The 23rd International Technical Conference on Circuits/Systems,
Computers and Communications (ITC-CSCC 2008)

A Note on Three-Dimensional Firing Squad Synchronization Algorithm

Takuya Yamawaki[1], Takashi Amesara[1], and Hiroshi Umeo[1]
[1]Faculty of Information Science and Technology, Osaka Electro-Communication Univ.,
Neyagawa-shi, Hatsu-cho 18-8, Osaka, 572-8530, Japan
Tel.: +81-72-824-1131, Fax.: +81-72-824-1140
E-mail: {yamawaki, amesara, umeo}@cyt.osakac.ac.jp
Keywords: Cellular automaton, Synchronization algorithm, Three-dimensional cellular automaton, Firing squad synchronization problem

Abstract: The firing squad synchronization problem on cellular automata has been studied extensively for more than forty years, and a rich variety of synchronization algorithms have been proposed [1-10]. In the present paper, we propose an optimum-time firing squad synchronization algorithm for three-dimensional cellular automata. The algorithm can synchronize any three-dimensional array of size $m \times n \times \ell$ with a general at an arbitrary corner cell at exactly $m + n + \ell + max(m, n, \ell) - 4$ steps. A lower bound in time complexity is also given.

1. Introduction

We study a synchronization problem that gives a finite-state protocol for synchronizing a large scale of 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 40 years [1-10]. In this paper, we propose an optimum-time firing squad synchronization algorithm for three-dimensional cellular automata. The algorithm can synchronize any three-dimensional array of size $m \times n \times \ell$ with a general at a corner cell at exactly $m + n + \ell + max(m, n, \ell) - 4$ steps.

2. Firing Squad Synchronization Problem

2.1 FSSP on One-Dimensional Cellular Arrays

The firing squad synchronization problem (FSSP) is formalized in terms of the model of cellular automata. All cells (except the end cells) are identical finite state automata. The array operates in lock-step mode such that the next state of each cell (except the end cells) is determined by both its own present state and the present states of its right and left neighbors. All cells (*soldiers*), except the left end cell, are initially in the *quiescent* state at time $t = 0$ and have the property whereby the next state of a quiescent cell having quiescent neighbors is the quiescent state. At time $t = 0$ the left end cell (*general*) is in the *fire-when-ready* state, which is an initiation signal to the array. The firing squad synchronization problem is stated as follows: Given an array of n identical cellular automata, including a *general* on the left end which is activated at time $t = 0$, we want to give the description (state set and next-state transition function) of the automata so that, *at some future time*, all of the cells will *simultaneously* and, *for the first time*, enter a special *firing* state. The set of states and the next-state transition function must be independent of n. The problem was first solved by J. McCarthy and M. Minsky who presented a $3n$-step algorithm. In 1962, the first optimum-time, i.e. $(2n - 2)$-step, synchronization algorithm for 1-D arrays of length n 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 a six-state synchronization algorithm which, at present, is the algorithm having the fewest states.

2.2 FSSP on Three-Dimensional Cellular Arrays

Figure 1 shows a finite three-dimensional cellular array consisting of $m \times n \times \ell$ 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, west, lower and upper 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 upper corner cell C_{111} 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 $m \times n \times \ell$ of the array. The set of states and transition rules must be independent of m, n and ℓ.

Several synchronization algorithms on 2-D arrays have been proposed by Beyer [1969], Grasselli [1975], Shinahr [1974], Szwerinski [1982] and Umeo, Maeda, Hisaoka and Teraoka [2006]. As for the three-dimensional synchronization problem, Shinahr [1974] has shown that any cube of size $n \times n \times n$ with a general at an arbitrary corner cell can be synchronized in optimum $3n - 3$ steps. Up to now no algorithm has been proposed for more general 3-D array.

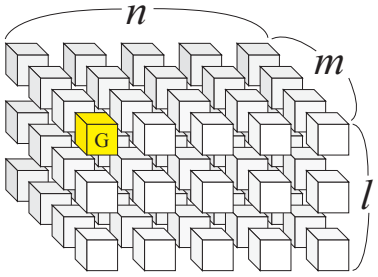

Figure 1. A three-dimensional cellular automaton.

