Minimum and non-Minimum Time Solutions to the Firing Squad Synchronization Problem

Margherita Napoli and Mimmo Parente(✉)

Dipartimento di Informatica, Università di Salerno, Salerno, Italy
{napoli,parente}@unisa.it

Abstract. In this paper we present a survey on the minimum and non minimum time solutions to the Firing Squad Synchronization Problem. Particular emphasis is on the contribution given by Jozef Gruska, in honor of which this article is dedicated.

The problem consists in synchronizing a Cellular Automata (CA) whose cells work at discrete steps at unison. The first cell is initially in a particular state, called the *General*, and all the others are in a *Latent* state. The problem is solved when, all the cells enter for the first time and simultaneously a *Firing* state. In its original and basic formulation, the cells are arranged as a line, here we consider also other shapes like rings, rectangular grids and toruses. Also other variations of the problem are considered, such as the limited link capacitiesand different numbers and positions of the General state.

We consider both the minimum time needed to synchronize the CA and some algorithms synchronizing in particular times. Some open problems are also proposed.

Keywords: FSSP · Cellular Automata · Synchronous Computations · Channel Capacity

1 Introduction

Cellular Automata (CA) are perhaps the most intriguing and fascinating model of computation, and Quantum Cellular Automata, perhaps, the most important model of information processing by nature. It is therefore of great interest and large importance to study in depth many natural variants of the basic models of cellular automata and relations among them. CA have been investigated from many points of view and applied in fascinating ways in so many areas of science and technology. One of the first, and still fascinating, problems concerning cellular automata is the Firing Squad Synchronization Problem (FSSP). When and how one can make interaction of isolated and simply interconnected identical

Work partially supported by Italian FARB project grant cod. ORSA124598, 2012. The first author acknowledges also the Italian PRIN project grant 2010-2011 "Logical Methods of Information Management", cod. 2010FP79LR 001.

C.S. Calude et al. (Eds.): Gruska Festschrift, LNCS 8808, pp. 114–128, 2014.
DOI: 10.1007/978-3-319-13350-8_9

finite automata, to cooperate so effectively that they achieve almost impossible tasks, to completely synchronize their actions.[1]

The FSSP was originally proposed by John Myhill in 1957 and printed later in [Moo64]. Roughly speaking, its setting consists of a network of identical cells (finite automata) working synchronously at discrete time steps and connected linearly, see Fig. 1a. The cells are all in a quiescent state until the leftmost, stimulated by the external environment, enters a *General* state and issues a *Fire when ready* command and later all the cells (soldiers) enter simultaneously a *Fire* state.

In his seminal paper, Myhill said that at most four hours are needed to solve the FSSP, also by non computer science experts, as reported in [GK12]. However, though this may sound a little intimidating, we think that, as often happens in describing math problems, the simplicity of its formulation, and why not, also the elegance, can lead to this idea. What is certainly true is that the basic solution is quite simple to understand. The next natural step was to find the fastest solution and at the best of our knowledge, as reported by Umeo in [Ume96], Goto gave the first minimum time solution in a course note of Harvard University [Got62], and since then other minimum time solutions have been given, as we will see in in section 3.1.

In the original formulation of the problem, the cells are arranged as a line. We discuss time solutions also when cells are arranged in different shapes, such as ring-shaped or rectangular arrays, and spend few words on higher dimensional arrays. Besides the time, also the *size* of the finite automata is important, in fact processors with few states can function at higher clock rates and the solutions are faster in absolute time (see [GK12]). In section 3 we will discuss also on this aspect, and emphasize some open problems.

The importance of FSSP lies clearly also in the fact that synchronizing algorithms are useful when it is necessary to automatically start at the same moment various activities (e.g. different algorithms). This essentially motivated our studies reported in section 4, concerning the problem of synchronizing a network at given times.

Let us underline that this paper does not pretend to be a complete survey on the FSSP. Actually, in literature there is abundance of material and several beautiful survey papers have been written by Hiroshi Umeo.

The rest of the paper is organized as follows. In section 2 we give the basic definitions of the FSSP and of the different networks we consider. In section 3 we present the minimum time solutions. In the last section 4 we survey some algorithms that synchronize at particular times.

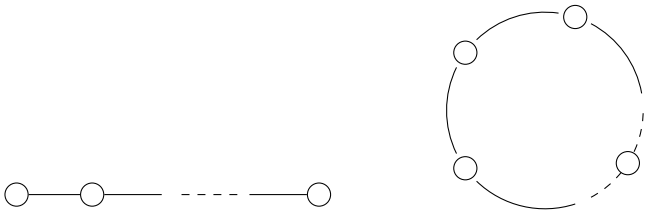
2 The Firing Squad Synchronization Problem

Cellular Automata. A cellular automaton (CA) consists of a regular *network* of *cells*, each in one of a finite number of *states* (Q denotes the set of such states).

