IEICE TRANS. INF. & SYST., VOL.E93–D, NO.12 DECEMBER 2010

PAPER

The Firing Squad Synchronization Problems for Number Patterns on a Seven-Segment Display and Segment Arrays

Kazuya YAMASHITA[†a)], Mitsuru SAKAI[†], Sadaki HIROSE[†], *and* Yasuaki NISHITANI[††], *Members*

SUMMARY The *Firing Squad Synchronization Problem* (FSSP), one of the most well-known problems related to cellular automata, was originally proposed by Myhill in 1957 and became famous through the work of Moore [1]. The first solution to this problem was given by Minsky and McCarthy [2] and a minimal time solution was given by Goto [3]. A significant amount of research has also dealt with variants of this problem. In this paper, from a theoretical interest, we will extend this problem to number patterns on a seven-segment display. Some of these problems can be generalized as the FSSP for some special trees called segment trees. The FSSP for segment trees can be reduced to a FSSP for a one-dimensional array divided evenly by joint cells that we call segment array. We will give algorithms to solve the FSSPs for this segment array and other number patterns, respectively. Moreover, we will clarify the minimal time to solve these problems and show that there exists no such solution.

key words: *firing squad synchronization problem, cellular automaton, seven-segment display, segment tree, segment array*

1. Introduction

The *Firing Squad Synchronization Problem* (FSSP), one of the most well-known problems related to cellular automata, was originally proposed by Myhill in 1957 and became famous through the work of Moore [1]. The first solution to the FSSP was given by Minsky and McCarthy [2], and requires $3n + O(\log n)$ steps for n cells using thirteen states. It is easy to show that any solution to the FSSP requires at least $2(n - 1)$ steps. Goto [3] gave a minimal time solution using several thousands of states. Inspired by his results, many researchers tried to reduce the number of states: in 1966, Waksman [4] gave a solution with sixteen states; in 1967, Balzer [5] gave a solution with eight states; in 1987, Gerken [6] reduced this to seven states; and in the same year, Mazoyer [7] gave a solution with six states, the minimal time solution with the fewest states at present. Moreover, it is shown in [5] that any solution needs at least five states.

A significant amount of research has also dealt with variants of the FSSP. The FSSP has been studied on a ring of n cells [8], and on arrays of two and three dimensions [9], [10]. The FSSP with multiple generals [11]–[13] has also been studied. The FSSP for Cayley graphs [14] and another particular class of graphs [15] have also been studied. Some constrained variants of the FSSP have been developed to find solutions on reversible (i.e., backward deterministic) cellular automata [16] and cellular automata with a number-conserving property (i.e., a state is a tuple of non-negative integers whose sum is constant) [17]. Other kinds of constraints that have been considered involve the amount of information exchange between any pair of adjacent cells [18], [19].

In this paper, from a theoretical interest, we will extend this problem to number patterns on a seven-segment display. Some of these problems can be generalized as the FSSP for some special trees called segment trees. The FSSP for segment trees can be reduced to a FSSP for a one-dimensional array divided evenly by joint cells that we call segment array. We will give algorithms to solve the FSSPs for this segment array and other number patterns, respectively. Moreover, we will clarify the minimal time to solve these problems and show that there exists no such solution.

2. The FSSP

Since the FSSP is defined on cellular automata, we will briefly describe them.

A cellular automaton is defined by an interconnection network with a finite automaton. Finite automata on nodes of the network, called *cells*, are copies of the finite automaton and communicate through edges of the network. For a cell v, cells connected to v directly are called *neighbor* cells of v. Each cell changes its state in discrete time by a state transition function of its own state and its neighbor cells' states (a set of a cell v and its neighbor cells is called its *neighborhood*). An interval of the time in which cells change their states once is called a *step*.

In a one-dimensional cellular automaton, an interconnection network is a path of size n for any positive integer n, and the neighbor cells of v are its left and right ones. This is therefore called a three-neighborhood one-dimensional cellular automaton. In the two-dimensional case, an interconnection network is an $m \times n$ array for any positive integers m and n, and the neighbor cells of v consist of those cells horizontally and vertically adjacent to it. A cellular automaton of this type is called a five-neighborhood two-dimensional cellular automaton.

2.1 Definition of the FSSP

The FSSP is defined on three-neighborhood one-dimensional cellular automata.

Consider a one-dimensional array of n cells, consisting

Manuscript received June 25, 2010.

[†]The authors are with the Faculty of Engineering, University of Toyama, Toyama-shi, 930–8555 Japan.

[††]The author is with the Faculty of Engineering, Iwate University, Morioka-shi, 020–8551 Japan.

a) E-mail: kazuya@eng.u-toyama.ac.jp

DOI: 10.1587/transinf.E93.D.3276

of $C_0, C_1, \cdots, C_{n-1}$. Assume that n is arbitrary, but finite. The cell C_0 at the left end is called a *general*, and the remainder of cells $C_1, C_2, \cdots, C_{n-1}$ are called *soldiers*. The problem is to design a set of states and a state transition function so that the general can cause all cells to enter a particular terminal state, called the *firing* state, for the first time and at exactly the same time.

A more precise definition is as follows.

$\mathcal{A} = (\mathcal{S}, \delta)$ is a finite automaton of each cell C_i $(0 \leq i \leq n-1)$. $\mathcal{S}$ is a finite set of states: a *general* state $G \in \mathcal{S}$, a *quiescent* state $Q \in \mathcal{S}$, and a *firing* state $F \in \mathcal{S}$, each differing from each other. δ is a state transition function, namely, $\delta : (\mathcal{S} \cup \{B\}) \times \mathcal{S} \times (\mathcal{S} \cup \{B\}) \rightarrow \mathcal{S}$. $B \notin \mathcal{S}$ is a signal that shows border of the array. For the quiescent state Q, we define $\delta(s_Q, Q, s_Q) = Q$; $s_Q \in \{Q, B\}$.

Initially, all soldiers are assumed to be in the quiescent state, and only the general is assumed to be in the general state.

