A Class of Minimum-Time Minimum-State-Change Generalized FSSP Algorithms

Hiroshi Umeo(✉) and Keisuke Imai

University of Osaka Electro-Communication,
Hastu-cho, 18-8, Neyagawa-shi, Osaka 572-8530, Japan
umeo@cyt.osakac.ac.jp

Abstract. The firing squad synchronization problem (FSSP, for short) on cellular automata has been studied extensively for more than fifty years, and a rich variety of FSSP algorithms has been proposed. Here we consider the FSSP from a view point of state-change-complexity that models the energy consumption of SRAM-type storage with which cellular automata might be built. In the present paper, we propose a class of minimum-time, minimum-state-change generalized FSSP (GFSSP, for short) algorithms for synchronizing any one-dimensional (1D) cellular automaton, where the synchronization operations are started from any cell in the array. We construct two minimum-time minimum-state-change GFSSP algorithms: one is based on Goto's algorithm, known as the first minimum-time FSSP algorithm that is reconstructed again recently in Umeo et al. [13], and the other is based on Gerken's one. These algorithms are optimum not only in time but also in state-change complexity.

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 [6] to synchronize some/all parts of self-reproducing cellular automata. The problem has been studied extensively for more than fifty years, and a rich variety of synchronization algorithms has been proposed.

Here we consider the FSSP from a view point of state-change-complexity that models the energy consumption of SRAM-type storage with which cellular automata might be built. In the present paper, we propose a class of $n - 2 + \max(k, n - k + 1)$ minimum-time, $\Theta(n \log n)$ minimum-state-change generalized FSSP (GFSSP, for short) algorithms for synchronizing any one-dimensional (1D) cellular automaton of length n, where the synchronization operations are started from any cell k ($1 \leq k \leq n$) in the array. We construct two minimum-time minimum-state-change GFSSP algorithms, one is based on Goto's algorithm, known as the first minimum-time FSSP algorithm that is reconstructed again

S. El Yacoubi et al. (Eds.): ACRI 2016, LNCS 9863, pp. 144–154, 2016.
DOI: 10.1007/978-3-319-44365-2_14

recently in Umeo et al. [13], and the other is based on Gerken's one. The Goto-based GFSSP algorithm is realized on a cellular automaton with 434 internal states and 13328 state-transition rules. The Gerken-based one is implemented on a cellular automaton with 215 internal states and 4077 state-transition rules. These algorithms are optimum not only in time but also in the state-change complexity. The implemented minimum-time GFSSP algorithms are the first ones having the minimum-state-change complexity.

In Sect. 2 we give a description of the 1D FSSP and review some basic results on FSSP and GFSSP algorithms. Section 3 gives new implementations and generalizations to the GFSSP algorithm having minimum-state-change complexity.

2 Firing Squad Synchronization Problem

2.1 Definition of Firing Squad Synchronization Problem

The firing squad synchronization problem (FSSP, for short) is formalized in terms of a model of cellular automata. Consider a 1D array of finite state 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 one *general* 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 *general* cell is in the *fire-when-ready* state, which is an initiation signal to the array. The FSSP 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 initial general is on the left end of the array in the original FSSP.

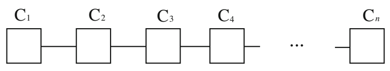

Fig. 1. A one-dimensional (1D) cellular automaton.

Figure 1 shows a finite 1D cellular array consisting of n cells, denoted by C_i, where $1 \leq i \leq n$. The set of states and the next-state transition function must be independent of n. Without loss of generality, we assume $n \geq 2$. The tricky part of the problem is that the same kind of soldiers having a fixed number of states must be synchronized, regardless of the length n of the array.

A formal definition of the FSSP is as follows: a cellular automaton $\mathcal{M}$ is a pair $\mathcal{M} = (\mathcal{Q}, \delta)$, where

1. $\mathcal{Q}$ is a finite set of states with three distinguished states $\mathtt{G}$, $\mathtt{Q}$, and $\mathtt{F}$. $\mathtt{G}$ is an initial general state, $\mathtt{Q}$ is a quiescent state, and $\mathtt{F}$ is a firing state, respectively.
2. δ is a next state function such that $\delta : \mathcal{Q} \cup \{*\} \times \mathcal{Q} \times \mathcal{Q} \cup \{*\} \rightarrow \mathcal{Q}$. The state $* \notin \mathcal{Q}$ is a pseudo state of the border of the array.
3. The quiescent state $\mathtt{Q}$ must satisfy the following conditions: $\delta(\mathtt{Q}, \mathtt{Q}, \mathtt{Q}) = \delta(*, \mathtt{Q}, \mathtt{Q}) = \delta(\mathtt{Q}, \mathtt{Q}, *) = \mathtt{Q}$.