[1] This paragraph is the incipit of the introduction written by Jozef in [GLP07], that we chose to quote in full here for the passion about the CA that it contains and still distils.

A *neighborhood* relation is defined, indicating the *neighbors* of each cell. All the cells have the same number N of neighbors, except a fixed number of *boundary cells* which have less neighbors. A boundary cell has less than N neighbors. A cell is intended to be linked to each of its neighbors through *communication channels*, and can send and receive, in each time step, binary sequences whose length is bounded by the capacity of the channels.

Time in the model is discrete. On each time step, every cell updates its state in accordance to a transition rule that takes as input the state of the cell itself and the sequences obtained from the cells in its neighborhood. The cells are then finite automata which operate synchronously, at discrete time unit. At each time t, a *configuration* specifies the state of each cell. A *computation step* modify the configuration, in accordance to the transition function and depending on both the current configuration and the sequences sent by the cells. An *initial configuration* is a configuration at time 1. Observe that in the classical definition of CA, the transition function takes as input the state of the cell itself and those of its neighbors at the previous step. Such classical definition is captured here when the capacity of the channels is $log|Q|$ and then each cell can send its whole state in a step.



(a) Line **(b)** Ring

Fig. 1. One dimensional networks

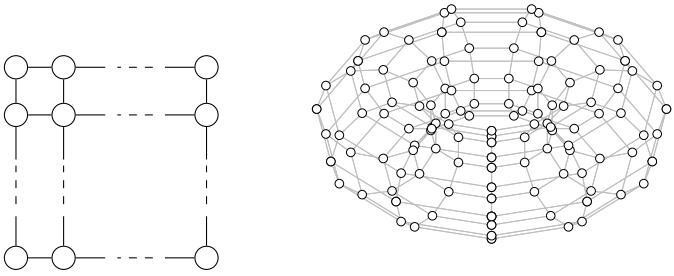
The FSSP Problem. We assume that among the states of the considered cellular automaton three are distinguished states: G, the *General* state, L, the *Latent* state, and F, the *Firing* state. The state L has the property that a cell in this state can send only 0 to its neighbors, and remains in the same state unless it receives a different from 0 sequence from one of its neighbors.

The initial configuration for the FSSP problem is such that a cell in a predetermined position in the network is in the state G, and is called General, and all the others cells, called Latent, are in the state L (in some generalization of the problem, more then one General may occur in the initial configuration). The problem is to determine a description of a CA (state set and transition function), which does not depend on the number of cells, such that, starting from the initial configuration, at some future time, all the cells will simultaneously and, for the first time, enter the firing state F (*synchronization*).

We are interested in the time when the cells enter F, and we express it as a function of a parameter n of the size of the network. (e.g. for a Line or a Ring

n is its length, for a SQUARE it is the number of rows). A cellular automaton which provides a synchronization in time $t(n)$ is also called a *solution in time* $t(n)$ of the FSSP, or simply a *solution*.

Communication Networks. The original FSSP problem was defined over a linear sequence of cells. Later, a number of variations and generalizations of the problem have been introduced. We deal with FSSP defined over different *Communication Networks*: the network may have different shape, number of dimensions[2], and neighborhood relation, and the channels may vary for capacity and for direction of the information flow (either *two-way* or *one-way*).



(a) Square **(b)** Torus

Fig. 2. Two-dimensional networks

- One-dimensional networks:
  - LINE: a linear sequence of cells, with the first and the last cell being the boundary cells. Boundary cells have just one neighbor, the other cells have two neighbors. Channels are two-way, and connect every pair of adjacent cells. The capacity of the channels is $log|Q|$, that is, they have the ability to transmit a state of the CA, (see Figure 1a).
  - RING: a one-dimensional network with two-way communication channels with capacity $log|Q|$. The peculiarity is that there are no boundary cells, and all the adjacent cells are linked to each other, (see Figure 1b). Cells are numbered $1, \cdots, n$, as for a LINE, with the first cell arbitrarily chosen.
  - oneWay-RING: a RING in which the communication channels are *one-way*, that is, they are directed links from cell i to the cell $i+1$, modulo n. Thus, at each step the i-th cell receives the information from the the cell $i-1$, and sends the information to the cell $i+1$.
  - 1-LINE and a 1-RING: similar to the LINE and to the RING, respectively, with communication channels having capacity 1. In this way, they have the ability to transmit just one bit in a time.
- Two-dimensional networks

[2] We use the term *dimension* here with the intended meaning to indicate the number of coordinates needed to locate a cell.

