THE FIRING SQUAD SYNCHRONIZATION PROBLEM WITH MANY GENERALS FOR ONE-DIMENSIONAL CA

Hubert Schmid, Thomas Worsch
IAKS Vollmar, Fakultät für Informatik
Universität Karlsruhe, Germany
worsch@ira.uka.de

Abstract The Firing Squad Synchronization Problem is one of the classical problems for cellular automata. In this paper we consider the case of more than one general. A synchronous and an asynchronous version of the problem are considered. In the latter case the generals may start their activities at different times. In the synchronous case there are optimum-time solutions. Very simple and elegant techniques for constructing one of them are the main contribution of this paper on the algorithmic side. For the asynchronous case an exact formula for the optimum synchronization time of each instance is derived. We prove that no CA can solve all instances in optimum time, but we describe a CA whose running time is very close to it; it only needs additional $\log n$ steps.

Keywords: Cellular automata, Firing Squad Synchronization Problem

Introduction

The *Firing Squad Synchronization Problem* (FSSP) is one of the most well studied algorithmic problems for cellular automata. Proposed by Myhill in 1957, first solutions date back at least to the early sixties [2].

With a few exceptions there are mainly two types of results. Part of the research is concerned with the task to find CA with as few states as possible which still solve the problem (possibly in optimum time) [7]. In other papers modifications and generalizations of the classical FSSP are investigated. The present paper is of the second type.

It is organized as follows. In Section 1 we quickly review the basic definitions of cellular automata and then proceed to describe the generalized synchronization problems we are interested in. In Section 2 an exact formula for the optimum time of any instance of the asynchronous multi-general FSSP is de-

rived and it is shown that no CA solving the problem in general can achieve this time for all instances. Concrete CA algorithms for the synchronous and the asynchronous multi-general FSSP are the topic of Sections 3 and 4 respectively.

The results presented are part of the diploma thesis of the first author [10].

1. Basic notions

For two sets A and B we write B^A for the set of all functions $f : A \to B$. The set of integers will be denoted by $\mathbf{Z}$, the set of positive integers by $\mathbf{N}$ and the set of nonnegative integers by $\mathbf{N}_0$.

1.1 Cellular automata

We assume that the reader is familiar with the standard model of one-dimensional CA with von Neumann neighborhood of radius 1, that is we use neighborhood $N = \{-1, 0, 1\}$. We will denote by S the finite set of states and by $\delta : S^N \to S$ the local transition function.

A global configuration is a mapping $C \in S^{\mathbf{Z}}$. Given a configuration C and a cell $i \in \mathbf{Z}$ we write $C(i + N)$ for the local configuration observed by cell i which is defined as $C(i + N) : N \to S : n \mapsto C(i + n)$. The local transition function induces the global transition function $\Delta : S^{\mathbf{Z}} \to S^{\mathbf{Z}}$ as usual: For a configuration $C : \mathbf{Z} \to S$, its successor configuration $\Delta(C)$ is defined by the requirement that for all $i \in \mathbf{Z}$ one has: $(\Delta(C))(i) = \delta(C(i + N))$. If C^0 is a configuration we sometimes abbreviate $\Delta^t(C^0)$ as C^t.

We will assume that there is always a designated quiescent state $\mathtt{q}$, which for the rest of this paper is in fact even a "dead" state in the sense that $C(i) = \mathtt{q} \Longrightarrow (\Delta(C))(i) = \mathtt{q}$.

1.2 Firing squad synchronization problems

The standard FSSP. The standard formulation of the FSSP requires the existence of a designated state $\mathtt{g}$ for the "general", a designated state $\mathtt{s}$ for the "soldiers" and designated "firing" state $\mathtt{f}$.

The task is to find a CA (S, δ) with $\{\mathtt{q}, \mathtt{g}, \mathtt{s}, \mathtt{f}\} \subseteq S$ such that:

- $\delta(\ell) = \mathtt{q}$ for all local configurations such that $\ell(0) = \mathtt{q}$;
- $\delta(\ell) = \ell(0)$ for all $\ell : N \to \{\mathtt{q}, \mathtt{s}\}$, i.e. cells in state $\mathtt{s}$ don't start any activities "by themselves".

These conditions are *always* required. We will not list these requirements again, but it is to be understood that the CA for the generalized FSSPs considered below have to fulfill them, too.