A cellular automaton $\mathcal{M}_n$ of length n, consisting of n copies of $\mathcal{M}$, is a 1D array whose positions are numbered from 1 to n. Each $\mathcal{M}$ is referred to as a cell and denoted by C_i, where $1 \leq i \leq n$. We denote a state of C_i at time (step) t by $\mathtt{S}_i^t$, where $t \geq 0, 1 \leq i \leq n$. A *configuration* of $\mathcal{M}_n$ at time t is a function $\mathcal{C}^t : [1, n] \rightarrow \mathcal{Q}$ and denoted as $\mathtt{S}_1^t \mathtt{S}_2^t$ .... $\mathtt{S}_n^t$. A *computation* of $\mathcal{M}_n$ is a sequence of configurations of $\mathcal{M}_n$, $\mathcal{C}^0$, $\mathcal{C}^1$, $\mathcal{C}^2$, ..., $\mathcal{C}^t$, ..., where $\mathcal{C}^0$ is a given initial configuration. The configuration at time $t+1$, $\mathcal{C}^{t+1}$, is computed by synchronous applications of the next transition function δ to each cell of $\mathcal{M}_n$ in $\mathcal{C}^t$ such that:

$$\mathtt{S}_1^{t+1} = \delta(*, \mathtt{S}_1^t, \mathtt{S}_2^t),\ \mathtt{S}_i^{t+1} = \delta(\mathtt{S}_{i-1}^t, \mathtt{S}_i^t, \mathtt{S}_{i+1}^t), \text{ and } \mathtt{S}_n^{t+1} = \delta(\mathtt{S}_{n-1}^t, \mathtt{S}_n^t, *).$$

A *synchronized configuration* of $\mathcal{M}_n$ at time t is a configuration $\mathcal{C}^t$, $\mathtt{S}_i^t = \mathtt{F}$, for any $1 \leq i \leq n$.

The FSSP is to obtain an $\mathcal{M}$ such that, for any $n \geq 2$,

1. A synchronized configuration at time $t = T(n)$, $\mathcal{C}^{T(n)} = \overbrace{\mathtt{F}, \cdots, \mathtt{F}}^{n}$ can be computed from an initial configuration $\mathcal{C}^0 = \mathtt{G}\overbrace{\mathtt{Q}, \cdots, \mathtt{Q}}^{n-1}$.
2. For any t, i such that $1 \leq t \leq T(n) - 1$, $1 \leq i \leq n, \mathtt{S}_i^t \neq \mathtt{F}$.

The generalized FSSP (GFSSP) is to obtain an $\mathcal{M}$ such that, for any $n \geq 2$ and for any k such that $1 \leq k \leq n$,

1. A synchronized configuration at time $t = T(k, n)$, $\mathcal{C}^{T(k,n)} = \overbrace{\mathtt{F}, \cdots, \mathtt{F}}^{n}$ can be computed from an initial configuration $\mathcal{C}^0 = \overbrace{\mathtt{Q}, \cdots, \mathtt{Q}}^{k-1}\mathtt{G}\overbrace{\mathtt{Q}, \cdots, \mathtt{Q}}^{n-k}$.
2. For any t, i, such that $1 \leq t \leq T(k, n) - 1$, $1 \leq i \leq n, \mathtt{S}_i^t \neq \mathtt{F}$.

No cells fire before time $t = T(k, n)$. We say that $\mathcal{M}_n$ is synchronized at time $t = T(k, n)$ and the function $T(k, n)$ is a time complexity for the synchronization.

2.2 Some Related Results on FSSP and GFSSP

Here we summarize some basic results on FSSP algorithms.