- SQUARE: a regular grid of $n \times n$ cells, numbered (i, j), for $1 \leq i, j \leq n$. Channels, with capacity $log|Q|$, are two-way, and connect every pair of adjacent cells, i.e. each cell (i, j), except for the boundary cells, is connected to cells $(i-1, j)$, $(i, j-1)$, $(i+1, j)$ and $(i, j+1)$ (a boundary cell may have two or three neighbors, depending on the values of i and j) (see Figure 2a).
- RECTANGLE: similar to the SQUARE, but with a grid of $m \times n$ cells.
- TORUS: Simplifying, it is obtained as a rotation of rings and can be seen as a two-dimensional regular grid of $n \times n$ cells, without boundary cells: as shown in Figure 2b, every cell has four neighbors.
- oneWay-TORUS: similar grid to the TORUS, with one-way channels from each cell (i, j) to the cells $(i, j + 1)$ and $(i + 1, j)$, modulo n.
- 1-SQUARE and 1-TORUS: SQUARE and TORUS with one-bit communication channels.

In a natural way, these definitions can be extended to communication networks with higher dimensions.

As already said, in the initial configuration for FSSP, a GENERAL is a cell in the state G. In the above described networks, the GENERAL is the first cell in the grid, that is the cell 1 in the one-dimensional networks and the cell $(1, 1)$ in the two-dimensional cases. Several generalizations of the problem have been considered in literature, by modifying the number or the position of the GENERALs in the initial configuration. Let us mention some of them.

- *Two-End FSSP*, also called *Two-End Synchronization*: the network is a LINE of n cells, and the synchronization starts from an initial configuration with a GENERAL at each end (i.e., at cells 1 and n).
- *Four-End FSSP*: the network is a SQUARE of $n \times n$ cells, and the synchronization starts from an initial configuration with GENERALs at the four corners (i.e., at cells $(1, 1)$, $(1, n)$, $(n, 1)$ and (n, n)).
- *Generalized FSSP* (GFSSP): the GENERAL is located on an arbitrary cell.
- *Multi-General FSSP* (MG-FSSP): in the initial configuration there is an arbitrary number of GENERALs, in arbitrary positions.
- *Asynchronous Multi-General FSSP* (A-MG-FSSP): a generalization of the MG-FSSP, in which the GENERALs start their work in an asynchronous way.

3 Minimum Time Solutions to the FSSP

In this section we present minimum time synchronization for several models of communication networks.

3.1 One-Dimensional Networks

Let us first consider one-dimensional models. We will discuss the optimum time for each variant of the problem and show that in almost all the considered cases, optimum time solutions exist.

Line. It is well known that any solution to FSSP over a LINE of n cells requires at least time $2n - 1$. Actually, the time required for the GENERAL to wake up

all the LATENT cells is n, and other $n-1$ steps need to get back the information that all the cells have been awakened.

First minimal solution to FSSP over a LINE, using a rather huge number of states (several thousands), was given in 1962 by Goto. Then, Waksman in [Wak66] showed a 16-state CA which is a well known algorithm, often considered to be the first minimum time solution to FSSP. In the same years, Balzer gave an 8-state solution, [Bal67]. Twenty years later Mazoyer constructed a 6-state minimum time solution which is the best known solution with respect to the number of states, [Maz87]. Optimum time solutions to the FSSP over a LINE have been analyzed and compared in [UHS05].

In the generalized version of FSSP problem (GFSSP) the GENERAL is located on any cell, that is an arbitrary cell may be in state G in the initial configuration. Moore and Langdon, in [ML68], showed a 17-state solution to GFSSP in time $n-1+max(p; n-p+1)$, which is the minimum time required to synchronize the cells of a LINE where p is the number of cells between the GENERAL and the closest end of the LINE.

Other variations of the problem have been considered in literature, by modifying the number of the GENERALS in the initial configuration of the given CA. In the Multi-General FSSP problem (MG-FSSP), there are a number k of GENERAL cells, each being p_i cells far from the leftmost one, for $i \in [1, \cdots, k]$. In [SW04] a minimum time solution has been given in time $n+max(min_i(p_i); n-1-max_i(p_i))$. In the same paper, a more general case has also been considered in which the GENERALS start their work in an asynchronous way, each at a time t_i (A-MG-FSSP problem). In this case a lower bound has been provided but it has been proved that for each CA solving A-MG-FSSP, there are infinitely many instances for which the CA does not synchronize in the optimum time.