- For any $n \in \mathbf{N}$ let C_n^0 denote the configuration

$$C_n^0(i) = \begin{cases} \mathtt{q} & \text{iff not } 0 \le i < n \\ \mathtt{g} & \text{iff } i = 0 \\ \mathtt{s} & \text{otherwise, i.e. iff } 1 \le i < n \end{cases}$$

Then for each n there has to be a $t \in \mathbf{N}_0$ such that the CA *fires* after t time steps when started with initial configuration C_n^0, i.e. all initially non-quiescent cells enter the firing state after the same number of steps for the first time:

- for all $0 \leq i < n$: $C_n^t(i) = \mathtt{f}$ and
- for all $0 \leq i < n$ and for all $t' < t$: $C_n^{t'}(i) \neq \mathtt{f}$.

In this case one problem instance is completely characterized by the number n of cells to be synchronized.

There are several generalizations and modifications which have been considered in the literature. These include different types of underlying "geometries", e.g. [1], the inclusion of "faulty" cells, e.g. [13], synchronization in a prescribed but non-optimum time, e.g. [6] and others. In this paper we are interested in the case of more than one general.

The synchronous multi-general FSSP. For a generalized problem which already has been investigated and solved quite some time ago, there is still one general, but its position is not known. Each problem instance is then characterized by the number n of cells and the position p, $0 \leq p < n$, of the general. The first optimum time solution for this problem is due to Moore [8]. In Section 3 we will describe a different approach which basically allows to apply any solution developed for the standard FSSP also in this generalized case.

In the present paper the restriction of having exactly one general is dropped. In the simpler case each problem instance is characterized by the number n of cells, a number $k \leq n$ of generals and arbitrary initial positions $0 \leq p_i < n$, $1 \leq i \leq k$, of the generals. In other words the initial configurations look like this:

$$C_n^0(i) = \begin{cases} \mathtt{q} & \text{iff not } 0 \leq i < n \\ \mathtt{g} & \text{iff } i \in \{p_1, \ldots, p_k\} \\ \mathtt{s} & \text{otherwise} \end{cases}$$

We call this the *synchronous multi-general FSSP*, abbreviated as S-MG-FSSP. First ideas for its (optimum-time) solution have been sketched by Hisaoka et al. [3]. In Section 3 we will present a slightly different approach.

The asynchronous Multi-General FSSP. Assume that one wants to construct the composition of two CA with local rules δ_1 and δ_2 in the following sense. Initially all cells use δ_1. After some time some cells will observe certain local configurations indicating that a first sub-goal of the algorithm has been reached and that now all (non-quiescent) cells should switch to δ_2 simultaneously. In some applications of this method of constructing new CA form old ones, different cells will note at *different* times, that the mode of operation should be switched.

Thus, what is really useful is yet another generalization of the FSSP where possibly several generals start their work at different times. We call this the *asynchronous multi-general FSSP* (where the adjective refers to the asynchronous start of the generals, of course).

In this case each problem instance $I = (n, T)$ is characterized by the number n of cells (called the *length* $\mathrm{len}(I)$ of I), a number $k \leq n$ of generals and a set of pairs $T = \{(p_1, t_1), \ldots, (p_k, t_k)\}$, where the p_i, $0 \leq p_i < n$, are the arbitrary initial positions of the generals and each $t_i \geq 0$ is the point in time when a general "appears" at position p_i and starts to work.

We formalize this as follows. The initial configuration is

$$C_n^0(i) = \begin{cases} \mathtt{q} & \text{iff not } 0 \leq i < n \\ \mathtt{s} & \text{iff } 0 \leq i < n \end{cases}$$

Given a configuration C_n^t its successor configuration C_n^{t+1} is determined in two phases:

- First, for each p_i with $(p_i, t) \in T$ the state of cell p_i in configuration C_n^t is set to $\mathtt{g}$.
- To the resulting configuration then the global transition function is applied.

This description is not a CA. But in applications the $(p_i, t_i) \in T$ are not some "external events" but indeed result from a CA which produces certain specific local configurations observed by cells p_i at times t_i.

In order to avoid problems (which do not occur in applications anyway) we slightly restrict the set of allowed problem instances. A general must not appear in cell p_i at time t_i if that might already have entered a state different from $\mathtt{s}$. That is for any two different $(p_i, t_i) \in T$ and $(p_j, t_j) \in T$ must hold: $|t_j - t_i| < |p_j - p_i|$.

We call this the *asynchronous multi-general FSSP*, abbreviated as A-MG-FSSP. Vollmar [14, 15] has described CA solving this problem but did not investigate questions concerning the optimum time. In general Vollmar's solutions are considerably slower than the solution described below in Section 4.

1.3 Optimum time for the S-MG-FSSP and the A-MG-FSSP

Let **SP** be a solvable synchronization problem with a set $\mathcal{I}$ of instances and $f : \mathcal{I} \to \mathbf{N}$ a function. A CA A solves **SP** if for each $I \in \mathcal{I}$ as the initial configuration the CA A eventually fires. The number of steps needed for this is denoted as $T_A(I)$. We say that f is a *lower bound* for **SP** if for each CA A solving **SP** holds: $\forall I \in \mathcal{I} : f(I) \leq T_A(I)$.

