An Investigation into State-Change Complexities in Synchronization Algorithms for Cellular Automata

Kouhei Matsumoto
Univ. of Osaka Electro-Communication
Neyagawa-shi, Hatsu-cho 18-8
Osaka, 572-8530, Japan

Hiroshi Umeo
Univ. of Osaka Electro-Communication
Neyagawa-shi, Hatsu-cho 18-8
Osaka, 572-8530, Japan

Abstract

This paper presents a comparative study of state-change complexity in optimum- and linear-time synchronization algorithms for a large-scale of cellular automata on one- and two-dimensional arrays. We implement most of the synchronization protocols developed so far and investigate their state-change complexities on a computer.

1 Introduction

In recent years cellular automata (CA) have been establishing increasing interests in the study of modeling real phenomena occurring in biology, chemistry, ecology, economy, geology, mechanical engineering, medicine, physics, sociology, public traffic, etc. Cellular automata are considered to be a nice model of complex systems in which an infinite one-dimensional array of finite state machines (cells) updates itself in synchronous manner according to a uniform local rule.

The synchronization in cellular automata has been known as firing squad synchronization problem since its development, in which it was originally proposed by J. Myhill to synchronize all parts of self-reproducing cellular automata [9]. The firing squad synchronization problem has been studied extensively for more than 40 years [1-18].

The complexity of synchronization algorithms has received much attention, and those synchronization algorithms have been studied from viewpoints of time complexity, number of internal states of the automata, and number of transition rules. Vollmar [15, 16] introduced a concept of state-change complexity as a measure of energy consumption in the cellular spaces and showed that any optimum-time synchronization algorithms operating on one-dimensional array need $\Omega(n \log n)$ state-changes for synchronizing n-cells.

In this paper, we give a comparative study of state-change complexity in the synchronization algorithms developed so far. The synchronization algorithms being compared in this paper are seven optimum- and linear-time algorithms developed by Balzer [1], Fischer [2], Gerken [3], Mazoyer [8], Minsky [9], Waksman [17] and Yunés [18] . The state-change complexity in several synchronization algorithms on two-dimensional arrays are also studied in the last section.

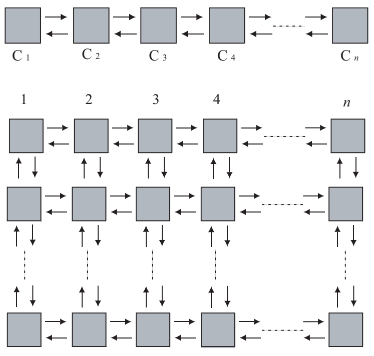

Figure 1: Cellular automata.

2 Firing squad synchronization problem on one-dimensional cellular automaton

In this section we give informal definitions of cellular automata, firing squad synchronization problem and a measure of state-change complexity for synchronization algorithms on one-dimensional cellular automata.

2.1 One-dimensional cellular automaton

Figure 1 (above) shows a finite one-dimensional cellular array consisting of n cells, denoted by C_i, where $1 \leq i \leq n$. Each cell is an identical (except the border cells) finite-state automaton. The array operates in lock-step mode in such away that the next state of each cell (except border cells) is determined by both its own present state and the present states of its left and right neighbors. All cells, except the left end cell, are initially in a *quiescent* state at time $t = 0$. The quiescent state has a property that the next state of a quiescent cell with quiescent neighbors is the quiescent state again.

2.2 Firing Squad Synchronization Problem

The firing squad synchronization problem is formalized in terms of the model of cellular automata. All cells (*soldiers*), except the left end cell, are initially in the *quiescent* state at time $t = 0$. 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 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 function must be independent of n. Without loss of generality, we assume that $n \geq 2$. 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 length n of the array.

2.3 State-change complexity in one-dimensional synchronization algorithms

Vollmar [15, 16] introduced a state-change complexity in order to measure the efficiency of cellular algorithms and showed that $\Omega(n \log n)$ state changes are required for the synchronization of n cells in $(2n - 2)$ steps.

[Theorem 1][16] $\Omega(n \log n)$ state-change is necessary for synchronizing n cells in $(2n - 2)$ steps.

3 State-change complexity in one-dimensional synchronization algorithms

3.1 Optimum-time synchronization algorithms

Waksman [17] presented a 16-state optimum-time synchronization algorithm. Afterward, Balzer [1] and Gerken [3] developed an eight-state algorithm and a seven-state synchronization algorithm: Gerken I, respectively, thus decreasing the number of states required for the synchronization. In 1987, Mazoyer [8] developed a six-state synchronization algorithm which, at present, is the algorithm having the fewest states. The state change complexity for those algorithms is investigated on a computer, and the next theorem is established.