- **Minimum-time FSSP algorithms with a general at one end**
  The FSSP problem was first solved by J. McCarthy and M. Minsky who presented a $3n$-step algorithm for n cells. In 1962, the first minimum-time,

i.e. $(2n-2)$-step, synchronization algorithm was presented by Goto [3], with each cell having several thousands of states. Waksman [16] presented a 16-state minimum-time synchronization algorithm. Afterward, Balzer [1] and Gerken [2] developed an eight-state algorithm and a seven-state synchronization algorithm, respectively, thus decreasing the number of states required for the synchronization. In 1987, Mazoyer [4] developed a six-state synchronization algorithm which, at present, is the algorithm having the fewest states.

Theorem 1 **(**Goto [3], Waksman [16]**).** There exists a cellular automaton that can synchronize any 1D array of length n in minimum $2n-2$ steps, where the general is located at a left (or right) end.

- **Generalized minimum-time FSSP algorithms**
  The generalized FSSP (GFSSP, for short) has also been studied, where an initial general can be located at any position in the array. The same kind of soldiers having a fixed number of states must be synchronized, regardless of the position k of the general and the length n of the array. Moore and Langdon [7] first studied the problem and presented a 17-state minimum-time GFSSP algorithm, i.e. operating in $n-2+\max(k, n-k+1)$ steps for n cells with the general on the kth cell from left end of the array. See Umeo et al. [12] for a survey on GFSSP algorithms and their implementations. Concerning the GFSSP, it has been shown impossible to synchronize any array of length n in less than $n-2+\max(k, n-k+1)$ steps, where the general is located on C_k, $1 \leq k \leq n$.

Theorem 2 **(**Moore and Langdon [7] (Lower Bounds)**).** The minimum-time in which the generalized firing squad synchronization could occur is no earlier than $n-2+\max(k, n-k+1)$ steps, where the general is located on the kth cell from left end.

Theorem 3 **(**Umeo et al. [12]**).** There exists an 8-state cellular automaton that can synchronize any 1D array of length n in minimum $n-2+\max(k, n-k+1)$ steps, where the general is located on the kth cell from left end.

3 A Class of Minimum-Time, Minimum-State-Change GFSSP Algorithms

3.1 Designing Minimum-Time GFSSP Algorithms

In this section we develop a general methodology for designing a minimum-time GFSSP algorithm based on freezing-thawing technique. We can construct a minimum-time GFSSP algorithm from any minimum-time FSSP algorithm with a general at one end. The *freezing-thawing technique* developed in Umeo [10] enables us to have an FSSP algorithm with an arbitrary synchronization delay for 1D arrays. The freezing-thawing technique can be employed efficiently for the design of a minimum-time, minimum-state-change GFSSP algorithms in Sects. 3.3 and 3.4. The technique is described as follows:

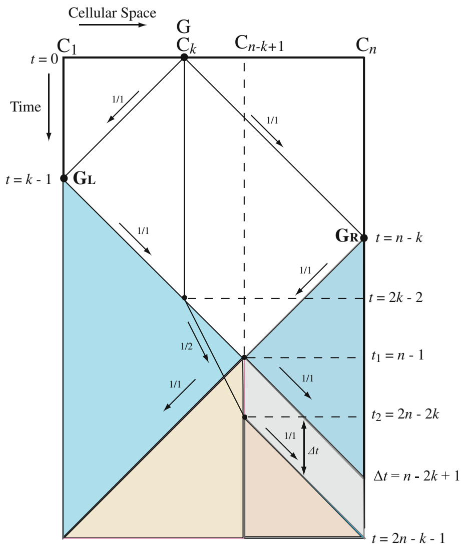

Fig. 2. Space-time diagram for the construction of minimum-time GFSSP algorithm.

Theorem 4 (Umeo [10])**.** Let t_0, t_1, t_2 and Δt be any integer such that $t_0 \geq 0$, $t_1 = t_0 + n - 1$, $t_1 \leq t_2$ and $\Delta t = t_2 - t_1$. We assume that a usual synchronization operation is started at time $t = t_0$ by generating a special signal which acts as a general at the left end of 1D array of length n. We also assume that the right end cell of the array receives another special signals from outside at time $t_1 = t_0 + n - 1$ and $t_2 = t_1 + \Delta t$, respectively. Then, there exists a 1D cellular automaton that can synchronize the array of length n at time $t = t_0 + 2n - 2 + \Delta t$.