An interesting case is when two GENERALS occur in both the boundary cells, and the problem arising is known as Two-End synchronization of a LINE, [Cul89]. A Two-End synchronization of a LINE of n cells can be obtained in time n by considering the line as being split into two halves. Each of these half-lines can be seen as a LINE with one GENERAL, and is synchronized by a minimum time solution which starts from the boundary cells. In the case n is odd, the central cell belongs to both half-lines, and each half-line has $(n+1)/2$ cells. Thus, all the cells fire simultaneously in time $2(n+1)/2-1=n$. When n is even, the line is divided in two sub-lines of $n/2$ cells. Anyway, each of the two central cells can act as the last cell of its half-line only when it receives an input from the other half-line, thus the synchronization of each half-line needs a further unit of time, and then it works in time n also in this case.

Concerning cellular automata with the ability to transmit just one bit of information, Mazoyer, in [Maz96], showed that a minimum time solution exists even for a 1-LINE. Observe that in general 1-CA, i.e. CA using 1-bit channels, can simulate standard CA but this simulation causes a slow down of a factor $log|Q|$, where $|Q|$ is the number of states of the CA.

The Two-End synchronization outlined above does not work if one uses the solution to the case of 1-bit communication channel. The main reason is that the synchronization of 1-CA makes an important use of the information on the parity of the bits received by each cell. In [GLNP06, GLP07], the authors show that a unit time of delay is then necessary when the number of cells is even. This leads to a Two-End synchronization in time $2\lfloor n/2 \rfloor + 1$.

Ring and oneWay-Ring. A solution to the FSSP over a RING of n cells can simulate a Two-End synchronization of a LINE of $n+1$ cells. Actually, the cell number 1 of the RING can act as both the boundary cells of the LINE. This idea is due to Culik, but he considered that an n cell RING could simulate a Two-End synchronization of an n cell LINE, thus obtaining an imprecise solution in time n, [Cul89].

We have already said that a Two-End synchronization of a LINE of $n+1$ cells can be obtained in time $n+1$, which is indeed the minimum time for an n cell RING. This is actually the time required for the GENERAL to wake all the cells up and receive the information back (see [GLP07] for a proof).

When the channels can communicate just one bit of information, the simulation of a Two-End synchronization of a LINE of $n+1$ cells leads to a solution to 1-RING in time $2\lceil n/2 \rceil + 1$, [GLNP06, GLP07]. Hence, there is a gap between the lower and the upper bounds for the synchronization of a 1-RING with an odd number of cells.

The lower bound for the synchronization time, has been refined in [LNP96], in the case of oneWay-RINGS: it has been proved by contradiction that any solutions over a oneWay-RING requires at least time $2n$. Suppose that a solution in time less than $2n$ exists for FSSP over a oneWay-RING A with n cell, the same solution does not work correctly for a oneWay-RING B of $2n$ cell. One can see that the state entered by the n-th cell of A in any time $t < 2n$ may be the same as the state entered by the n-th cell of B at the same time. Hence, if the cells of A enter the FIRING state in a time $t < 2n$, then, when the algorithm runs on B, the n-th cell enters the FIRING state (the same state entered in A), while the last cell of B is still LATENT (it has not been awakened yet). Observe that in [Cul89], an imprecise solution in time $2n-1$ was given.

It is easy to see that any synchronization of a LINE A of n processors in time $t(n)$ can be simulated on a oneWay-RING B in time $2t(n)-1$. Actually, suppose that a cell i of A enters a state p in one step, in accordance to the transition function, and depending on the states q_L, q and q_R, of the cells $i-1$, i and $i+1$, respectively. By using two steps, the cell $i+1$ of B can get both q_L and q from the cells $i-1$ and i, and then can enter the state p. Thus, if A and B start from the same initial configuration at time 1 and A execute $t(n)-1$ steps to reach a firing configuration, then, with the above simulation, all the cells of B fire after $2(t(n)-1)$ steps, in time $2t(n)-1$. The above idea can be exploited to get a solution to the FSSP for oneWay-RING in time $2n$. Consider a Two-End synchronization S of a LINE which takes time n. In the first step, a solution over the oneWay-RING lets also the second cell enters the state G. Now, the oneWay-RING can be seen as beginning from the second cell and ending to the first cell,

thus having the same initial configuration as the LINE when S starts. With an algorithm implementing the above described simulation, the oneWay-RING is synchronized in time $2n$, see [LNP96].

The main results of the section are now summarized.

Theorem 1. *The following results hold:*

- *Every solution to FSSP over a* LINE *of n cells requires at least time $2n-1$ and optimum time solutions to FSSP over a* LINE *and a* 1-LINE *exist;*
- *Every solution to FSSP over a* RING *of n cells requires at least time $n+1$ and an optimum time solution exists;*
- *Every solution to FSSP over an oneWay-*RING *requires at least time $2n$, and an optimum time solution exists;*
- *Every solution to FSSP over* 1-RING *requires at least time $n+1$ and there is a synchronization of a* 1-RING *in time $2\lceil n/2 \rceil + 1$.*

The table in Fig. 3 schematically summarizes the results of this subsection.