If $\mathcal{A}$ denotes the set of CA solving **SP**, it is easy to see [4] that the function $T_{\min} : \mathcal{I} \to \mathbf{N}$ defined by $T_{\min}(I) = \min\{T_A(I) \mid A \in \mathcal{A}\}$ is a lower bound for **SP** and in fact is the greatest lower bound. We call $T_{\min}$ the *optimum time* for **SP**.

Note, that this definition is somewhat non-uniform, because for different I it may be that $T_{\min}(I)$ can only be achieved by different CA.

It is one of the surprises of the classical FSSP, that its optimum time can be achieved by one CA for all instances. But since one is accustomed to that it will probably come as an even greater surprise, that *each CA solving the asynchronous multi-general FSSP must solve infinitely many instances in a time which is not optimum time.* See Theorem 5 below.

Informally $T_{\min}(I)$ can be described easily for all synchronization problems considered in this paper: The optimum time for instance I is the time needed so that the leftmost and the rightmost cell can send some "information" to the other end (i.e. the rightmost or the leftmost cell respectively). This consists of $n-1$ steps needed to transmit the information plus the number of steps needed before the border cells can enter a state different from `s` for the first time. For example in the standard problem setting it takes $n-1$ steps, before a signal sent by the general at the left end has reached the rightmost cell $n-1$, resulting in an optimum time of $2n-2$.

In the asynchronous multi-general case from the general at position p_i it takes p_i steps to reach the leftmost cell 0 and $n-1-p_i$ steps to reach the rightmost cell $n-1$. This should make it plausible that one has

$$T_{\min}(I) = n - 1 + \max\left\{\min_i(t_i + p_i),\ \min_i(t_i + n - 1 - p_i)\right\} .$$

This is the first main result which we are going to prove now.

2. Optimum time for the multi-general FSSPs

THEOREM 1 *The optimum time for an instance I of the asynchronous multi-general FSSP with n cells and $T = \{(p_1, t_1), \ldots, (p_k, t_k)\}$ is*

$$T_{\min}(I) = n - 1 + \max\begin{cases}\min_i(t_i + p_i)\\ n - 1 + \min_i(t_i - p_i)\end{cases}$$

Proof. We split the proof in two parts. First we show (Lemma 2) that the time given above is indeed a lower bound for the running time of CA solving the A-MG-FSSP. In Lemma 3 we show that for each I there is a CA solving the problem and needing only time $T_{\min}(I)$ for I. □

LEMMA 2 *Let A be any CA which solves the asynchronous multi-general FSSP and let I be any of its instances. Then*

$$T_A(I) \geq T_{\min}(I) = n - 1 + \max\begin{cases}\min_i(t_i + p_i)\\ n - 1 + \min_i(t_i - p_i)\end{cases}$$

Proof. The proof is by contradiction and similar to the one for the standard FSSP. Let A be any CA solving the problem and assume that I is an instance such that A needs time $t_f = T_A(I) < T_{\min}(I)$ for the synchronization. Assume that for I one has $\min_i(t_i + p_i) \leq \min_i(t_i + n - 1 - p_i)$ (the other case can be treated analogously) and let j be an index such that $t_j - p_j$ becomes minimum.

Let $D = \{(x,t) \mid 0 \le x < n \wedge t \ge 0 \wedge x + t \le t_f\}$ denote the set of all points in the space-time diagram for instance I which "have an influence" on cell 0 at time t_f. In particular for cell $n-1$ one has $D \cap \{n-1\} \times \mathbf{N}_0 = \{(n-1,t) \mid 0 \le t \le t_f - (n-1)\}$. But $t_f - (n-1) < T_{\min}(I) - (n-1) = n - 1 + t_j - p_j$ and the latter is the first time when cell $n-1$ could possible have left state `s`.

Therefore for all t such that $(n-1,t) \in D$ we have $C_n^t(n-1) = \mathtt{s}$.

Now consider a new instance I' which consists of $n' = n + t_f + 1$ cells and the same set T as I. While in instance I cell $n-1$ always was in state `s` during the first $t_f - (n-1)$ steps because it had `q` as its right neighbor, in instance I' cell $n-1$ will always be in state `s` during the first $t_f - (n-1)$ steps because it has `s` as its right neighbor and again no signals can have reached it until then.