The state of C_i $(0 \leq i \leq n-1)$ at time t is shown as s_i^t. The state s_i^{t+1}, which is the state of C_i at time $t+1$, is determined by the state transition function and the states of its neighborhood as follows.

$$s_i^{t+1} = \delta\left(s_{i-1}^t, s_i^t, s_{i+1}^t\right)$$

The state s_{i-1}^t is B for $i = 0$, and the state s_{i+1}^t is B for $i = n-1$.

If all cells first enter the firing state F at time t_F, we say that the array of n cells is *fired* by the finite automaton $\mathcal{A}$. So, the state of each cell C_i $(0 \leq i \leq n-1)$ is as follows.

$$s_i^{t_F} = F \text{ and } s_i^t \neq F \text{ for all } 0 \leq t < t_F$$

The time t_F required to solve the FSSP by a finite automaton $\mathcal{A}$ for an array of n cells is written as $t_F(n, \mathcal{A})$.

A finite automaton $\mathcal{A}$ solving the FSSP for all n is called a *solution*. The *minimal time* of the solutions for an array of n cells is written as $t_{F\min}(n)$.

$$t_{F\min}(n) = \min\{t_F(n, \mathcal{A}) \mid \mathcal{A} \text{ is a solution for the FSSP}\}$$

If $t_F(n, \mathcal{A}) = t_{F\min}(n)$ holds for all n, the solution $\mathcal{A}$ is called the *minimal time solution*.

2.2 Outline of a Minimal Time Solution

An outline of a minimal time solution (Waksman's algorithm [4]) to solve the FSSP is as follows.

At time $t = 0$, the general C_0 simultaneously sends signals[†] at propagation speeds of $\frac{1}{1}, \frac{1}{3}, \frac{1}{7}, \cdots, \frac{1}{2^k-1}, \cdots$, where k is any natural number, to the right. These signals are called the *firing signals* and play an important role in dividing the array into two, four, eight, $\cdots$, equal parts synchronously. When a signal of speed $\frac{1}{1}$ reaches the right end of the array, at time $t = n-1$, the rightmost cell C_{n-1} enters a general state and acts like the original general, except that it sends the firing signals to the left instead of the right. The signal of speed $\frac{1}{3}$ which C_0 sent at time $t = 0$ and the signal of speed $\frac{1}{1}$ which C_{n-1} sent at time $t = n-1$ meet at the center of C_0 and C_{n-1}. Then, the middle cell enters the general state and acts like the original general, sending the firing signals to the left and also to the right. As a result of this process, the array is divided into two equal parts. This process is repeated until all cells in the array are in the general state. Every cell enters the firing state at the next step. Note that the above process insures synchronization of the array and also permits the determination of firing if it is in the general state and the cells on either side of it are also in the general state.

It is easily understood that the time required for the above-mentioned process is $2(n-1)$ steps. It equals the time required for the signal at propagation speed of $\frac{1}{1}$, which C_0 sent, to go to C_{n-1} and return.

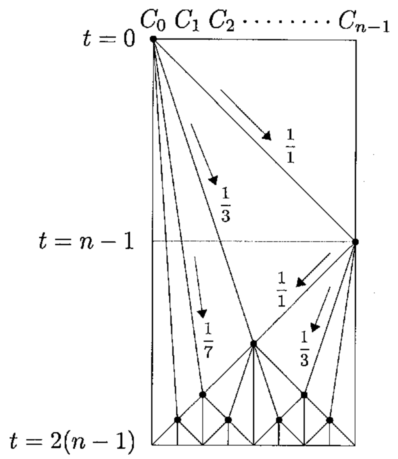


Fig. 1 Time-space diagram of a minimal time solution.

Figure 1 shows a time-space diagram of the minimal time solution. The horizontal axis is the cell space, and the vertical axis is the time. The fractions in the figure are propagation speeds of the firing signals, and the dots show that those cells are in the general state.

3. The FSSPs for Number Patterns on a Seven-Segment Display

In this section, from a theoretical interest, we will extend the FSSP to number patterns on a seven-segment display.

3.1 Number Patterns on a Seven-Segment Display

A seven-segment display, shown in Fig. 2, is commonly used in electronics to display decimal numbers. It is composed of seven elements, called *segment*, as its name indicates. Individually turned on or off, they can be combined to produce simplified representations of numbers.

Numbers from 0 to 9 are displayed on a seven-segment display as patterns, as shown in Fig. 3.

[†]A signal at propagation speed of $\frac{1}{t}$ moves at a rate of one cell every t steps.

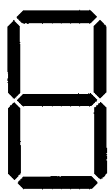


Fig. 2 A seven-segment display.

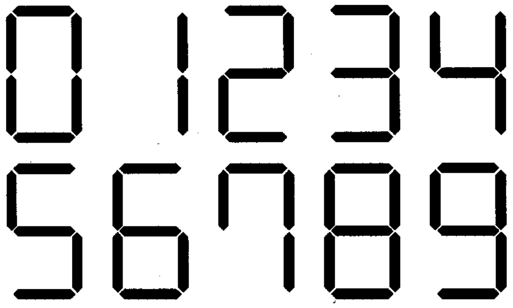


Fig. 3 Number patterns displayed on a seven-segment display.

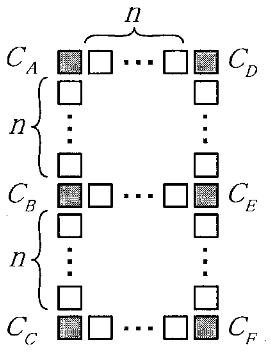


Fig. 4 Modeling of a seven-segment display.

3.2 Modeling of Number Patterns

A seven-segment display is modeled on a two-dimensional cell space, as shown in Fig. 4. The seven segments consist of n cells, respectively. Six cells, called the *joint cells*, are placed between the segments. They are shaded in the figure. The six joint cells are called C_A, C_B and C_C from top to bottom of the left column, and C_D, C_E and C_F from top to bottom of the right column.