Model	Variation	Synchronization time	Reference	Minimum time?
LINE	–	$2n-1$	[Wak66]	YES
LINE	Generalized	$n-1+max(p; n-p+1)$	[ML68]	YES
LINE	k GENERALS	$n-1+max(min_i(p_i); n-max_i(p_i)+1)$	[SW04]	YES
LINE	1-bit	$2n-1$	[Maz96]	YES
LINE	Two-End	n	[Cul89]	YES
LINE	Two-End, 1-bit	$2\lfloor n/2 \rfloor + 1$	[GLNP06, GLP07]	for odd n
RING	–	$n+1$	[Cul89]	YES
RING	1-bit	$2\lceil n/2 \rceil + 1$	[GLNP06, GLP07]	for even n
RING	$\geq$ 2-bit	$n+1$	[GLNP06, GLP07]	YES
RING	oneWay	$2n$	[LNP96]	YES

Fig. 3. Solutions for one-dimensional Communication Networks

3.2 Two and Higher Dimensions

Square. Also in this case, a plethora of scenarios and algorithms have been given. The spark for the two-dimensional case with the general in a corner cell, was given by the results contained in the Ph.D. thesis of Wendel Terry Beyer, [Bey69] and of Ilka Shinahr, this latter then published in [Shi74]. For an $n \times n$ SQUARE, lower and upper bounds were given in time $2n-1$. The solution in [Shi74] used 17 states, and still today there is an active research for this case. In [UK10] were used only 7 states adopting the so-called zebra mapping, and recently another algorithm has been presented in [UUN11], based on a new technique called one-sided recursive halving markings and using 37 states and 3271 rules, which is quite different from the well known classical algorithms of [Bey69,Shi74]. We omit here even the basic idea of classical algorithms, since it has been exposed several times in literature, see for example Wikipedia.

Rectangle. In the case of a RECTANGLE of $m \times n$ cells, $m \neq n$, in [Shi74] the author gave the lower bound for a solution in time $n+m+\max(m,n)-2$ along with a matching time algorithm. For this case too there is an active research: in [UKT13] a new algorithm has been given which has some nice properties, like the

easiness in the verification of its correctness and the fact that it can be extended to a solution for generalized FSSP, where the general is at an arbitrary position in the array (for this feature see also [UK12]). Another peculiarity is that it is isotropic with respect to the shape of a given rectangle array, that is there is no need to control the FSSP algorithm for longer-than-wide and wider-than-long input rectangles.

1-*Square.* In the case the communication channel is restricted to just 1-bit a trivial lower bound on the minimum firing time of a square $n \times n$ is clearly $2n - 1$. For this model, in [GLP04,GLP07], a tight upper bound of time $2n - 1$ has been shown, for the first time in this case.

As regards Four-End FSSP, where there are GENERALs positioned in the four corners of an $n \times n$ SQUARE, some synchronization algorithms have been given. When the channels have the capacity to transmit a state, it is in time n, while for the 1 bit case, it is in time $2\lfloor n/2 \rfloor + 1$, in a similar way as for Two-End synchronization of a LINE.

It should be noted that many of the optimum time solutions proposed for the two-dimensional case, are derived from well-known solutions for the one-dimensional case, for example those in [UK12,GLP07], embed the synchronization algorithms given by Mazoyer, the former resembles the one of [Maz87] using only 6-states, and the latter, given for 1-bit communication channels, combines that in [Maz96] and the classical of [Shi74].

Torus. As in the one-dimensional case for the RINGs, the minimum time to synchronize is at least the time necessary by the GENERAL to send and hence receive back a message to/from all the other cells. In [GLP07] it has been shown that in the case of a TORUS (sometime called also Square of Rings) this time is $n+1$. It is not trivial at all to establish that, in the case of one-way communication, this time rises to $3n-1$, as shown in [LNP96], along with matching upper bound algorithms. This lower bound is shown by contradiction, in a similar way as for the lower bound of the synchronization time of a RING.

In the case of a 1-bit channel, by extensively using the result for Four-End FSSP, in [GLNP06,GLP07], an almost optimum upper bound is given in time $2\lceil n/2 \rceil + 1$. Let us underline that the restriction on the channel capacity is to be meant strictly and not as a shorthand for a big-O notation of some constant. In fact, it is worth noting that in the case of a channel with capacity strictly greater than 1-bit, in this same paper Gruska et al. showed an algorithm synchronizing in time $n + 1$, thus optimum in time. The idea exploited there was to use the second bit to recognize the type of messages, a thing that in the case of 1-bit channel was done exploiting the arrival time delay of the messages, thus not allowing to get the tightness of the bound.