2.3 Lower Bound in Time Complexity

In this subsection, we show that there exists no algorithm that can synchronize any 3-D 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 1] 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 three-dimensional 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 $\mathcal{M}$ 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_0 such that:

$$t_0 < m_0 + 2n_0 + \ell_0 - 4 \quad (1)$$

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 $1 \leq i \leq m_0$ and $1 \leq k \leq \ell_0$. Consider the signal propagation from the cell $C_{1\ 1\ 1}$ to $C_{m_0\ 1\ \ell_0}$ via an any cell $C_{i\ n_0\ k}$. It takes:

$$(i-1) + (n_0-1) + (k-1) + (m_0-i) + (n_0-1) + (\ell_0-k) \\ = m_0 + 2n_0 + \ell_0 - 4 \quad (2)$$

steps for the signal to travel from $C_{1\ 1\ 1}$ to $C_{m_0\ 1\ \ell_0}$ via an 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 $\{C_{i\ n_0\ k} | 1 \leq i \leq m_0, 1 \leq k \leq \ell_0\ \}$. 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 $\mathcal{M}$ 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 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 $\ell \leq n \leq m, n \leq \ell \leq m$ and so forth.

3. Optimum-time Synchronization Algorithm for Three-Dimensional Arrays

3.1 Delayed Firing Squad Synchronization Algorithm One-Dimensional Arrays

We introduce a *freezing-thawing* technique that yields a delayed synchronization algorithm for one-dimensional arrays. The technique is very useful in the design of time-efficient synchronization algorithms for one- and two-dimensional arrays in Umeo [2004], Yunès [2006] and Umeo and Uchino [2006]. A similar technique was used by Romani [1977] in the tree synchronization. The technique is stated as in the following theorem.

[Theorem 2][Umeo [2004]] Let t_0, t_1, t_2 and Δt be any integer such that $t_0 \geq 0$, $t_0 \leq t_1 \leq n-1$, $t_1 \leq t_2$ and $\Delta t = t_2 - t_1$. We assume that a usual optimum-time synchronization operation is started at time $t = t_0$ by generating a special signal at the left end of one-dimensional array and the right end cell of the array receives another special signals from outside at time $t = t_1$ and t_2, respectively. Then, there exists a one-dimensional cellular automaton that can synchronize the array of length n at time $t = t_0 + 2n - 2 + \Delta t$.

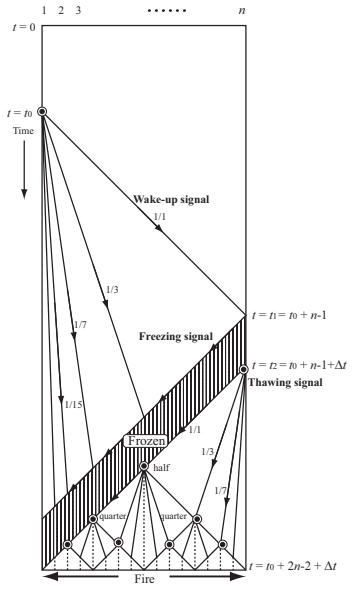

Figure 2. Time-space diagram for delayed firing squad synchronization scheme based on the *freezing-thawing* technique.

The array operates as follows:

1. Start an optimum-time firing squad synchronization algorithm at time $t = t_0$ at the left end of the array. A 1/1-speed signal is propagated towards the right direction to wake-up cells in quiescent state. We refer the signal as *wake-up signal*. A *freezing* signal is given from outside at time $t = t_1$ at the right end of the array. The signal is propagated in the left direction at its maximum speed, that is, 1 cell per 1 step, and freezes the configuration progressively. Any cell that receives the freezing signal from its right neighbor has to stop its state-change and transmits the freezing signal to its left neighbor. The frozen cell keeps its state as long as no thawing signal will arrive.
2. A special signal supplied with outside at time $t = t_2$ is used as a *thawing* signal that thaws the frozen configuration. The thawing signal forces the frozen cell to resume its state-change procedures immediately. See Fig. 2. The signal is also transmitted toward the left end at speed 1/1.

The readers can see how those three signals work. We can freeze the entire configuration during Δt steps and delay the synchronization on the array for Δt steps. It is easily seen that the freezing signal can be replaced by the reflected signal of the wake-up signal, that is generated at the right end cell at time $t = t_0 + n - 1$. See Fig. 2. We refer the scheme as *freezing-thawing* technique.

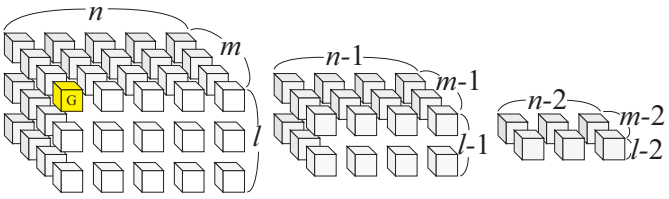

Figure 3. Recurrent decomposition.