The FSSPs for number patterns on a seven-segment display can be defined as problems on five-neighborhood two-dimensional cellular automata. Initial configurations of number patterns are shown in Fig. 5. The general is the joint cell at the starting position of the stroke order of the number[†], thus the general of number patterns 0, 1, 6, 8 and 9 is the joint cell C_D, and that of number patterns 2, 3, 4, 5 and 7 is the joint cell C_A. These are shaded in the figure. Of course, we assume that the length of one segment n is arbitrary, but finite.

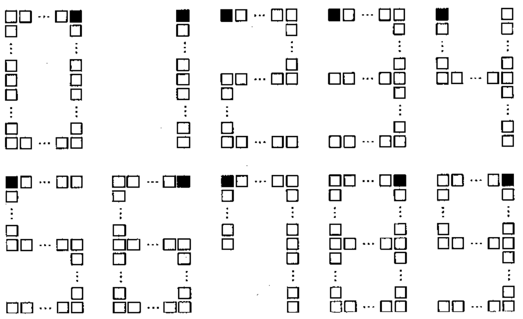


Fig. 5 Initial configurations of number patterns.

4. The Firing Squad Synchronization Algorithms

Some of the FSSPs for number patterns on a seven-segment display can be generalized as the FSSP for some special trees called segment trees. The FSSP for segment trees can be reduced to a FSSP for one-dimensional array divided evenly by joint cells. In this section, we will give algorithms to solve the FSSPs for this array and other number patterns, respectively.

First, please pay attention to the joint cells at the initial configurations of number patterns shown in Fig. 5.

On number patterns 2, 3, 4, 5, 6, 8 and 9, each joint cell C_A, C_B, C_C, C_D, C_E and C_F has a different neighborhood from the other cells in the segments. Therefore, they can recognize that they are joint cells with their neighborhood. However, the joint cells C_B and C_E of number pattern 0 and the joint cell C_E of number patterns 1 and 7 cannot recognize that they are joint cells because they have the same neighborhood as cells in the segments. This difference is very important in the composition of the algorithms.

4.1 Algorithms for Number Patterns 2, 3, 4, 5, 6, 8 and 9

In number patterns 2, 3, 4, 5, 6, 8 and 9, number patterns 2, 3, 4 and 5 are so-called trees. In Fig. 6, they are shown as dangling images. The nodes at level i ($0 \leq i \leq 5$) are joint cells and the node at level 0 is the general. Every branch that connects nodes corresponds to a segment. However each branch is drawn as a line in the figure, consisting of n cells.

On the other hand, number patterns 6, 8 and 9 are graphs, other than trees, as shown in Fig. 7. Note the four nodes drawn as circles divided in half by lines in the figure. Each of these four nodes has multiple paths of the same length from the root (general). As one path is sufficient to exchange signals (information) with the general, we divide each of the four nodes into half. They can then be drawn as trees shown in Fig. 8.

By the method described above, the FSSPs for number patterns 2, 3, 4, 5, 6, 8 and 9 can be generalized as the FSSPs for *segment trees* defined below.

The *segment trees* are trees that satisfy the following

[†]We adopt the stroke order generally taught in elementary schools in Japan.

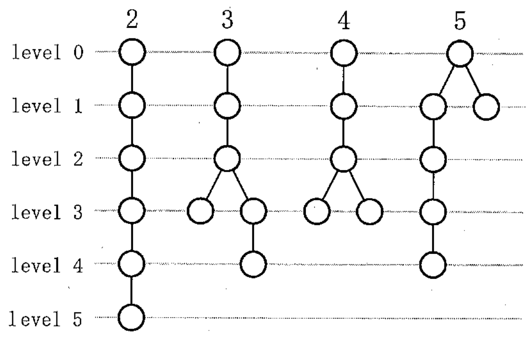


Fig. 6 Trees for number patterns 2, 3, 4 and 5.

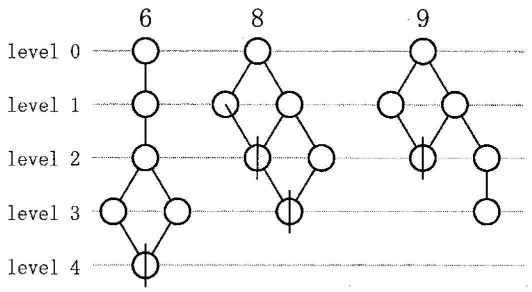


Fig. 7 Graphs for number patterns 6, 8 and 9.

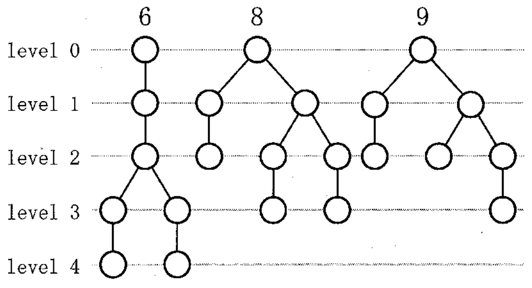


Fig. 8 Trees for number patterns 6, 8 and 9.

conditions.

- A branch is a segment that consists of n cells.
- A node is also a cell, but it is distinguished from cells composing segments.

The number of segments (branches) that exist on the path from the root to a node is called the *level* of the node, and the largest level of all nodes is called the *height* of the segment tree.

Letting the root be a general, the FSSPs for the segment trees can be defined similarly[†].

4.2 The FSSPs for Segment Arrays

We can easily see that the FSSP for a segment tree of height h can be reduced to a FSSP for a one-dimensional array of cells divided evenly by $h+1$ joint cells.

To define the FSSP for such an array precisely, we introduce a new state Q_J called *joint* state. The joint state Q_J satisfies the following conditions where the symbols s_Q, s and $*$ are elements of $\{Q, B\}$, $\mathcal{S}$ and $\mathcal{S} \cup \{B\}$, respectively.

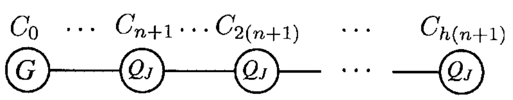


Fig. 9 A segment array.

$$\delta(s_Q, Q_J, s_Q) = Q_J$$
$$\delta(*, s, Q_J) = \delta(*, s, Q)$$
$$\delta(Q_J, s, *) = \delta(Q, s, *)$$

The joint state Q_J conforms with the quiescent state Q. It does not change state when its right and left cells are in the quiescent state or border, and its neighbor right and left cells cannot distinguish its state from the quiescent state. However, once the joint state receives a signal from the general, it can transit to a state, which is different from the state that the quiescent state does, and even behave like the general.