Still for 1-bit communication channel, in [UMK03] some nice non optimum-time algorithms for synchronize any $n \times n$ SQUARE and $m \times n$ RECTANGLE in $2n$ and $m + n + max(m, n) + 1$ steps, have been given, respectively. The time complexities for these two algorithms are one step larger than the optimum ones.

Array of Dimension Greater than 2. In [UNK12] lower and upper bounds on optimum synchronization time for arrays of dimension $k > 2$ have been given. In particular it has been shown that, in case of one GENERAL in a corner of an array of size $n_1 \times n_2 \times \ldots \times n_k$, the minimum time is $\sum_{i=1}^{k} n_i + \max(n_1, n_2, \ldots, n_k) - k$ steps. In the same paper, this result has been generalized also for a GENERAL in any position.

We can now summarize the main results of this subsection.

Theorem 2. *The following results hold:*

- *Every synchronization of a* SQUARE *of* $n \times n$ *cells requires at least time* $2n-1$*, and an optimum time solution exists over a* SQUARE *and a* 1-SQUARE*;*
- *Every synchronization of a* TORUS *of* $n \times n$ *cells requires at least time* $n+1$*, and an optimum time solution exists;*
- *Every synchronization of a oneWay-*TORUS *of* $n \times n$ *cells requires at least time* $3n-1$*, and an optimum time solution exists;*
- *Every synchronization of a* 1-TORUS *of* $n \times n$ *cells requires time at least time* $n+1$ *and a solution exists in time* $2\lceil n/2 \rceil + 1$*.*

The table in Fig. 4 schematically summarizes the results cited in this subsection.

Model	Variation	Synchronization time	Reference	Minimum time?
RECTANGLE	–	$n + m + \max(m, n) - 2$	[Shi74]	YES
SQUARE	–	2n-1	[Bey69, Shi74]	YES
SQUARE	1-bit	$2n - 1$	[GLP04, GLP07]	YES
SQUARE	Four-End, 1-bit	$2\lfloor n/2 \rfloor + 1$	[GLNP06, GLP07]	for odd n
TORUS	–	$n + 1$	[GLNP06, GLP07]	YES
TORUS	1-bit	$2\lceil n/2 \rceil + 1$	[GLNP06, GLP07]	for even n
TORUS	2-bit	$n + 1$	[GLNP06, GLP07]	YES
TORUS	one-way	$3n - 1$	[LNP96]	YES

Fig. 4. Solutions for two-dimensional Communication Networks

A particular mention deserves the, somewhat orthogonal, work on the FSSP of Kojiro Kobayashi started, at the best of our knowledge, in 1977 with the paper [Kob77] till the beautiful recent one [GK12]. The trait of Kobayashi's studies is easily recognized as his research has been almost always focused on non regular shapes of the network cells and the FSSP has been related to a combinatorial problem for which only exponential algorithms are known. For example in [Kob01] the cells of the network are placed along a path of two-dimensional array space, and along with Goldstein, in [GK05, GK12], it has been shown that if a minimum-time solution exists, (for example for the path problem mentioned above), then there must exist polynomial time algorithms for solving the diameter problem in the standard RAM model of computation, this implying that P = NP.

The FSSP has been also studied in a setting where the processors do not share a global clock, though working in synchrony, and can be faulty [CDDS89]. In this paper upper and lower bounds are shown on the number of faulty processors that can be tolerate and still reach a solution. Recently this fault tolerant scenario

has been resumed in [DHM12] and enriched by the first self-stabilizing firing squad algorithm, allowing thus recovery from arbitrary transient errors. This algorithm is optimum with regards to two aspects: if the algorithm is in a safe state, it reaches the consensus as fast as any other algorithm does once a cell receives a start signal, and the second is that starting from an arbitrary state, it converges to a safe state as fast as any other algorithm does.

Finally let us also mention a series of interesting papers related to the FSSP which have been very recently published by Arnold L. Rosenberg, see for example [Ros14,Ros13], that model robots with teams of cellular automata whose aim is to identify and search within squares of $n \times n$ mesh of tiles.

3.3 Open Problems

One of the best known open problem in this area regards the minimum number of states necessary to synchronize a Line. Clearly, any CA solving FSSP has at least three states (G, L and F). Sanders, in [San94], showed that no 4-state CA can solve FSSP. Since the known solution with the least number of states is the one due to Mazoyer, with 6 states, it is unknown whether the minimum number of states is indeed 6 or there is a CA beating Mazoyer's solution.

All the solutions in Theorem 1 and in Theorem 2 are optimum in time, and thus the lower bounds are tight, except for a gap between the lower and the upper bounds for synchronization of both a 1-Ring and a 1-Torus with an odd number of cells. In these cases, the best known synchronization algorithms require time $n+2$, whilst the known lower bound is $n+1$. In [GLNP06,GLP07], solutions in time $n+1$ have been shown for a Ring and for a Torus with 2-bit communication channels, thus showing the tightness of the given lower bound in the case of $O(1)$-bit channels. It is still an open problem to state whether such an optimum time solution exists for 1-Ring, or the given lower bound can be made tighter.