It is an easy exercise to show that as a consequence for all $(x,t) \in D$ the states in the space-time diagrams at positions (x,t) coincide for the instances I and I'. In particular for instance I' cell 0 will enter state `f` at time t_f. But *at the same time* cell $(n'-1)$ will still be in state `s` because it is t_f cells to the right of cell $(n-1)$ and hence cannot have been reached by any signal. But this is in contradiction to the requirement that always all cells have the enter state `f` simultaneously. □

LEMMA 3 *For each instance I of the asynchronous multi-general FSSP there is a CA A solving the problem for all instances and needing only $T_A(I) = T_{\min}(I)$ for instance I.*

Proof. Let I be any instance and denote by n the number of non-quiescent cells. Let A be any CA solving the A-MG-FSSP. It is clear that such CA exist. For example one can send a signal to the left from any general. As soon as the first signal arrives at the leftmost cell that one becomes the "real" general and starts an algorithm for the standard FSSP erasing all signals coming from the right.

Below we will describe a CA A_n which synchronizes all instances with $m \le n$ cells, and these in optimum time. For instances with more than n cells the CA either never fires any cell or does fire them synchronously.

Given A_n and A one can construct a new CA A' by running those two in parallel. By definition A' fires as soon as either A or A_n would fire. A' solves the A-MG-FSSP and only needs optimum time in particular for instance I.

A_n works as follows. Each general sends a signal to the left and one to the right. Whenever two such signals meet, they erase each other. When a signal arrives at the left or right border, a counter is initialized with 0. The counter then moves to the opposite side with speed 1 and is incremented by 1 in each step. There are two possibilities:

- A counter signal reaches value n before meeting the counter signal coming from the other end. Then the counter is replaced by a state "∞", which is not the firing state and which spreads to all cells.

- The counter signals meet before they enter state ∞. Then the maximum of their values is taken and propagated to all cells with speed 1. Further-

more all cells decrement the value by 1 in each step. When 0 is reached, the cells fire.

In Figure 1 the algorithm is sketched for two instances of length 12; on the left side $T = \{(1,1),(8,0)\}$ and on the right $T = \{(1,0),(8,0)\}$. □

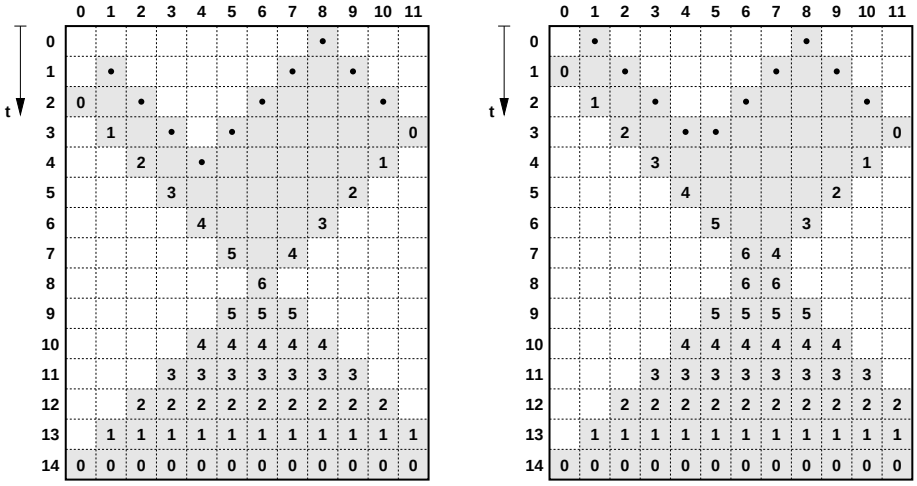

Figure 1. Solving instances of the multi-general FSSP with bounded size using a counter of finite size.

The question now is whether optimum time can be achieved for all instances by a single CA. In the synchronous case all $t_i = 0$; hence for the S-MG-FSSP holds:

$$T_{\min}(I) = n - 1 + \max\{\min_i p_i,\ n - 1 - \max_i p_i\} .$$

It follows from the algorithm in Section 3 that indeed one has:

THEOREM 4 *There exists a CA which solves the synchronous multi-general FSSP in optimum time for all instances.*

This has to be contrasted with the result we are going to prove now:

THEOREM 5 *For each CA with k states solving the asynchronous multi-general FSSP there are infinitely many instances I such that*

$$T_A(I) \geq T_{\min}(I) + \lfloor \log_k \mathrm{len}(I) \rfloor ,$$

i.e. for which the CA does not achieve optimum synchronization time.