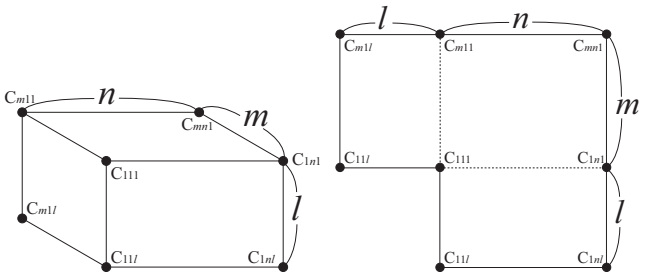

Figure 4. Layer development.

3.2 An Optimum-time Synchronization Algorithm for Three-dimensional Arrays

In this section we propose an optimum-time algorithm that can synchronize any three-dimensional array of size $m \times n \times \ell$ with a general on an arbitrary corner cell at exactly $m+n+\ell+max(m,n,\ell)-4$ steps.

[Algorithm] Here we give an overview of the algorithm. Without loss of generality, we assume that $\ell \leq m \leq n$. In our design we regard a three-dimensional array as consisting of many two-dimensional layers. The decomposition is as follow: A three-dimensional array is decomposed into $\min(m,n,\ell)$ 2-D layers. Figure 3 shows that a three dimension array is decomposed into those

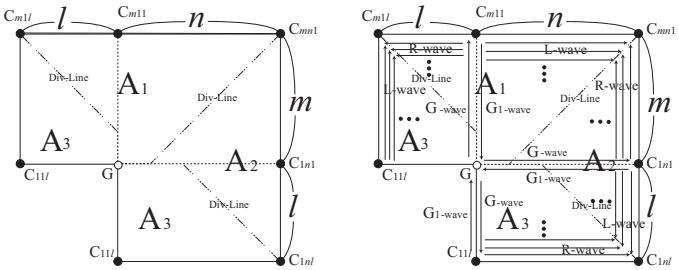

Figure 5. Decomposition of the first layer.

layers. Figure 4 shows a two-dimensional expansion of the layer. The layer is also decomposed into three areas A_1, A_2, and A_3. Figure 5 shows a decomposition of the first layer. We design the algorithm such that those synchronizing operations in first layer occur at exactly $m+n+\ell+max(m,n,\ell)-4$ steps. The same algorithm can be applied to the second layer. It can synchronize at exactly $t=(m-1)+(n-1)+(\ell-1)+max((m-1),(n-1),(\ell-1))-4$ steps. It is noted that their synchronizing time for the first and the second layer differs by 4 steps. The same argument holds for all other neighboring layers. Thus, the initial *general* is scheduled to be generated at every 4 steps in the layer. Each area consists of many one-dimensional arrays. A *sub-general* on a one-dimensional array is generated by the signal from the *general* in the layer and starts the synchronization algorithm on each one-dimensional array. The *sub-general* is the state of the general in one-dimensional cell array. The synchronization operation on area A_1 of first layer is as follow: First, the *general* G in the layer sends a 1/1-speed signal at time $t=0$. Secondly, C_{m11} receives the 1/1-speed signal, generates *sub-general* for the one-dimensional array, and the 1/3-speed signal returns. Similarly, each cell which has received the 1/3-speed signal generates the *sub-general*. The 1/3-speed signal changes its propagation speed on the way. See Figures 6 and 7. Figure 6 shows the location where the sub-generals are generated and the changes for the propagation speed of the signals are made.

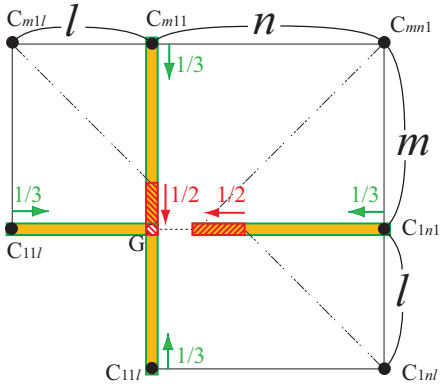

Figure 6. Location of sub-generals and signal speed.