Finally, another interesting matter regards the synchronization of CA over a Ring and over a Torus with both the restrictions: one-way and 1-bit communication channels. Clearly, no solution can consume less than $2n$ time, for a 1-bit oneWay-Ring and less than $3n-1$ time, for a 1-bit oneWay-Torus. Anyway we do not know whether such solutions. The idea of the algorithms given for oneWay-Ring and oneWay-Torus does not work in the same time when 1-bit communication channels are used, since the number of steps necessary for a cell to send its state is greater than one in this case.

4 Synchronization Algorithms in Particular Time

In this section we will survey some of the main results known in literature to get *non minimum time* synchronization problems. More precisely, particular times in which, and also particular ways to, synchronize cellular automata ([LNP98, LNP00,GLNP06]).

The general pattern to get a synchronization in a particular given (not minimum) time, is the following simple idea: given a communication network A of parameter n, a signal[3] is generated from the General cell and comes back to it in time $(t(n) - min(n))$ where $min(n)$ is the minimum time for a CA to solve the FSSP over A, and $t(n) \geq n$. Then a minimum time solution starts, synchronizing thus A in time $t(n)$. For example, in the case A is an n-cell LINE or an $n \times n$ SQUARE, then $min(n) = 2n - 1$. However, despite its simplicity, sometimes the implementation of this idea is far from trivial, since it implies different compositions of different signals and synchronization algorithms. All the results have been obtained for the most general case, the 1-bit communication channel thus they hold *sic et simpliciter* for all types of channel capacities.

Theorem 3. *Synchronization algorithms in time* n^2, 2^n, $n\lceil \log n \rceil$, *and* $n\lceil \sqrt{n} \rceil$ *exist, for a* 1-LINE, 1-SQUARE, 1-RING, *and a* 1-TORUS.

Moreover other results have been given in the case of one-way communication.

Theorem 4. *Synchronization algorithms in time* n^2, 2^n, *and* $n\lceil \log n \rceil$ *exist, for a* 1-*one Way*-RING *and* 1-*one Way*-TORUS.

To obtain other particular synchronization times, some signal and synchronization compositional operations have been introduced: *parallel, sequential* and *iterated*. Also sufficient conditions when these operations may be applied have been provided.

It is easy to implement the standard cross product of automata between two CA's A_1, A_2, using a greater channel capacity: the communication channels are kept distinct and therefore $A_1 \times A_2$ can run in parallel the synchronization algorithms. On top of this a CA that *selects* between two distinct synchronization algorithms, according to a given condition $P(n)$, has also been given. Examples of $P(n)$ are the parity of n or the fastest/slowest synchronization times. In particular this last feature can be realized as follows: given two CA's synchronizing in time $t_1(n)$ and $t_2(n)$, respectively, another CA is obtained as the cross product of the two and of a third one that *selects* according to the result of the test $t_1(n) \leq t_2(n)$.

Theorem 5. *Let* A *and* B *be solutions to FSSP with channel capacity* a *and* b, *respectively. It is possible to design a CA with channel capacity* $a + b + c$, *for some integer* c, *synchronizing in the minimum (maximum) times of* A *and* B.

Other particular times have been obtained, such as the sum and the product, holding for all the 1-bit models considered so far. In particular, the technique to obtain the product is interesting. It extends the cross product to let a solution in time $t_1(n)$ to be iterated $t_2(n)$ times, without using extra capacity of the channels.

[3] The concept of signal is often used in literature as a way to facilitate the description of a CA. Informally speaking, a signal can be seen in the space-time unrolling of the CA as a set of cells that, at a given time, receive/send words, different from all-zero words, from/to adjacent cells.

Theorem 6. *Given two solutions to the FSSP in time $t_1(n)$ and $t_2(n)$, and an integer $d \geq 0$, solutions in time $t_1(n) + t_2(n) + d$, and $t_1(n) \cdot t_2(n)$ exist.*

The next result provides solutions for a 1-SQUARE and for a 1-RECTANGLE, starting from that for a 1-LINE. Actually, in an $(n \times m)$-cell RECTANGLE, several LINEs of $n+m-1$ cells can be individuated, starting from cell $(1,1)$ and ending to cell (n,m) (for example, the cells in the first row and in the last column form one of such LINEs). Analogously, $(2n-1)$-cell LINEs can be individuated in an $n \times n$ SQUARE. All such LINEs can be synchronized in parallel, using the same algorithm.

Theorem 7. *If there is a synchronization of a* 1-LINE *in time $t(n)$, then a synchronization on a* 1-SQUARE *in time $t(2n-1)$ and a synchronization on a* 1-RECTANGLE *in time $t(n+m-1)$ exist.*

From this last result and Theorem 3, the next theorems follow.

Theorem 8. *Synchronizations on a $n \times n$* 1-SQUARE *in time $K^2, 2^K, K \cdot \lceil \log K \rceil$ and $K \cdot \lceil \sqrt{K} \rceil$, for $K = 2n-1$, exist.*

Theorem 9. *Synchronizations on a $n \times m$* 1-RECTANGLE *in time $J^2, 2^J, J \cdot \lceil \log J \rceil$ and $J \cdot \lceil \sqrt{J} \rceil$, for $J = n+m-1$, exist.*

Finally let us also mention that synchronizing algorithms for all the models we have dealt with in this paper, exist in any time that can be expressed as a polynomial of n, formally stated in the following theorem.

Theorem 10. *Let $h \geq 2$ and $a_0, \ldots, a_h$ be integers with $a_h \geq 1$. A synchronization in time $a_h n^h + a_{h-1} n^{h-1} + \ldots + a_1 n + a_0$ exists on an n-cell* 1-LINE, 1-RING, 1-*oneWay*-RING, *and on an $(n \times n)$-cell* 1-SQUARE, 1-TORUS, 1-*oneWay*-TORUS.

References

[Bal67] Balzer, R.: An 8-state minimal time solution to the firing squad synchronization problem. Information and Control **10**(1), 22–42 (1967)

[Bey69] Beyer, W.T.: Recognition of topological invariants by iterative arrays. PhD thesis, Massahcusetts Institute of Technology (1969)

[CDDS89] Coan, B.A., Dolev, D., Dwork, C., Stockmeyer, L.J.: The distributed firing squad problem. SIAM J. Comput. **18**(5), 990–1012 (1989)

[Cul89] Karel Culik, I.I.: Variations of the firing squad problem and applications. Inf. Process. Lett. **30**(3), 153–157 (1989)

[DHM12] Dolev, D., Hoch, E.N., Moses, Y.: An optimal self-stabilizing firing squad. SIAM J. Comput. **41**(2), 415–435 (2012)

[GK05] Goldstein, D., Kobayashi, K.: On the complexity of network synchronization. SIAM J. Comput. **35**(3), 567–589 (2005)

[GK12] Goldstein, D., Kobayashi, K.: On minimal-time solutions of firing squad synchronization problems for networks. SIAM J. Comput. **41**(3), 618–669 (2012)

[GLNP06] Gruska, J., La Torre, S., Napoli, M., Parente, M.: Different time solutions for the firing squad synchronization problem on basic grid networks. RAIRO - Theoretical Informatics and Applications, ITA **40**(2), 177–206 (2006)

[GLP04] Gruska, J., La Torre, S., Parente, M.: Optimal time and communication solutions of firing squad synchronization problems on square arrays, toruses and rings. In: Calude, C.S., Calude, E., Dinneen, M.J. (eds.) DLT 2004. LNCS, vol. 3340, pp. 200–211. Springer, Heidelberg (2004)

[GLP07] Gruska, J., La Torre, S., Parente, M.: The firing squad synchronization problem on squares, toruses and rings. Int. J. Found. Comput. Sci. **18**(3), 637–654 (2007)

[Got62] Goto, E.: A minimum time solution of the firing squad problem. In: Course Notes for Applied Mathematics 298, Harvard University, pp. 52–59 (1962)

[Kob77] Kobayashi, K.: The firing squad synchronization problem for two-dimensional arrays. Information and Control **34**(3), 177–197 (1977)

[Kob01] Kobayashi, K.: On time optimal solutions of the firing squad synchronization problem for two-dimensional paths. Theor. Comput. Sci. **259**(1–2), 129–143 (2001)

[LNP96] La Torre, S., Napoli, M., Parente, M.: Synchronization of one-way connected processors. Complex Systems **10**(4), 239–256 (1996)

[LNP98] La Torre, S., Napoli, M., Parente, D.: Synchronization of a line of identical processors at a given time. Fundam. Inform. **34**(1–2), 103–128 (1998)

[LNP00] La Torre, S., Napoli, M., Parente, M.: A compositional approach to synchronize two dimensional networks of processors. Theoretical Informatics and applications, ITA **34**(6), 549–564 (2000)

[Maz87] Mazoyer, J.: A six-state minimal time solution to the firing squad synchronization problem. Theor. Comput. Sci. **50**, 183–238 (1987)

[Maz96] Mazoyer, J.: On optimal solutions to the firing squad synchronization problem. Theor. