiated by the general in step 1. Every cell on the center column(s) acts as a general at the next Step 3.
Step 3. Synchronize each row using Lemma 4, initiated by the general generated in Step 2. This yields the final synchronization of the array.

The array can be synchronized at time $t = m+n+\max(m,n)-\min(r,m-r+1)-\min(s,n-s+1)-1$ in optimum-steps.

Theorem 14 There exists an optimum-time synchronization algorithm $\mathscr{A}_4$ that can synchronize any $m \times n$ rectangular array with a general at $\mathrm{C}_{r,s}$ in optimum $m+n+\max(m,n)-\min(r,m-r+1)-\min(s,n-s+1)-1$ steps, where $1 \le r \le m, 1 \le s \le n$.

We have implemented the algorithm $\mathscr{A}_4$ on a 2D cellular automaton having 269 states and 163662 local rules. We have checked the rule set for any array of size $m \times n$, with $2 \le m,n \le 100$, and any general's position in the array. In Figure 11 we present some snapshots of the synchronization processes of the algorithm on an 8×13 array with a general on $\mathrm{C}_{3,5}$.

6.3 Generalized FSSP on Multi-Dimensional Arrays

Theorems 13 and 14 can be expanded to three or more dimensional arrays.

Theorem 15 The minimum time in which the firing squad synchronization could occur is no earlier than $n_1+n_2+n_3+\max(n_1,n_2,n_3)-\min(r_1,n_1-r_1+1)-\min(r_2,n_2-r_2+1)-\min(r_3,n_3-r_3+1)-1$ for any 3D array of size $n_1 \times n_2 \times n_3$ with a general at C_{r_1,r_2,r_3}.

Theorem 16 There exists an optimum-time synchronization algorithm $\mathscr{A}_5$ that can synchronize any 3D array of size $n_1 \times n_2 \times n_3$ with a general at C_{r_1,r_2,r_3} in optimum $n_1+n_2+n_3+\max(n_1,n_2,n_3)-\min(r_1,n_1-r_1+1)-\min(r_2,n_2-r_2+1)-\min(r_3,n_3-r_3+1)-1$ steps.

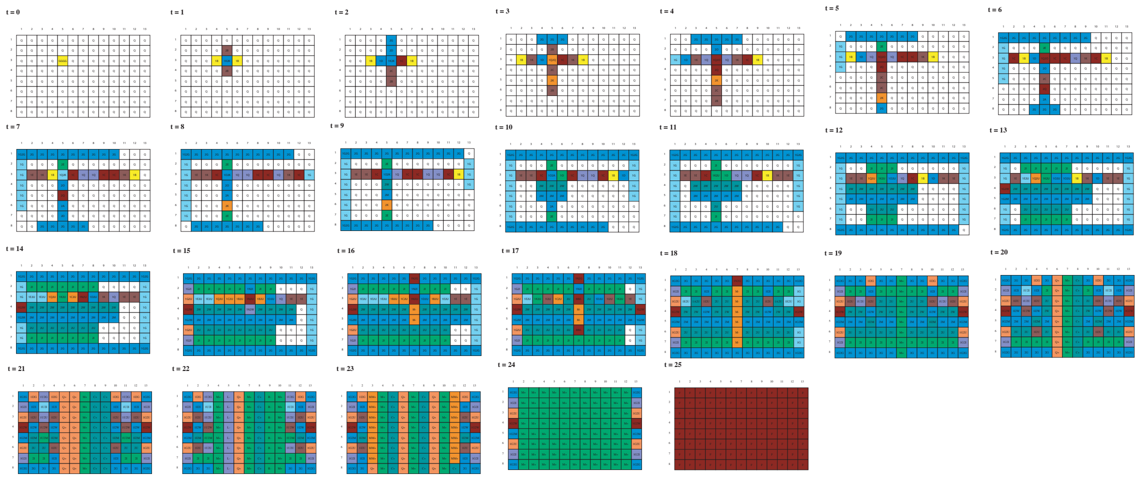

Figure 11: Snapshots of the generalized synchronization algorithm $\mathscr{A}_4$ on an 8×13 array with a general on $C_{3,5}$.

Theorem 17 There exists no cellular automaton that can synchronize any kD array of size $n_1 \times n_2 \times ... \times n_k$ with a general at $C_{r_1,r_2,...,r_k}$ in less than $\sum_{i=1}^{k} n_i + \max(n_1, n_2, ..., n_k) - \sum_{i=1}^{k} \min(r_i, n_i - r_i + 1) - 1$ steps.

Theorem 18 There exists an optimum-time synchronization algorithm $\mathscr{A}_6$ that can synchronize any kD array of size $n_1 \times n_2 \times ... \times n_k$ with a general at $C_{r_1,r_2,...,r_k}$ in optimum $\sum_{i=1}^{k} n_i + \max(n_1, n_2, ..., n_k) - \sum_{i=1}^{k} \min(r_i, n_i - r_i + 1) - 1$ steps.

7 Conclusions

We have proposed a new class of optimum-time multi-dimensional FSSP algorithms based on recursive-halving marking. The class includes the well-known optimum-time FSSP algorithms developed by Waksman [1964], Balzer [1966] and Gerken [1987] with a general at one end and Moore and Langdon [1968] with a general at any position.

Acknowledgments The authors would like to thank reviewers for helpful comments.