Eventually, the array starts an optimum-time firing squad synchronization algorithm on one-dimensional array. As for the area A_2 and A_3, similar discussions can be made. However, the 1/3-speed signal dose not change its speed only in the area A_3. Because, *sub-general* is never located on the end of the one-dimensional cell arrays unlike other areas. Moreover, it is necessary to stop the *sub-general* generation signal in area A_2 once. It should be noted that in the area A_2 the signal arrives earlier than other areas. For the purpose of simultaneous synchronizations the *freezing-thawing* technique given in [Theorem 2] is used, since the delay is necessary to synchronize each one-dimensional array. The delayed time is

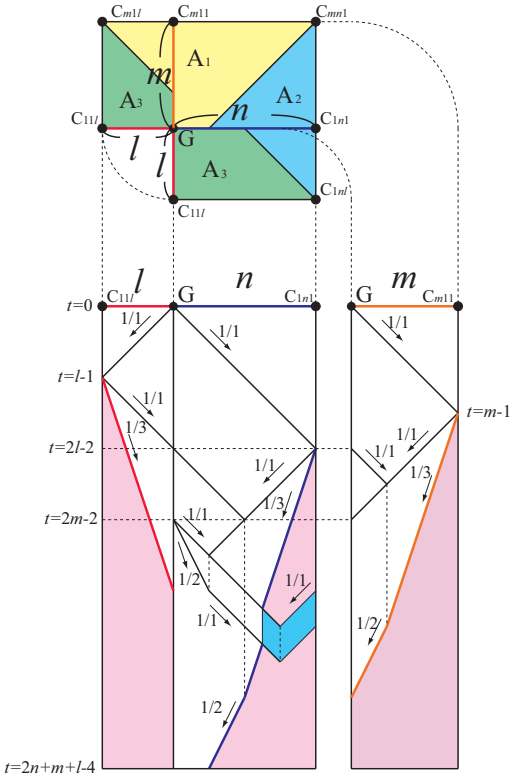

Figure 7. Generation of sub-generals for each synchronization.

equal to steps in difference of length, that is, two edges except the shortest edge of three (m,n,ℓ). That is, it is necessary to delay $n-m$ steps, since we assume that $\ell \le m \le n$.

Figure 8 shows a freezing-thawing signal of the first layer. The *general* G in the layer sends three signals for C_{mm1}, $C_{1\ell\ell}$ and $C_{\ell 1\ell}$ in addition to generate the sub-general signals at time $t=0$. Sending a 1/2-speed signal when those signals collide with the 1/1-speed signal from the *sub-general* on most distant one-dimensional cell array in each area. If the signal doesn't collide, the 1/1-speed signal turns when it reaches the end of one-dimensional cell array. The cell that collides mutually the turned signal becomes the starting point cell of the freezing signal. When the 1/2-speed signal reaches the end of one-dimensional cell array, it spreads to other areas and it becomes a 1/1-speed signal. This signal reaches the starting point cell of the freezing signal, the thawing signal is transmitted. The delay is generated by such a method.

Figure 9 is a time-space diagram of one-dimensional firing squad synchronization algorithm. It illustrates a timing scheme of firing squad synchronization algorithm for the ith, jth and kth one-dimensional cell arrays from G on each area. It is time when beginning of the freezing signal of the jth cell array in area A_2 reaches a 1/1-speed signal from starting point cell the freezing signal in the most distant from G on the nth cell array. Here, the time that the sub-general is generated is indicated as follows:

- Area A_1
  $t=m-1+3(m-i), \ell-1 \le i \le m,$
  $t=m-1+2(m-i), 0 \le i < \ell-1,$
- Area A_2
  $t=n-1+3(n-j), n-m+\lceil \frac{m}{2} \rceil \le j \le n$
  $t=n-1+3(n-j)+(n-m), m-\ell \le j < n-m+\lceil \frac{m}{2} \rceil$
  $t=n-1+2(n-j), n-m \le j < m-\ell,$
- Area A_3
  $t=\ell-1+3(\ell-k), 0 \le k \le \ell.$

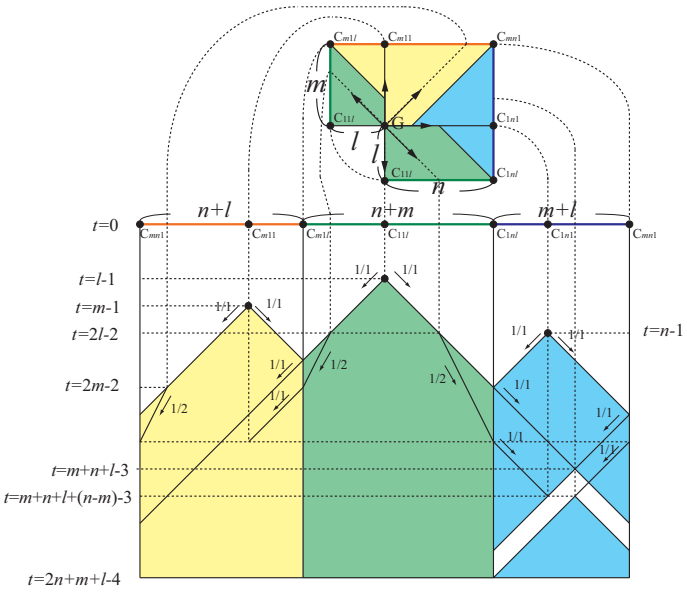

Figure 8. Freezing-thawing signal on first layer.

Thus we have:

[Theorem 3] There exists a three-dimensional cellular automaton that can synchronize any three-dimensional array of size $m \times n \times \ell$ with a general on an arbitrary corner cell at exactly $m + n + \ell + max(m, n, \ell) - 4$ steps.

4. Discussions

It has been shown that there exists no algorithm that can synchronize any three-dimensional 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. We also have given an overview on an optimum-time firing squad synchronization algorithm for three-dimensional cellular automata. We need a further study for its real realizations in terms of internal states and transition rules in a three-dimensional cellular automaton.

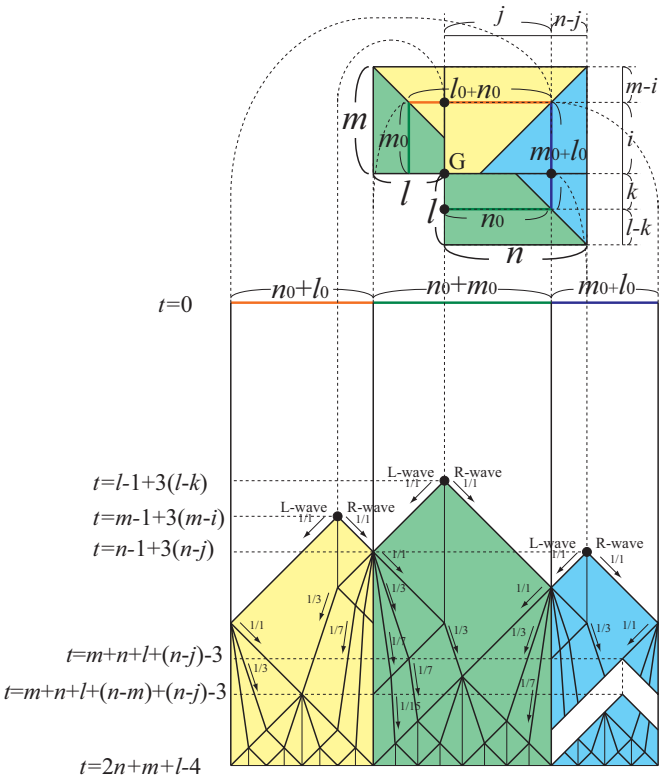

Figure 9. Time-space diagram.

References

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

[2] Hans-D. Gerken: Über Synchronisations - Probleme bei Zellularautomaten. *Diplomarbeit*, Institut für Theoretische Informatik, Technische Universität Braunschweig, (1987) pp. 50.

[3] J. Mazoyer. (1997): A minimal-time solution to the FSSP without recursive call to itself and with bounded slope of signals. Draft version, pp. 8.

[4] M. Minsky. (1967): *Computation: Finite and infinite machines*. Prentice Hall, pp. 28-29.

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

[6] I. Shinahr: Two- and three-dimensional firing squad synchronization problems. *Information and Control*, vol. 24(1974), pp. 163-180

[7] 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.

[8] H. Umeo: A simple design of time-efficient firing squad synchronization algorithms with fault-tolerance. *IEICE Trans. on Information and Systems*, Vol. E87-D, No.3, 2004, pp.733-739(2004).

[9] H. Umeo, M. Maeda and N. Fujiwara: An efficient mapping scheme for embedding any one-dimensional firing squad synchronization algorithm onto two-dimensional arrays. *Proc. of the 5th International Conference on Cellular Automata for Research and Industry*, LNCS 2493, Springer-Verlag, pp.69-81(2002).

[10] A. Waksman. (1966): An optimum solution to the firing squad synchronization problem. *Information and Control*, vol. 9 (1966), pp. 66-78.