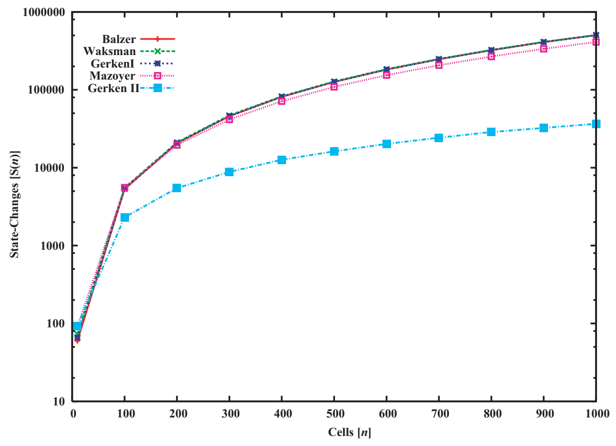

Figure 2: State-changes in optimum-time synchronization algorithms.

[Theorem 2] Each optimum-time synchronization algorithm developed by Balzer [1], Gerken [3], Mazoyer [8] and Waksman [17] has an $O(n^2)$ state-change complexity, respectively.

Figure 2 shows a comparison between state-changes of the optimum-time synchronization algorithms. Gerken [3] has shown that his 155-state algorithm has $\Theta(n \log n)$ state-change complexity.

3.2 Linear-time synchronization algorithms

The firing squad synchronization problem was first solved by J. McCarthy and M. Minsky [9] who presented a $3n$-step algorithm. Fischer [2] gave a 15-state implementation for the $3n$-step synchronization scheme. Both of the algorithms were based on the divide-and-conquer scheme that were realized with 1/1- and 1/3-speed signals interacting each other. These signals look like threads in the time-space diagram and the $3n$-step synchronization algorithm based on the scheme is said to be *thread-like* algorithm. Yunés [18] presented a seven-state implementation based on the thread-like algorithm. In his construction, the width of the threads is larger than the previous design. We can get the state change complexity for the $3n$-step thread-like synchronization algorithms with finite-width threads.

[Theorem 3] Any $3n$-step finite-width thread-like synchronization algorithm has an $O(n \log n)$ state-change complexity.

[Theorem 4] Each linear-time $3n$-step synchronization algorithm developed by Fischer [2], Minsky and MacCarthy [9], and Yunés [18] has an $\Omega(n \log n)$ state-change complexity, respectively.

Figure 3 shows a comparison between state-changes of the $3n$-step synchronization algorithms with finite-width threads.

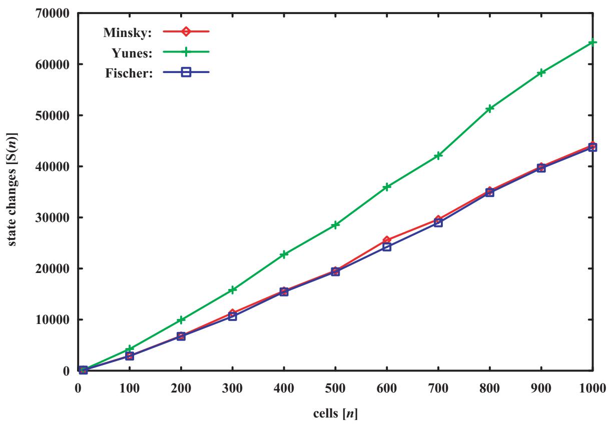

Figure 3: State-changes in $3n$-step synchronization algorithms.

4 State-change complexity in two-dimensional synchronization algorithms

Figure 1 (below) shows a finite two-dimensional (2-D) cellular array 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 $C_{1,1}$ is in the *fire-when-ready* state, which is the initiation signal for synchronizing 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 set of states must be independent of m and n. Maeda and Umeo [5] developed a simple and efficient mapping scheme that enables us to embed any one-dimensional firing squad synchronization algorithm onto two-dimensional arrays without introducing additional states. We see that any configuration on a 1-D CA consisting of $m + n - 1$ cells can be mapped onto 2-D $m \times n$ arrays. Therefore, when the embedded 1-D CA fires $m+n-1$ cells in $T(m+n-1)$ steps, the corresponding 2-D CA fires the $m \times n$ array in $T(m+n-1)$ steps. We can design state-efficient synchronization algorithms based on the mapping. Those algorithms are stated as follows:

[Theorem 5][5,14] There exists a 6-state firing squad synchronization algorithm that can synchronize any $m \times n$ rectangular array in $2(m + n) - 4$ steps.

[Theorem 6][4,5,14] There exists a 13-state firing squad synchronization algorithm that can synchronize any $m \times n$ rectangular array in optimum $m + n + \max(m, n) - 3$ steps.

In Fig. 4 (left), we show snapshots of the 13-state optimum-time generalized firing squad synchronization algorithm. Shaded squares in Fig. 4 (right) mean cells that change their states in the computation. Figure 5 shows a comparison of the state-change complexity of the algorithm in the case where the initial general is located on C_1,$C_{n/4}$ and $C_{n/2}$. Figure 6 illustrates the state-change complexities between those two linear- and optimum-time algorithms for rectangular arrays.