For a cell array $C_0, C_1, \ldots, C_{l(n+1)}, \ldots, C_{h(n+1)}$ of length $h(n+1)+1$, C_0 is in the general state G, $C_{l(n+1)}$ $(1 \le l \le h)$ are in the joint state Q_J, and other cells are in the quiescent state Q. Such an array is called a *segment array*. The segment array is divide into h parts with general or joint cells at each end, as shown in Fig. 9. Thus, the FSSP for the segment array can be defined similarly.

Now, we will present the algorithm to solve the FSSP for segment array. Without loss of generality, we assume that the length of a segment n equals $\alpha l - 1$ for some natural number α.

Algorithm 1:

1. At time $t = 0$, the general C_0 sends signals S_i $(0 \le i \le h\alpha)$ at propagation speeds of $\frac{i}{h\alpha}$.
2. At time $t = n+1$, C_{n+1} receives the signal $S_{h\alpha}$ from C_0, then sends signals T_i $(0 \le i \le h\alpha)$ at propagation speeds of $\frac{i-\alpha}{h\alpha}$[††].
3. At time $t = h(n+1)$, $C_{\frac{i}{\alpha}(n+1)}$ $(0 \le i < h\alpha)$ receive the signals S_i and T_i simultaneously and enter the general state. At the same time, $C_{h(n+1)}$ receives the signal $T_{h\alpha}$ and enters the general state.
4. $C_{\frac{i}{\alpha}(n+1)}$ $(0 \le i \le h\alpha)$ apply the minimal time algorithm to each of the connected arrays.

Figure 10 shows a time-space diagram of the FSSP algorithm described above when $h = 3$, $\alpha = 2$. The circle indicates that the cell C_0 is in the general state and the dots indicate that the cells have entered the general states of the minimal time algorithm of Sect. 2.2. The square indicates that the cell C_{n-1} in the joint state has received the signal S_6 from the general and now beccome to be able to send signals T_i $(0 \le i \le 6)$.

[†]Note that the problems are no longer the problems defined on five-neighborhood two-dimensional cellular automata.

[††]Note that a signal whose propagation speed is positive propagates to the right and a signal whose propagation speed is negative propagates to the left.

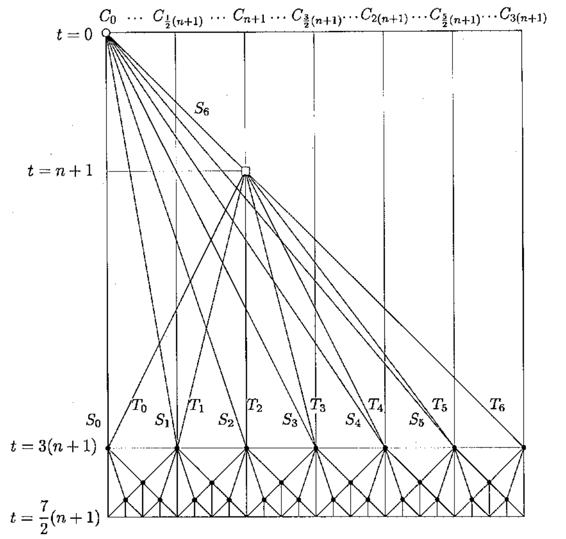

$C_0 \cdots C_{\frac{1}{2}(n+1)} \cdots C_{n+1} \cdots C_{\frac{3}{2}(n+1)} \cdots C_{2(n+1)} \cdots C_{\frac{5}{2}(n+1)} \cdots C_{3(n+1)}$
$t=0$
$t=n+1$
$t=3(n+1)$
$t=\frac{7}{2}(n+1)$
S_0
S_1
S_2
S_3
S_4
S_5
S_6
T_0
T_1
T_2
T_3
T_4
T_5
T_6

Fig. 10 Time-space diagram of the FSSP algorithm for a segment array when $h = 3$, $\alpha = 2$.

It requires $h(n+1)$ steps for $C_{\frac{i}{\alpha}(n+1)}$ $(0 \le i \le h\alpha)$ to enter the general state. Moreover, it requires $\frac{1}{\alpha}(n+1)$ steps for a cell array of length $\frac{1}{\alpha}(n+1)+1$, which has the general states at both ends, to fire using the minimal time algorithm. Therefore, the total time of this algorithm is $h(n+1)+\frac{1}{\alpha}(n+1)$ steps. The bigger α becomes, the smaller $h(n+1)+\frac{1}{\alpha}(n+1)$ becomes. By incrementing the value of α, we can approximate the total time to $h(n+1)$ unboundedly.

As the number patterns 4, 8 and 9 can be represented as segment trees of height 3, the algorithm described above solves the FSSPs for them in $3(n+1)+\frac{1}{\alpha}(n+1)$ steps. Also, as the number patterns 3, 5 and 6 can be represented as segment trees of height 4, the algorithm solves the FSSPs for them in $4(n+1)+\frac{1}{\alpha}(n+1)$ steps. Moreover, as the number pattern 2 can be represented as a segment tree of height 5, the algorithm solves the FSSP for it in $5(n+1)+\frac{1}{\alpha}(n+1)$ steps.

4.3 The Minimal Time to Solve the FSSPs for Segment Arrays

We have presented an algorithm to solve the FSSP for a segment array of length $h(n+1)+1$ that requires $h(n+1)+\frac{1}{\alpha}(n+1)$ steps when n equals $\alpha l - 1$ for some natural number α.

In this section, we show that the minimal time to solve the FSSP for the segment array is $h(n+1)$.

Because the cell $C_{h(n+1)}$ is in the joint state Q_J until the time $t = h(n+1)-1$, the solution time cannot be less than or equal to $h(n+1)$. Therefore, the following lemma holds.