Consider a cellular array C_1, C_2, ..., C_n of length n with an initial general on C_k, where $1 \leq k \leq n$. At time $t = 0$ the general sends a unit speed (1 cell/1 step) signal to both ends. The cell C_k keeps its state to indicate its initial position on the array. The signal reaches at the left and right ends at time $t = k - 1$ and $t = n - k$, respectively, and generates a new general, denoted as $\mathrm{G_L}$ and $\mathrm{G_R}$ at each end. In Fig. 2, we illustrate a space-time diagram for the GFSSP construction. Each general, $\mathrm{G_L}$ and $\mathrm{G_R}$, starts minimum-time synchronization operations for the cellular space where the general is at its end by sending out a wake-up signal. At time $t = n - 1$ the two signals collide with each other on the cell C_{n-k+1} and the cellular space is divided into two parts by the collision. First, we consider the case where the initial general is in the left half of the given cellular space, i.e. $k \leq n - k + 1$. The wake-up signal generated by $\mathrm{G_L}$ reaches C_k at time $t = 2k - 2$, then collides with the wake-up signal generated by $\mathrm{G_R}$. The larger part (left one in this case) is synchronized by a usual way, however, the small one is synchronized with time delay $\Delta t = n - 2k + 1$. The wake-up signal for the larger part splits into two signals on C_k, one is an original wake-up

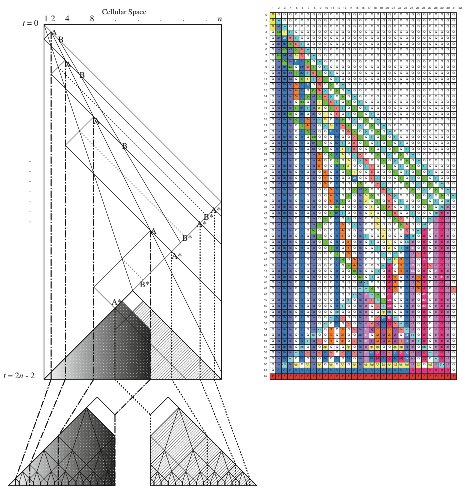

Fig. 3. An overview of the reconstructed Goto's FSSP algorithm (left) and its snapshots on 32 cells of the 166-state, 4378-transition-rule implementation in Umeo et al. [13].

signal and the other is a new slow signal which follows the wake-up signal at 1/2-speed. Note that the wake-up signal for the smaller part (right one in this case) never reaches C_k. As for the synchronization for the smaller part, a freezing-signal is generated at time $t_1 = n - 1$ on C_{n-k+1} and the state configuration in the smaller part is frozen by the propagation of the 1/1-speed right-going freezing signal. At time $t_2 = 2n - 2k$, the split slow signal reaches C_{n-k+1} and there a thawing signal is generated. The thawing signal thaws the frozen configuration progressively. Theorem 4 shows that the smaller part of length k is synchronized at time $t = 2n - k - 1$. The larger part is also synchronized at time $t = 2n - k - 1$. Thus, the whole space can be synchronized at time $t = 2n - k - 1 = n - 2 + \max(k, n - k + 1)$. Similar discussions can be made in the case where the initial general is in the right half of the cellular space. It is seen that any minimum-time FSSP algorithm with a general at one end can be embedded as a sub-algorithm for the synchronization of divided parts. A similar technique was used for solving FSSP with many generals in Schmid and Worsch [8]. Thus, we have:

Theorem 5. The schema given above can realize a minimum-time GFSSP algorithm by implementing two minimum-time FSSP algorithms with a general at one end.

3.2 State-Change Complexity

Vollmar [15] introduced a *state-change complexity* in order to measure the efficiency of cellular automata, motivated by energy consumption in certain SRAM-type memory systems. The state-change complexity is defined as the sum of *proper* state changes of the cellular space during the computations. A formal definition is as follows: Consider an FSSP (GFSSP) algorithm operating on n cells. Let $T(n)$ (resp., $T(k,n)$) be synchronization steps of the FSSP (GFSSP) algorithm. We define a matrix C of size $T(n) \times n$ ($T(n)$ rows, n columns) (resp., $T(k,n) \times n$ ($T(k,n)$ rows, n columns)) over $\{0,1\}$, where each element $c_{i,j}$ on ith row, jth column of the matrix C is defined:

$$c_{i,j} = \begin{cases} 1 & \mathtt{S}_i^j \neq \mathtt{S}_i^{j-1} \\ 0 & \textit{otherwise.} \end{cases} \tag{1}$$

The state-change complexity $SC(n)$(resp., $SC_g(n)$) of the FSSP (GFSSP) algorithm is the sum of 1's elements in C defined as:

$$SC(n) = \sum_{j=1}^{T(n)} \sum_{i=1}^{n} c_{i,j}, \tag{2}$$

$$SC_g(n) = 1/n \sum_{k=1}^{n} \sum_{j=1}^{T(k,n)} \sum_{i=1}^{n} c_{i,j}. \tag{3}$$

Vollmar [15] showed that $\Omega(n \log n)$ state-changes are required for synchronizing n cells in $(2n-2)$ steps.

Theorem 6 (Vollmar [15]). $\Omega(n \log n)$ state-change is necessary for synchronizing n cells in minimum-steps.

Gerken [2] presented a minimum-time, $\Theta(n \log n)$ minimum-state-change FSSP algorithm with a general at one end.

Theorem 7 (Gerken [2]). $\Theta(n \log n)$ state-change is sufficient for synchronizing n cells in $2n-2$ steps.

Goto's algorithm (Goto [3]) has been known as the first minimum-time FSSP algorithm, however the paper itself has been a mysterious one for a long time due to its hard accessibility. Umeo [9] reconstructed the Goto's algorithm and it is noted in Umeo [11] that the algorithm has $\Theta(n \log n)$ minimum-state-change complexity. Mazoyer [5] also reconstructed the algorithm again. Yunès [17] gave a new construction of Goto-like algorithms using the Wolfram's rule 60. Recently, Umeo et al. [13] reconstructed the Goto's algorithm again and realized it on a cellular automaton having 166-state and 4378 transition rules.

Theorem 8 (Umeo [11], Umeo et al. [13]). The reconstructed Goto's algorithm has $\Theta(n \log n)$ state-change complexity for synchronizing n cells in $2n-2$ steps.

In order to get a minimum-time, minimum-state-change GFSSP algorithm, we embed the reconstructed Goto's algorithm and Gerken's one with a general at one end. The state-change complexity in the right and left parts in Fig. 2 is

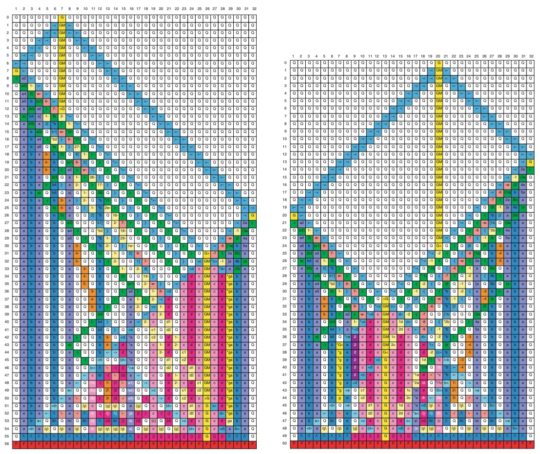

Fig. 4. Snapshots of configurations for Goto-based minimum-time, minimum-state-change GFSSP algorithm developed on $n = 32$ cells with a general on C_7 (left) and C_{20} (right), respectively.

$O((n-k+1)\log(n-k+1))$ and $O(k \log k)$, respectively, thus the total state-change-complexity of the constructed GFSSP algorithm is $O((n-k+1)\log(n-k+1)) + O(k \log k) \leq O(n \log n)$.

Thus, we have:

Theorem 9. There exists a minimum-time, minimum-state-change GFSSP algorithm.

3.3 An Implementation of Goto-Based Minimum-Time, Minimum-State-Change GFSSP Algorithm

The algorithm that Umeo [9] reconstructed is a non-recursive algorithm consisting of a marking phase and a $3n$-step synchronization phase. In the first phase, by printing a special marker in the cellular space, the entire cellular space is divided into many smaller subspaces, each length of which increases exponentially with a common ratio of two, that is 2^j, for any integer $j \geq 1$. The exponential marking is made by counting cells from both left and right ends of a given cellular space. In the second phase, each subspace is synchronized by starting a well-known conventional $3n$-step synchronization algorithm from center point of each divided subspace. Figure 3 illustrates an overview of the reconstructed Goto's algorithm. It can be seen that the overall algorithm does not call itself. Based on the reconstructed Goto's algorithm, we realize a minimum-time, minimum-state-change GFSSP algorithm on a cellular automaton with 434 internal states and 13328 state-transition rules. Figure 4 shows some snapshots for the constructed GFSSP algorithm on 32 cells with a general on C_7 (left) and C_{20} (right), respectively.

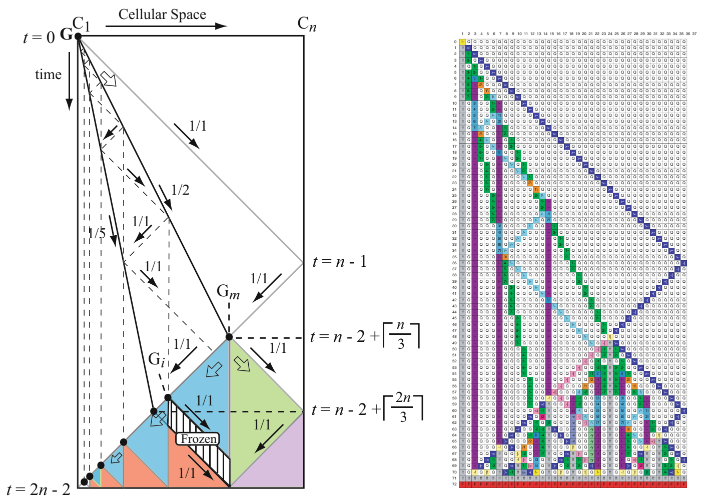

Fig. 5. Space-time diagram of Gerken's FSSP algorithm (left) and its snapshots on 37 cells.

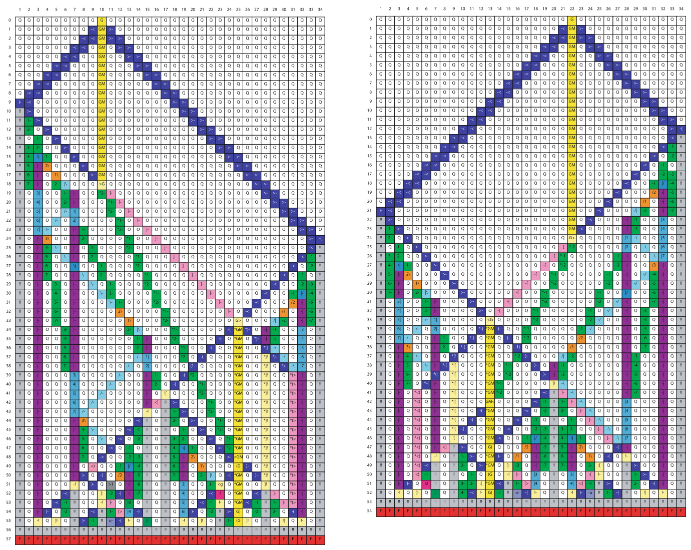

Fig. 6. Snapshots of configurations of Gerken-based minimum-time, minimum-state-change GFSSP algorithm developed on $n = 34$ cells with a general on C_{10} (left) and C_{22} (right), respectively.

3.4 An Implementation of Gerken-Based Minimum-Time, Minimum-State-Change GFSSP Algorithm

Gerken [2] constructed a minimum-time 157-state FSSP algorithm and showed that the algorithm has a minimum-state-change complexity. The algorithm is the first one having the $\Theta(n \log n)$ minimum-state-change complexity for synchronizing n cells with a general at one end. Figure 5 gives a space-time diagram for the algorithm (left) and some snapshots for the synchronization processes on 37 cells. Based on the algorithm, we realize a minimum-time, minimum-state-change GFSSP algorithm on a cellular automaton with 215 internal states and 4077 state-transition rules. Figure 6 shows some snapshots for the constructed GFSSP algorithm on 34 cells with a general on C_{10} (left) and C_{22} (right), respectively. Different snapshots can be found in Umeo et al. [14].

4 Summary

We studied the FSSP from a view point of state-change-complexity that models the energy consumption of SRAM-type storage with which cellular automata might be built. We have constructed two minimum-time minimum-state-change GFSSP algorithms: one is based on Goto's algorithm, known as the first minimum-time FSSP algorithm, and the other is based on Gerken's one. The Goto-based GFSSP algorithm is realized on a cellular automaton with 434 internal states and 13328 state-transition rules. The Gerken-based one is implemented on a cellular automaton with 215 internal states and 4077 state-transition rules. These algorithms are optimum not only in time but also in the state-change complexity. The implemented minimum-time GFSSP algorithms are the first ones having the minimum-state-change complexity.

References

1. Balzer, R.: An 8-state minimal time solution to the firing squad synchronization problem. Inf. Control **10**, 22–42 (1967)
2. Gerken, H.D.: Über Synchronisationsprobleme bei Zellularautomaten. Diplomarbeit, Institut für Theoretische Informatik, Technische Universität Braunschweig, p. 50 (1987)
3. Goto, E.: A minimal time solution of the firing squad problem. Dittoed course notes for Applied Mathematics 298, Harvard University, pp. 52–59 (1962)
4. Mazoyer, J.: A six-state minimal time solution to the firing squad synchronization problem. Theoret. Comput. Sci. **50**, 183–238 (1987)
5. Mazoyer, J.: A minimal-time solution to the FSSP without recursive call to itself and with bounded slope of signals. Unpublished draft version, pp. 1–25 (1997)
6. Moore, E.F.: The firing squad synchronization problem. In: Moore, E.F. (ed.) Sequential Machines, Selected Papers, pp. 213–214. Addison-Wesley, Reading MA (1964)
7. Moore, F.R., Langdon, G.G.: A generalized firing squad problem. Inf. Control **12**, 212–220 (1968)

8. Schmid, H., Worsch, T.: The firing squad synchronization problem with many generals for one-dimensional CA. In: Proceedings of IFIP World Congress, pp. 111–124 (2004)
9. Umeo, H.: A note on firing squad synchronization algorithms - a reconstruction of Goto's first-in-the-world optimum-time firing squad synchronization algorithm. In: Kutrib, M., Worsch, T. (eds.) Proceedings of IFIP Cellular Automata Workshop. 1996, Schloss Rauischholzhausen, Giessen, Germany, p. 65 (1996)
10. Umeo, H.: A simple design of time-efficient firing squad synchronization algorithms with fault-tolerance. IEICE Trans. Inf. Syst. E87-D(3), 733–739 (2011)
11. Umeo, H.: Firing squad synchronization problem in cellular automata. In: Meyers, R.A. (ed.) Encyclopedia of Complexity and System Science, vol. 4, pp. 3537–3574. Springer, New York (2009)
12. Umeo, H., Kamikawa, N., Nishioka, K., Akiguchi, S.: Generalized firing squad synchronization protocols for one-dimensional cellular automata - a survey. Acta Phys. Pol. B, Proc. Suppl. 3, 267–289 (2010)
13. Umeo, H., Hirota, M., Nozaki, Y., Imai, K., Sogabe, T.: A reconstruction of Goto's FSSP algorithm (2016, draft in submission)
14. Umeo, H., Imai, K., Sousa, A.: A Generalized Minimum-Time Minimum-State-Change FSSP Algorithm. In: Dediu, A.-H., Magdalena, L., Martín-Vide, C. (eds.) TPNC 2015. LNCS, vol. 9477, pp. 161–173. Springer, Switzerland (2015). doi:10.1007/978-3-319-26841-5_13
15. Vollmar, R.: Some remarks about the efficiency of polyautomata. Inter. J. Theoret. Phys. **21**(12), 1007–1015 (1982)
16. Waksman, A.: An optimum solution to the firing squad synchronization problem. Inf. Control **9**, 66–78 (1966)
17. Yunès, J.B.: Goto's construction and Pascal's triangle: new insights into cellular automata synchronization. Proceedings of JAC **2008**, 195–203 (2008)