The main building block for the proof is the following lemma:

LEMMA 6 *Let A be a CA with k states solving the asynchronous multi-general FSSP. Then for each n there is an instance I of length n such that*

$$T_A(I) \geq T_{\min}(I) + \left\lfloor \log_k \frac{n}{2} \right\rfloor .$$

Proof. If $\lfloor \log_k \frac{n}{2} \rfloor \leq 0$, the statement is trivially true. Let us therefore assume that $\lfloor \log_k \frac{n}{2} \rfloor \geq 1$, i.e. $n \geq 2k$. Let $m = \lfloor \frac{n}{2} \rfloor$ and $w = \lfloor \log_k m \rfloor$. Since k is an integer, $w = \lfloor \log_k \frac{n}{2} \rfloor$ and hence $k^w \leq \frac{n}{2}$.

Let A be a CA with k states solving the asynchronous multi-general FSSP. We claim that the instance I with $T = \{(0,0),(n-1,0)\}$ has the required property. We will write t_f as an abbreviation for $T_A(I)$.

Denote by s_j the list of w states $C_n^j(j)C_n^{j+1}(j)\cdots C_n^{j+w-1}(j)$ in which a cell j, $0 \leq j < n/2$, is in w subsequent steps starting at time j. Since A has k states, there are only $k^w \leq n/2$ pairwise different lists s_j. Therefore the sequence $s_0, s_1, \ldots$ becomes periodic before there is any influence from the rightmost general. Let d denote a multiple of the period length such that $d \geq w$ and consider the instance I' of length $n' = n+d$ with $T' = \{(0,0),(n-1+d,d)\}$. It follows that $s_{n-1} = s_{n'-1}$.

Now assume by contradiction that $t_f = T_A(I) < T_{\min}(I) + \lfloor \log_k \frac{n}{2} \rfloor = T_{\min}(I)+w$. This means that s_{n-1} contains a firing state at position $t_f-(n-1)$. Therefore $s_{n'-1}$ contains a firing state, too. Since A is assumed to solve the asynchronous multi-general FSSP, it must in fact fire for instance I' at time t_f+d. But $t_f+d < T_{\min}(I)+w+d = n-1+w+d \leq n-1+d+d = n'-1+d$.

Similar to the proof of Lemma 2 this means that in particular cell 0 fires at a time when it cannot have been influenced by cell $n'-1$. Therefore increasing the length of I' to t_f+d+2 (and keeping the same T') one gets an instance I'' for which A would fail, because after t_f+d steps cell 0 would again enter a firing state but the rightmost cell would still be in state $\mathtt{s}$. □

It is now straightforward to finish this section:

Proof (of Theorem 5). It is known that even for the standard FSSP one needs at least $k \geq 5$ states [9]. For any w choose $n = k^w - 1$ and hence $\lfloor \log_k n \rfloor = w-1$. But

$$\frac{n}{2} = \frac{1}{2}(k^w - 1) \geq \frac{1}{2}(k^w - k^{w-1}) = \frac{1}{2}(k-1)k^{w-1} \geq k^{w-1}$$

and therefore $\lfloor \log_k \frac{n}{2} \rfloor \geq w - 1 = \lfloor \log_k n \rfloor$. □

3. A solution for the S-MG-FSSP

On the left side of Figure 2 an algorithm for the one-general case is depicted [16]. Compared to other solutions [8, 12] it has the advantage that any algorithm for the standard FSSP can be "plugged into" the scheme.

ALGORITHM 7 The general sends signals to both borders with speed 1. Upon arrival the border cells start a standard FSSP algorithm. The signals are reflected and meet at a point X of the space-time diagram (or at two cells at the same time; this case can be handled with the usual techniques). The left border cell starts synchronizing the segment to the left of X, the right border cell starts synchronizing the segment to the right of X, both using any algorithm for the standard FSSP.

In general there will be a longer and a shorter segment, which can be distinguished depending on whether a reflected initiation signal passes the general

(point G in the space-time diagram) before meeting the other one at X or not. While the synchronization of the longer segment ends at the optimum time for the whole instance, synchronization for the shorter segment would end too early. This problem is corrected by sending two signals to the shorter segment with speed 1:

- The first is initiated at point X in the space time and "freezes" the synchronization algorithm.
- The second is "thawing" it again. That signal is triggered when another signal which is started at G and which runs with speed $1/2$ arrives at X.

The area of the space-time diagram which is frozen is shown in gray in the left part of Figure 2. A straightforward calculation yields, that as a result the synchronization for the shorter segment will end at the same time as for the longer segment.