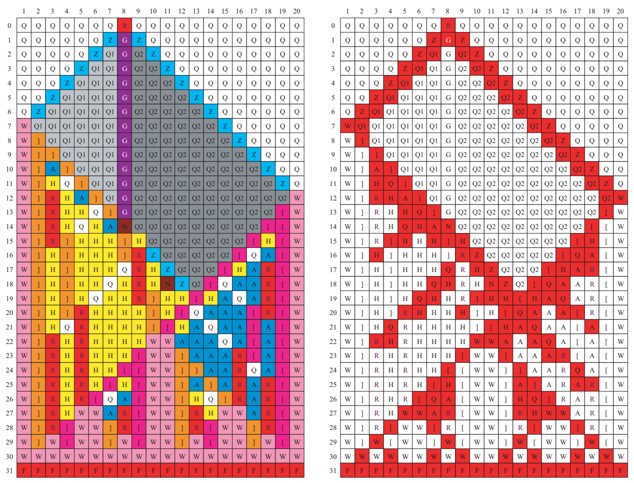

Figure 4: Snapshots of the generalized 13-states synchronization algorithm.

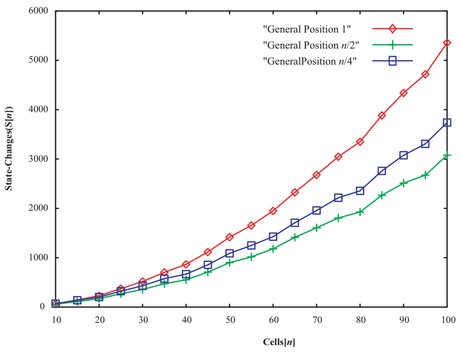

Figure 5: Comparison of state-changes in the generalized 13-states synchronization algorithm.

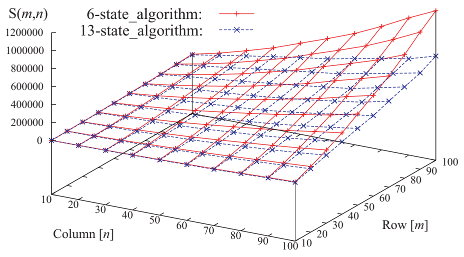

Figure 6: State-changes in the two-dimensional synchronization algorithms.

5 Conclusions

We have implemented most of the synchronization algorithms developed so far and investigated their state-change complexities on a computer. A comparative study of the state-change complexity in optimum- and linear-time synchronization algorithms on one- and two-dimensional cellular arrays has been presented.

References

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

[2] P. C. Fischer: Generation of primes by a one-dimensional real-time iterative array. *J. of ACM*, vol. 12, No. 3 (1965), pp. 388-394.

[3] Hans-D., Gerken: Über Synchronisations-Probleme bei Zellularautomaten. Diplomarbeit (1987), Institut fur Theoretische Informatik, Technische Universitat Braunschweig, pp. 50.

[4] M. Hisaoka, M. Maeda and H. Umeo: An implementation of two-dimensional optimum-time firing squad synchronization algorithms. *Proc. of Annual Conference IEICE*, D-1-9(in Japanese), (2004).

[5] M. Maeda and H. Umeo: A design of two-dimensional firing squad synchronization algorithms and its implementation. *Proc. of Cellular Automata 2001*, pp.102-107(in Japanese), (2001).

[6] K. Matsumoto, T. Sogabe and H. Umeo: State-change complexity for firing squad synchronization algorithms on one-dimensional cellular automata. *Proc. of IEEJ*, 3-099, pp.143-144(in Japanese), (2004).

[7] K. Matsumoto and Hiroshi Umeo: A State-Change Complexity in Synchronization Algorithms for Cellular Automata. *SICE Annual Conference 2004 in Sapporo (SICE2004)*, P-57, (2004).

[8] J. Mazoyer: A six-state minimal time solution to the firing squad synchronization problem. *Theoretical Computer Science*, vol. 50 (1987), pp. 183-238.

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

[10] 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.

[11] P. Sanders, R. Vollmar and T. Worsch: Cellular automata: Energy consumption and physical feasibility. *Fundamenta Informaticae*, 52(2002) 233-248.

[12] H. Umeo, T. Sogabe and Y. Nomura: Correction, optimization and verification of transition rule set for Waksman's firing squad synchronization algorithm. *Proc. of the Fourth Intern. Conference on Cellular Automata for Research and Industry*, Springer, pp. 152-160(2000).

[13] H. Umeo, M. Hisaoka and T. Sogabe: A comparative study ofoptimum-time synchronization algorithms for one-dimensional cellular automata -A survey-. *Proc. of the 6th International Conference on Cellular Automata for Research and Industry*, LNCS 3305, Springer-Verlag, pp.50-60(2004).

[14] 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).

[15] R. Vollmar: On Cellular Automata with a Finite Number of State Change. *Computing, Supplementum*, vol. 3(1981), pp. 181-191.

[16] R. Vollmar: Some Remarks about the "Efficiency" of Polyautomata. *International Journal of Theoretical Physics*, vol. 21, No. 12(1982), pp. 1007-1015.

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

[18] J. B. Yunés: Seven-state solution to the firing squad synchronization problem. *Theoretical Computer Science*, 127 (1994), pp.313-332.