Lemma 1: $t_F(n, \mathcal{A}) \ge h(n+1)$ holds for any solution $\mathcal{A}$ of the FSSP for segment arrays of length $h(n+1)+1$.

We can easily construct a finite automaton $\mathcal{A}_{n_0}$ that can fire a segment array of a particular fixed length $h(n_0+1)+1$ in $h(n_0+1)$ steps, but does not fire a segment array of length $h(n+1)+1$ for n different from n_0. $\mathcal{A}_{n_0}$ consists of the following set of states $\mathcal{S}$ and state transitions. The symbol $*$ indicates any element of $\mathcal{S} \cup \{B\}$.

$$\mathcal{S} = \{X, Q, Q_J, G = S_0, S_1, S_2, \ldots, S_{h(n_0+1)} = F\}$$

$$\begin{array}{ll}
\delta(S_{i-1}, Q, *) = S_i & \text{if } i \ne l(n_0+1), 1 \le l \le h \\
\delta(S_{i-1}, Q_J, *) = S_i & \text{if } i = l(n_0+1), 1 \le l \le h \\
\delta(*, S_{i-1}, s) = S_i & \text{if } s \ne X \\
\delta(S_{i-1}, Q_J, *) = X & \text{if } i \ne n_0+1 \\
\delta(*, S_i, X) = X & \\
\delta(S_{n_0}, Q, *) = X &
\end{array}$$

We now give a brief description of the behavior of $\mathcal{A}_{n_0}$. First, we consider a segment array of length $h(n_0+1)+1$. The general $G = S_0$ changes its state as $S_1, S_2, \ldots, S_{h(n_0+1)}$ successively counting the number of cells. The states propagate to the right one by one. As a result, at time $t = i$, all cells from C_0 to C_i enter the state S_i. So, at time $t = h(n_0+1)$, all cells enter the state $S_{h(n_0+1)} = F$.

When n is less than n_0, at time $t = n+1$ the cell C_{n+1} changes its state from Q_J to X. As the state X propagates to the left, at time $t = 2(n+1) < h(n_0+1)$ all cells are in the states X, Q or Q_J.

Similarly, when n is greater than n_0, at time $t = n_0+1$ the cell C_{n_0+1} changes its state from Q to X. Therefore, at time $t = 2(n_0+1) \le h(n_0+1)$ all cells are in the states X, Q or Q_J.

The finite automaton $\mathcal{A}'$ that mimics the above $\mathcal{A}_{n_0}$ together with an appropriate solution $\mathcal{A}$ is also a solution of the FSSP for segment arrays and $t_F(n_0, \mathcal{A}') = h(n_0+1)$ holds. Thus, the following theorem holds.

Theorem 1: $t_{F\,\min}(n) = h(n+1)$ holds for the FSSP for segment arrays of length $h(n+1)+1$.

4.4 The Minimal Time Solution to the FSSP for Segment Arrays

Though we have shown that the minimal time to solve the FSSP for a segment array of length $h(n+1)+1$ is $h(n+1)$, we must discuss whether such a minimal time solution exists or not. The following lemma shows that no such solution exists.

Lemma 2: Let $\mathcal{A} = (\mathcal{S}, \delta)$ be a solution of the FSSP for segment arrays of length $h(n+1)+1$. If $n+2 > |\mathcal{S}|^2$, $t_F(n, \mathcal{A}) > h(n+1)$ holds.

Proof: We will prove the lemma by assuming $t_F(n, \mathcal{A}) = h(n+1)$ and deriving a contradiction.

Let s_i^t be the state of cell C_i at time t.

Because $n+2 > |\mathcal{S}|^2$, there exist two different integers i and j $((h-1)(n+1) \le i < j \le h(n+1))$ such that $s_{i-1}^i = s_{j-1}^j$, $s_i^i = s_j^j$. That is, there exist two pairs of adjacent cells $C_{i-1}C_i$ and $C_{j-1}C_j$ on the right side of the cell $C_{(h-1)(n+1)}$ and the states $s_{i-1}^i s_i^i$ of the cells $C_{i-1}C_i$ at time i and the states $s_{j-1}^j s_j^j$ of the cells $C_{j-1}C_j$ at time j are the same.

Because $s_{i+1}^{i+1} = \delta\left(s_i^i, Q, Q\right)$ when $(h-1)(n+1) \leq i < h(n+1) - 1$, it follows that $s_{i+k}^{i+k} = s_{j+k}^{j+k}$ when $0 \leq k < h(n+1) - 1 - j$. Furthermore, because $s_i^{i+1} = \delta\left(s_{i-1}^i, s_i^i, Q\right)$ when $(h-1)(n+1) \leq i \leq h(n+1)-1$, it follows that $s_{i+k}^{i+k+1} = s_{j+k}^{j+k+1}$ when $0 \leq k \leq h(n+1) - 1 - j$. Therefore, we have that $s_{h(n+1)-1+i-j}^{h(n+1)+i-j} = s_{h(n+1)-1}^{h(n+1)}$.

From the assumption that the solution time is $h(n+1)$, $s_{h(n+1)-1}^{h(n+1)} = F$ holds. Therefore, $s_{h(n+1)-1+i-j}^{h(n+1)+i-j} = s_{h(n+1)-1}^{h(n+1)} = F$ also holds. As the cell $C_{h(n+1)-1+i-j}$ is in the firing state at time $t = h(n+1) + i - j$, $\mathcal{A}$ is not the solution and this fact contradicts the assumption. □

Theorem 2: There exists no minimal time solution of the FSSP for segment arrays.

4.5 The Firing Squad Synchronization Algorithm for Number Pattern 7

The FSSP for number pattern 7 cannot reduce to the FSSP for a segment array. However, it can be solved smilarly. Number pattern 7 is an array of length $4(n+1)+1$, consisting of $C_0, C_1, \ldots, C_{4(n+1)}$. Initially, the cell C_{n+1} is in the general state, the cells $C_{i(n+1)}$ $(i = 0, 2, 4)$ are in the joint state, and other cells are in the quiescent state.

Assuming n equals $\alpha l - 1$ for some natural number α, we present an algorithm to solve the FSSP for number pattern 7 as follows.

Algorithm 2:

1. At time $t = 0$, the general C_{n+1} sends the signals S_i $(0 \leq i \leq 4\alpha)$ at propagation speed of $\frac{i-\alpha}{3\alpha}$.
2. At time $t = n+1$, $C_{2(n+1)}$ receives the signal $S_{4\alpha}$ from C_{n+1}, then sends signals T_i $(0 \leq i \leq 4\alpha)$ at propagation speeds of $\frac{i-2\alpha}{2\alpha}$.
3. At time $t = 3(n+1)$, $C_{\frac{i}{\alpha}(n+1)}$ $(0 \leq i < 4\alpha)$ receive the signals S_i and T_i simultaneously and enter the general state. At the same time, $C_{4(n+1)}$ receives the signal $T_{4\alpha}$ and enters the general state.
4. $C_{\frac{i}{\alpha}(n+1)}$ $(0 \leq i \leq 4\alpha)$ apply the minimal time algorithm to each of the connected arrays.

Figure 11 shows a time-space diagram of the FSSP algorithm described above when $\alpha = 2$.

It requires $3(n+1)$ steps for $C_{\frac{i}{\alpha}(n+1)}$ $(0 \leq i \leq 4\alpha)$ to enter the general state, so the FSSP for the number pattern 7 can be solved in $3(n+1) + \frac{1}{\alpha}(n+1)$ steps. As α increases, the total time unboundedly approaches to $3(n+1)$ steps.

4.6 The Minimal Time to Solve the FSSP for Number Pattern 7

Because the cell $C_{4(n+1)}$ is in the joint state Q_J until time $t = 3(n+1) - 1$, the solution time cannot be less than or equal to $3(n+1)$. Therefore, the following lemma holds.

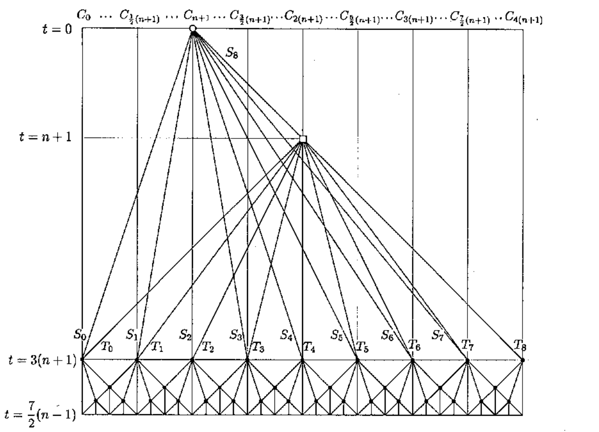

$C_0 \cdots C_{\frac{1}{2}(n+1)} \cdots C_{n+1} \cdots C_{\frac{3}{2}(n+1)} \cdots C_{2(n+1)} \cdots C_{\frac{5}{2}(n+1)} \cdots C_{3(n+1)} \cdots C_{\frac{7}{2}(n+1)} \cdots C_{4(n+1)}$
$t = 0$
S_8
$t = n+1$
S_0 T_0 S_1 T_1 S_2 T_2 S_3 T_3 S_4 T_4 S_5 T_5 S_6 T_6 S_7 T_7 T_8
$t = 3(n+1)$
$t = \frac{7}{2}(n-1)$

Fig. 11 Time-space diagram of the FSSP algorithm for number pattern 7 when $\alpha = 2$.

Lemma 3: $t_F(n, \mathcal{A}) \geq 3(n+1)$ holds for any solution $\mathcal{A}$ of the FSSP for number pattern 7.

By the same discussion in Sect. 4.3, we have the following theorem.

Theorem 3: $t_{F\min}(n) = 3(n+1)$ holds for the FSSP for number pattern 7.

4.7 The Minimal Time Solution for Number Pattern 7

Though the minimal time to solve the FSSP for the number pattern 7 is $3(n+1)$, we show there exists no such solution.

Lemma 4: Let $\mathcal{A} = (\mathcal{S}, \delta)$ be a solution of the FSSP for number pattern 7. If $2(n+1) > |\mathcal{S}|^2$, $t_F(n, \mathcal{A}) > 3(n+1)$ holds.

The proof of lemma 4 is essentially similar to the proof of lemma 2. Thus, the following theorem holds.

Theorem 4: There exists no minimal time solution of the FSSP for number pattern 7.

4.8 The Firing Squad Synchronization Algorithm for Number Pattern 0

The FSSP for number pattern 0 cannot reduce to the FSSP for a segment array. However, it can be also solved similarly. If we divide the joint cell C_F of number pattern 0 into half, it can be regarded as an array of length $6(n+1)+1$, consisting of $C_0, C_1, \ldots, C_{6(n+1)}$. Both the leftmost cell C_0 and the rightmost cell $C_{6(n+1)}$ are C_F. Initially, the cell $C_{3(n+1)}$ is in the general state, the cells $C_{i(n+1)}$ $(i = 0, 2, 5, 6)$ are in the joint state, and other cells are in the quiescent state.

Assuming n equals $\alpha l - 1$ for some natural number α, we present an algorithm to solve the FSSP for number pattern 0 as follows.

Algorithm 3:

1. At time $t = 0$, the general $C_{3(n+1)}$ sends the signals S_i $(0 \leq i \leq 6\alpha)$ at propagation speeds $\frac{i-3\alpha}{3\alpha}$.

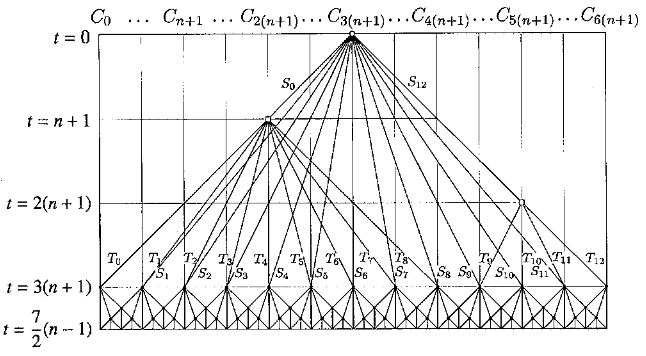

C_0 … C_{n+1} … $C_{2(n+1)}$ … $C_{3(n+1)}$ … $C_{4(n+1)}$ … $C_{5(n+1)}$ … $C_{6(n+1)}$
$t = 0$
$t = n + 1$
$t = 2(n + 1)$
$t = 3(n + 1)$
$t = \frac{7}{2}(n - 1)$
S_0
S_{12}
T_0 T_1 T_2 T_3 T_4 T_5 T_6 T_7 T_8 T_9 T_{10} T_{11} T_{12}
S_1 S_2 S_3 S_4 S_5 S_6 S_7 S_8 S_9 S_{10} S_{11}

Fig. 12 Time-space diagram of FSSP algorithm for number pattern 0 when $\alpha = 2$.

2. At time $t = n + 1$, $C_{2(n+1)}$ receives the signal S_0 from $C_{3(n+1)}$, then sends signals T_i $(0 \leq i \leq 4\alpha)$ at propagation speeds of $\frac{i-2\alpha}{2\alpha}$.
3. At time $t = 2(n + 1)$, $C_{5(n+1)}$ receives the signal $S_{6\alpha}$ from $C_{3(n+1)}$, then sends the signal T_i $(4\alpha < i \leq 6\alpha)$ at propagation speeds of $\frac{i-2\alpha}{2\alpha}$.
4. At time $t = 3(n + 1)$, $C_{\frac{i}{\alpha}(n+1)}$ $(0 < i < 6\alpha)$ simultaneously receive the signals S_i and T_i and enter the general state. At the same time, C_0 receives the signal T_0 and $C_{6(n+1)}$ receives the signal $T_{6\alpha}$. C_0 and $C_{6(n+1)}$ then enter the general state.
5. $C_{\frac{i}{\alpha}(n+1)}$ $(0 \leq i \leq 6\alpha)$ apply the minimal time algorithm to each of the connected arrays.

Figure 12 shows a time-space diagram of the FSSP algorithm described above when $\alpha = 2$.

It requires $3(n + 1)$ steps for $C_{\frac{i}{\alpha}(n+1)}$ $(0 \leq i \leq 6\alpha)$ to enter the general state, so the FSSP for the number pattern 0 can be solved in $3(n + 1) + \frac{1}{\alpha}(n + 1)$ steps. As α increases, the total time unboundedly approaches to $3(n + 1)$ steps.

As mentioned above, it is shown in [8] that the FSSP for a ring of n cells can be solved in n steps and n is the minimal time for this problem. If we regard the number pattern 0 as a ring of $6(n + 1)$ cells, the FSSP for the number pattern 0 can be solved in $6(n + 1)$ steps. Therefore, our algorithm is over $2(n + 1)$ steps faster than the minimal time algorithm for a ring. Because of that, our algorithm uses the information of the shape of pattern, that is, the joint cells C_A, C_C, C_D, and C_F are at the corner of pattern and it is $2n + 3$ cells in height and $n + 2$ cells in length.

4.9 The Minimal Time for Number Pattern 0

Because the cell $C_{6(n+1)}$ is in the joint state Q_J until time $t = 3(n + 1) - 1$, the solution time cannot be less than or equal to $3(n + 1)$. Therefore, the following lemma holds.

Lemma 5: $t_F(n, \mathcal{A}) \geq 3(n + 1)$ holds for any solution $\mathcal{A}$ of the FSSP for number pattern 0.

By the same discussion in Sect. 4.3, we have the following theorem.

Theorem 5: $t_{F\min}(n) = 3(n + 1)$ holds for the FSSP for number pattern 0.

4.10 The Minimal Time Solution for Number Pattern 0

Though the minimal time to solve the FSSP for number pattern 0 is $3(n + 1)$, we show there exists no such solution.

Lemma 6: Let $\mathcal{A} = (\mathcal{S}, \delta)$ be a solution of the FSSP for number pattern 0. If $2(n + 1) > |\mathcal{S}|^2$, $t_F(n, \mathcal{A}) > 3(n + 1)$ holds.

The proof of lemma 6 is also similar to the proof of lemma 2. Thus, the following theorem holds.

Theorem 6: There exists no minimal time solution of the FSSP for number pattern 0.

4.11 The Firing Squad Synchronization Algorithm for Number Pattern 1

As the joint cell C_E of number pattern 1 cannot be distinguished from the other cells in the segments with its neighborhood, the FSSP for this pattern is just as the original FSSP of length $2n+3$. Therefore, applying the minimal time algorithm, it can be solved in $4(n + 1)$ steps. It goes without saying that $4(n + 1)$ is the minimal time for this problem.

5. Conclusion

In this paper, from a theoretical interest, we extended the FSSP to number patterns on a seven-segment display. Generalizing the FSSPs for some number patterns, we defined segment trees and proposed the FSSP for segment trees. The FSSP for segment trees can be reduced to that for a one-dimensional array divided evenly by joint cells. We called such an array as segment array and also proposed the FSSP for those segment arrays. Moreover, we developed algorithms to solve the FSSPs for segment arrays and other number patterns, respectively.

Our algorithm to solve the FSSP for a segment array of length $h(n+1)+1$ requires $h(n+1)+\frac{1}{\alpha}(n+1)$ steps for some natural number α. Therefore, the FSSP for number patterns 4, 8 and 9, number patterns 3, 5 and 6, and number pattern 2 can be solved by this algorithm in $3(n + 1) + \frac{1}{\alpha}(n + 1)$ steps, $4(n + 1) + \frac{1}{\alpha}(n + 1)$ steps, and $5(n + 1) + \frac{1}{\alpha}(n + 1)$ steps, respectively. We then clarified that the minimal time of the FSSP for segment arrays is $h(n + 1)$ and showed that there exists no such solution.

We showed the algorithms to solve the FSSP for number patterns 0 and 7, which require $3(n+1)+\frac{1}{\alpha}(n+1)$ steps. We then clarified that the minimal time of the FSSP for number patterns 0 and 7 is $3(n + 1)$ and showed that there exists no such solution.

The FSSP for number pattern 1 is just as the original FSSP of array length $2n + 3$. Therefore, it can be solved in $4(n + 1)$ steps by the minimal time algorithm.

Acknowledgments

The authors are grateful to Mr. Takemasa and Mr. Asai (formerly of University of Toyama) for giving us a hint about the problem discussed in this paper, and to Prof. Okawa (University of Aizu) and Prof. Osato (Osaka Institute of Technology) for giving us valuable suggestions and advice.

References

[1] E.F. Moore, "The firing squad synchronization problem," in Sequential Machines, Selected Papers, ed. E.F. Moore, pp.213–214, Addison-Wesley, Reading MA., 1964.

[2] M. Minsky and J. McCarthy, Computation: Finite and Infinite Machines, pp.28–29, Prentice Hall, 1967.

[3] E. Goto, "A minimal-time solution of the firing squad synchronization problem," Course Notes for Applied Mathematics 298, Harvard University, pp.52–59, 1962.

[4] A. Waksman, "An optimum solution to the firing squad synchronization problem," Information and Control, vol.9, pp.66–78, 1966.

[5] R. Balzer, "An 8-state minimal time solution to the firing squad synchronization problem," Information and Control, vol.10, pp.22–42, 1967.

[6] H.D. Gerken, "Über synchronisations-probleme bei zellularautomaten," Diplomarbeit, Institut für Theoretische Informatik, Technische Universität Braunschweig, pp.1–50, 1987.

[7] J. Mazoyer, "A six states minimal time solution to the firing squad synchronization problem," Theor. Comput. Sci., vol.50, pp.183–238, 1987.

[8] K. Culik, "Variations of the firing squad problem and applications," Inf. Process. Lett., vol.30, pp.153–157, 1989.

[9] I. Shinahr, "Two and three-dimensional firing squad synchronization problems," Information and Control, vol.24, pp.163–180, 1974.

[10] K. Kobayashi, "The firing squad synchronization problem for two dimensional arrays," Information and Control, vol.34, pp.153–157, 1977.

[11] M. Hisaoka, H. Yamada, M. Maeda, T. Worsch, and H. Umeo, "A design of firing squad synchronization algorithm for multi-general problems and their implementations," IEICE Technical Report, COMP2002-133, 2003.

[12] M. Hisaoka, H. Yamada, M. Maeda, T. Worsch, and H. Umeo, "An optimum-time firing squad synchronization algorithm for multi-general problem and its implementation," IEICE Technical Report, COMP2003-18, 2003.

[13] H. Schmid and T. Worsch, "The firing squad synchronization problem with many generals for one-dimensional CA," IFIP TCS 2004, pp.111–124, 2004.

[14] Z. Roka, "The firing squad synchronization problem on Cayley graphs," Proc. MFCS'95, Prague, Czech Republic, 1995. Lect. Notes Comput. Sci., vol.969, pp.402–411, 1995.

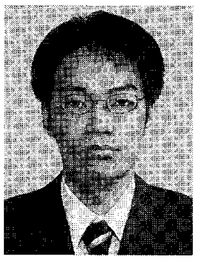
[15] Y. Nishitani and N. Honda, "The firing squad synchronization problem for graphs," Theor. Comput. Sci., vol.14, pp.39–61, 1981.

[16] K. Imai and K. Morita, "Firing squad synchronization problem in reversible cellular automata," Theor. Comput. Sci., vol.165, pp.475–482, 1996.

[17] K. Imai, K. Morita, and K. Sako, "Firing squad synchronization problem in number-conserving cellular automata," Proc. IFIP Workshop on Cellular Automata, Santiago, Chile, 1998.

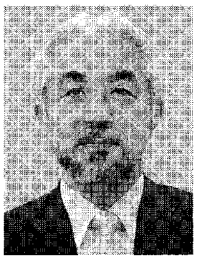
[18] J. Mazoyer, "On optimal solutions to the firing squad synchronization problem," Theor. Comput. Sci., vol.168, no.2, pp.367–404, 1996.

[19] H. Umeo, "A design of cellular algorithms for 1-bit inter-cell communications and related cellular algorithms," Proc. MCU'98, vol.1, pp.210–227, 1998.

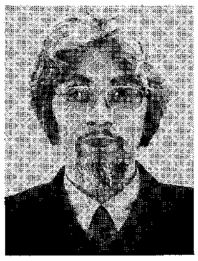


Kazuya Yamashita received B.E. and M.E. degrees in Intellectual Information Engineering from University of Toyama in 2004 and 2006, respectively. Since 2006, he has been with University of Toyama where he is presently an assistant of the Faculty of Engineering. His research interests include formal language theory, automaton theory and algorithms. He is a member of IPSJ.

Mitsuru Sakai received B.E. and M.E. degrees in Electronics and D.E. degree in Systems Engineering from University of Toyama in 1979, 1981 and 2000, respectively. Since 1982, he has been with University of Toyama where he is presently an associate professor of the Faculty of Engineering. His research interests include pattern recognition and image processing. He is a member of IPSJ and JSAI.



Sadaki Hirose received B.E. degree in Electronics from University of Toyama in 1974, and M.E. and D.E. degrees in Information Engineering from Tohoku University in 1976 and 1980, respectively. From 1980 to 1984, he was with Fujitsu Laboratories, Ltd., and from 1984 to 1989, he was an associate professor at Kanagawa University. In 1989, he joined University of Toyama where he is presently the dean of the Faculty of Engineering. His research interests include formal language theory, automaton theory, algorithms and complexity theory. He is a member of IPSJ.



Yasuaki Nishitani received B.E. degree in Electronics and M.E.and D.E. degrees in Computer Science from Tohoku University in 1975, 1977 and 1984, respectively. From 1981 to 1987, he was with Software Product Engineering Laboratory, NEC Corporation, and from 1987 to 2000, he was an associate professor at Gunma University. In 2000, he joined the Iwate University where he is presently a professor of the Faculty of Engineering. His research interests include switching theory, software engineering and distributed algorithms. He is a member of IPSJ.