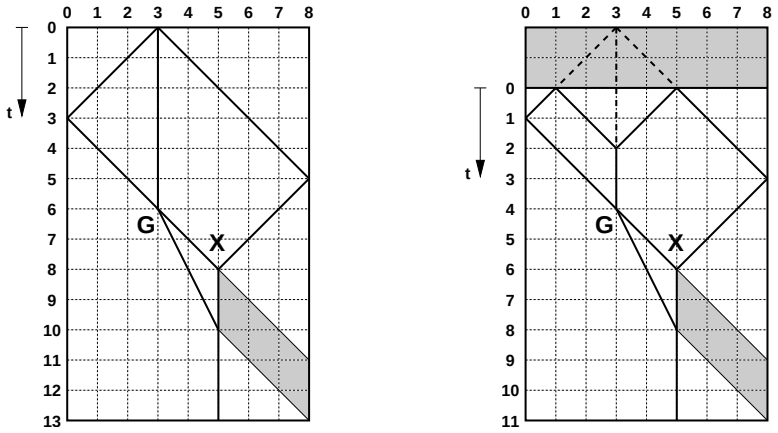

Figure 2. Synchronization using one and two generals at arbitrary positions.

An easy modification of this algorithm can be used for instances with two generals. The cell where a reflected initiation signals marks point G is determined by the meeting point of the not-yet reflected initiation signals. The right part of Figure 2 shows an example. From some time on the space-time diagram coincides with the one resulting from the case with one general at cell G. □

The transition from 2 generals to k generals can again be done in a generic way. We will briefly sketch how this can be achieved.

In the S-MG-FSSP the optimum time is $n - 1 + \max\{\min_i p_i,\ n - 1 - \max_i p_i\}$. This value is determined by the extremal positions $p_l = \min_i p_i$ and $p_r = \max_i p_i$ of generals, independently of the others. The following algorithm exploits this fact and makes sure that each cell will work after a certain number of steps as if only the outermost generals were present.

ALGORITHM 8 Assume that $A = (S, \delta)$ is a CA solving the FSSP for 2 generals with the usual states `s` and `g`. For the new CA $A' = (S', \delta')$ we write `s'` and `g'` for its soldier and general state. It basically uses 3 registers R_l, R_r and R_m. We will first describe the use of R_l. The use of R_r is analogous to R_l, but preferring information from the opposite direction (right instead of left). At last R_m will be explained.

Register R_l consists of a bit indicating the presence (`*`) or absence (`-`) of a signal and a state $x \in S$. Initially $(*, \mathtt{g})$ is induced here by state `g'` of A'. Using algorithm *LeftChoice* described below it is made sure that the state $x \in S$ of R_l of a cell always holds the state the corresponding cell of A would have as long as that only has been influenced by the leftmost of those generals which may have had an influence on it.

This can be realized by sending a signal from each general to the right with speed 1. In the left part of Figure 3 the signals are indicated by stars `*`. Depending on whether a cell observes a `*` signal arriving from its left neighbor it can act appropriately: Whenever its left neighbor has a `*`, a cell behaves as if itself and its right neighbor are in `s`; otherwise if a cell has `*` itself, it behaves as if its right neighbor is in `s`. In the remaining cases the cell behaves "normally".

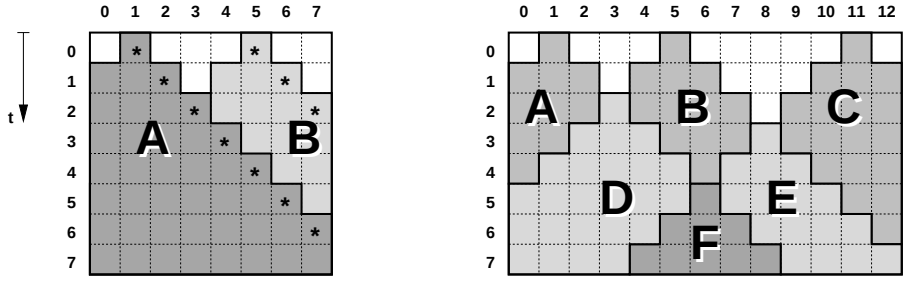

Figure 3. Left: R_l for preferring the left general. Right: R_m for choosing the "currently outermost" generals.

The transition rule for R_l can be described by the following table:

left	center	right	new center
$(*, x_l)$	y_m	y_r	$(*, \delta(x_l, \mathtt{s}, \mathtt{s}))$
$(-, x_l)$	$(*, x_m)$	y_r	$(-, \delta(x_l, x_m, \mathtt{s}))$
$(-, x_l)$	$(-, x_m)$	$(?, x_r)$	$(-, \delta(x_l, x_m, x_r))$

Analogous rules for preferring the right hand side are used for R_r.