Bibliography

[1] R. Balzer: An 8-state minimal time solution to the firing squad synchronization problem. *Information and Control*, vol. 10 (1967), pp. 22-42, doi:10.1016/S0019-9958(67)90032-0.

[2] W. T. Beyer: Recognition of topological invariants by iterative arrays. Ph.D. Thesis, MIT, (1969), pp. 144.

[3] H. D. Gerken. (1987): Über Synchronisationsprobleme bei Zellularautomaten. *Diplomarbeit*, Institut für Theoretische Informatik, Technische Universität Braunschweig, pp. 50.

[4] E. Goto: A minimal time solution of the firing squad problem. Dittoed course notes for Applied Mathematics 298, Harvard University, (1962), pp. 52-59.

[5] A. Grasselli: Synchronization of cellular arrays: The firing squad problem in two dimensions. *Information and Control*, vol. 28(1975), pp. 113-124, doi:10.1016/S0019-9958(75)90255-7.

[6] J. Mazoyer: A six-state minimal time solution to the firing squad synchronization problem. *Theoretical Computer Science*, vol. 50 (1987), pp. 183-238, doi:10.1016/0304-3975(87)90124-1.

[7] E. F. Moore: The firing squad synchronization problem. in *Sequential Machines, Selected Papers* (E. F. Moore, ed.), Addison-Wesley, Reading MA.,(1964), pp. 213-214.

[8] F. R. Moore and G. G. Langdon (1968): A generalized firing squad problem. *Information and Control*, 12, pp. 212-220, doi:10.1016/S0019-9958(68)90309-4.

[9] H. Schmid: Synchronisationsprobleme für zelluläre Automaten mit mehreren Generälen. Diplomarbeit, Universität Karsruhe, (2003).

[10] H. Schmid and T. Worsch: The firing squad synchronization problem with many generals for one-dimensional CA. *Proc. of IFIP World Congress*, pp.111-124, (2004).

[11] I. Shinahr: Two- and three-dimensional firing squad synchronization problems. *Information and Control*, vol. 24(1974), pp. 163-180, doi:10.1016/S0019-9958(74)80055-0.

[12] H. Szwerinski: Time-optimum solution of the firing-squad-synchronization-problem for *n*-dimensional rectangles with the general at an arbitrary position. *Theoretical Computer Science*, vol. 19(1982), pp. 305-320, doi:10.1016/0304-3975(82)90040-8.

[13] H. Umeo: Firing squad synchronization problem in cellular automata. In *Encyclopedia of Complexity and System Science*, R. A. Meyers (Ed.), Springer, Vol.4(2009), pp.3537-3574, doi:10.1007/978-0-387-30440-3_211.

[14] H. Umeo: Recent developments in firing squad synchronization algorithms for two-dimensional cellular automata and their state-efficient implementations. *Proc. of the 13th Intern. Conference on Automata and Formal Languages, AFL 2011*, pp.368-387, (2011).

[15] H. Umeo, M. Hisaoka, and S. Akiguchi: Twelve-state optimum-time synchronization algorithm for two-dimensional rectangular cellular arrays. *Proc. of 4th International Conference on Unconventional Computing: UC 2005, LNCS 3699*, (2005), pp.214-223, doi:10.1007/11560319_20.

[16] H. Umeo, M. Hisaoka and T. Sogabe: A survey on optimum-time firing squad synchronization algorithms for one-dimensional cellular automata. *Intern. J. of Unconventional Computing*, 2005, vol. 1, pp.403-426.

[17] H. Umeo, M. Hisaoka, M. Teraoka and M. Maeda: Several new generalized linear- and optimum-time synchronization algorithms for two-dimensional rectangular arrays. *Proc. of 4th International Conference on Machines, Computations and Universality : MCU 2004, LNCS 3354* (M. Margenstern (Ed.), (2005), pp.223-232, doi:10.1007/978-3-540-31834-7_18.

[18] H. Umeo and K. Kubo: Recent developments in constructing square synchronizers. *Proc. of the 10th International Conference on Cellular Automata for Research and Industry*, (2012, to appear).

[19] H. Umeo, M. Maeda, M. Hisaoka and M. Teraoka: A state-efficient mapping scheme for designing two-dimensional firing squad synchronization algorithms. *Fundamenta Informaticae*, Vol.74(2006), pp.603-623.

[20] H. Umeo, K. Nishide, and T. Yamawaki: A new optimum-time firing squad synchronization algorithm for two-dimensional rectangle arrays - one-sided recursive halving based. *Proc. of the Intern. Conf. on Models of Computation in Context, Computability in Europe 2011, CiE 2011, LNCS 6735*, (B. Löwe et al. (Eds.)), pp. 290-299, (2011).

[21] H. Umeo and H. Uchino: A new time-optimum synchronization algorithm for rectangle arrays. *Fundamenta Informaticae*, Vol.87, No.2, pp.155-164(2008).

[22] H. Umeo, H. Uchino and A. Nomura: How to synchronize square arrays in optimum-time. *Proc. of the 2011 International Conference on High Performance Computing and Simulation (HPCS 2011)*, pp. 801-807, IEEE, (2011).

[23] A. Waksman: An optimum solution to the firing squad synchronization problem. *Information and Control*, vol. 9 (1966), pp. 66-78, doi:10.1016/S0019-9958(66)90110-0.