It is now easy to use register R_m to simulate the behavior of the cells as it would happen whenever the state in R_l is from the leftmost general and the state in R_r is from the rightmost general: For R_l call a cell q in the neighborhood of a cell p *relevant* for p, iff register R_l of cell q is really used (see the rule table above); similarly for R_r. The new state of R_m of a cell p is computed from 3 states z_{p-1}, z_p and z_{p+1} each stemming from one of the registers of cells $p-1$, p and $p+1$ respectively. For $i \in \{p-1, p, p+1\}$ state z_i is chosen as follows:

If i is relevant for *both*, R_l and R_r (of cell p), then z_i is taken from register R_m if cell i. If i is relevant for exactly one of R_l and R_r (of cell p), then z_i is taken from *that* register of cell i. Otherwise a quiescent state is used.

As a result of these rules one gets for example the situation depicted the right part of Figure 3. Denote by A, B and C also the generals in the corresponding parts of the space-time diagram. Then in part D of the diagram the R_m registers of the cells "behave" as if there were only generals A and B and in part E they behave as if there were only generals B and C. And in part F all states of registers R_m are the same as for the case when there are only generals A and C.

Thus an instance is fired after the same number of steps needed for the simpler instance where all generals except the leftmost and rightmost one are deleted, which is the optimum time (see the remark on the S-MG-FSSP right before Theorem 4). □

4. A solution for the A-MG-FSSP

We remind the reader of the CA A_n which were used in the proof of Lemma 3. A_n is able to fire all instances of A-MG-FSSP of length $m \leq n$ in optimum time.

Below we describe the main aspects of a CA which solves any instance I of A-MG-FSSP in time $T_{\min}(I) + \log n$. In the light of Theorem 5 this is "close to optimal". The idea is to generalize the A_n to a CA A_∞ where there is no upper bound for the counters. Of course unbounded contents of counters cannot be stored in a single cell. Instead we use segments consisting of $\log_b x$ cells to represent a value x in b-adic representation. A well-known technique introduced by Vollmar [14] and used in several contexts [5, 11] are counters with the following properties:

- The initial value of the counter is 0, stored as a single digit.
- In each step the least-significant digit of the counter moves from one cell to next one, the other digits are following.
- In each step the content of the counter is incremented by 1 in such a way that the sequence of counter digits passing a cell c forms the representation of the distance of this cell from the one where the counter started.
- When incrementing a counter an overflow at the currently most-significant digit may happen. In this case the length of the counter is increased by 1 at the end.

Analogously, given a counter with a value $x > 0$ it is possible to decrement it in each cell while moving. But in this case the *length* of the counter is *not* decreased (otherwise speed 1 would be impossible for the counters). Instead zeroes at the most-significant positions are used. Such counters are used in the following CA algorithm.

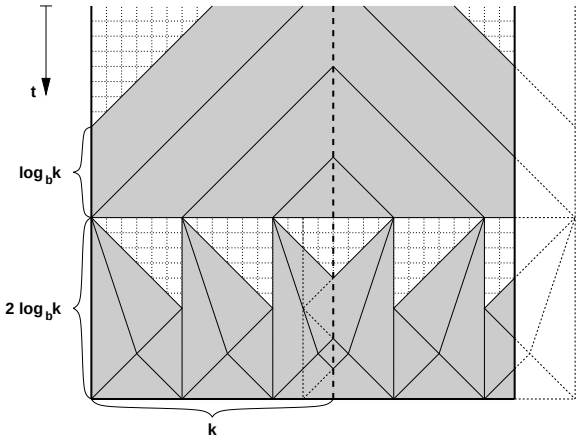

Figure 4. Solving instances of the multi-general FSSP with distributed counters.

ALGORITHM 9 Whenever a general "appears", it sends a signal to the left and one to the right. Whenever two such signals meet, they erase each other. When a signal arrives at the left or right border, a counter as described above is initialized with 0. The counters then move to the opposite sides with speed 1 and are incremented by 1 in each step.

Denote by X the cell where both counters meet. When they arrive at X they are not incremented any longer. Instead their contents are compared digit by digit. This is straightforward, since the least-significant digits arrive first and the other follow step by step. Both counters are kept until it is known which one stores the larger value. This one is used further on, the other one is destroyed.

Let k denote the larger value and K the counter, i.e. the "data structure", storing it. When K arrives at X it starts cycling as follows:

- K is reflected. Its digits move back until the least significant digit meets the most significant one in cell Y, which is $\frac{1}{2}\log k$ cells away from X.
- At Y the counter is reflected again, so that from then on all digits are moving back and forth between X and Y.

After coming back to X for the first time, it is clear that K has stored the larger value.

Now K is decremented in each step. And it is not only cycling, but copies are sent to both sides starting at X. These copies are also decremented in each step. In the upper part of Figure 4 the area where the copies of K are moving is shaded gray. Solid diagonal lines mark the boundaries between two copies.

Additionally each cell checks whether all digits of a counter passing through it are zero. This takes time $\log k$. At some point in time simultaneously some cells discover that this is the case. On both sides of X these cells are $\log k$ cells apart, and their positions are symmetrical with respect to X. The leftmost such cell is cell 0.

These cells start an algorithm for the 1-general FSSP (directed towards X) synchronizing the segment up to the next such cell. This is indicated in the lower part of Figure 4. As can be seen there, problems may arise in two areas:

- At the rightmost cell there may be a segment of unsuitable length. It would be nice if there were a few more cells at the right end as indicated by the dotted lines. Even though the cells are missing, by "folding around" that part of the space-time diagram their actions can easily be simulated in additional registers of the present cells.
- A similar trick can be used at X. The actions of the missing cells which one would have to add to make the lengths of the adjacent segments equal to that of other segments can be simulated in the registers used by the cells between X and Y for the cycling counter.

From the discussion of the above algorithm one immediately gets:

THEOREM 10 *There is a CA which solves any instance I of length n of the A-MG-FSSP in time $T_{\min}(I) + \log_b n$ for some $b \in \mathbf{N}$.*

5. Summary and outlook

Two variants of the generalized FSSP with several generals have been considered for 1-dimensional CA. Using a few simple and elegant techniques, it is possible to design a CA which solves the problem for all instances in optimum time, when all generals start to act synchronously.

When the generals may start their work asynchronously, the optimum time for an instance depends on the starting times, but it can still be computed exactly. Contrary to the case of synchronous generals, there is no longer one CA which can achieve optimum running time for all problem instances. However, we have described a CA whose firing times are quite close to the optimum; it needs only $\log n$ steps longer.

Of course, the problems can also be considered for CA working on 2- or higher-dimensional grids. Even for the synchronous multi-general case the situation becomes considerably more complicated. For example the choice of the neighborhood, e.g. Moore or von Neumann type, not only has an influence on the value of the optimum synchronization times. While for one neighborhood a CA always achieving optimum time is known, for the other neighborhood none has been found until now. This will be discussed in a follow-up paper.

References

[1] K. Čulik and S. Dube. An efficient solution of the firing mob problem. *Theoretical Computer Science*, 91:57–69, 1991.

[2] E. Goto. A minimum time solution of the Firing Squad Problem. Dittoed course notes for Applied Mathematics 298, Harvard University, 1962.

[3] M. Hisaoka, H. Yamada, M. Maeda, Th. Worsch, and H. Umeo. A design of firing squad synchronization algorithms for multi-general problems and their implementations. Unpublished manuscript, 2003.

[4] K. Kobayashi. On the minimal firing time of the firing squad synchronization problem for polyautomata networks. *Theoretical Computer Science*, 7:149–167, 1978.

[5] M. Kutrib and R. Vollmar. Minimal time synchronization in restricted defective cellular automata. *Journal of Information Processing and Cybernetics*, EIK 27:179–196, 1991.

[6] S. La Torre, M. Napoli, and M. Parente. Synchronization of a line of identical processors at a given time. *Fundamenta Informaticae*, 34:103–128, 1998.

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

[8] F. R. Moore and G. G. Langdon. A generalized firing squad problem. *Information and Control*, 12:17–33, 1968.

[9] P. Sanders. Suchalgorithmen auf SIMD-Rechnern — Weitere Ergebnisse zu Polyautomaten. Diploma thesis, Fakultät für Informatik, Universität Karlsruhe, 1993.

[10] H. Schmid. Synchronisationsprobleme für zelluläre Automaten mit mehreren Generälen. Diploma thesis, Fakultät für Informatik, Universität Karlsruhe, 2003.

[11] M. Stratmann and Th. Worsch. Leader election in d-dimensional CA in time $diam \cdot \log(diam)$. *Future Generation Computer Systems*, 18(7):939–950, 2002.

[12] H. Szwerinski. Time optimal solution of the firing squad synchronization problem for n-dimensional rectangles with the general at an arbitrary position. *Theoretical Computer Science*, 19:305–320, 1982.

[13] H. Umeo. A simple design of time-optimum firing squad synchronization algorithms with fault-tolerance. *IEICE Transactions on Information and Systems*, E87-D:733–739, 2004.

[14] R. Vollmar. Yet another generalization of the firing squad problem. Technical report, Technische Universität Braunschweig, Braunschweig, 1976.

[15] R. Vollmar. On two modified problems of synchronization in cellular automata. *Acta Cybernetica*, 3:293–300, 1977.

[16] Th. Worsch. Algorithmen in Zellularautomaten. Course notes, Fakultät für Informatik, Universität Karlsruhe, 2003.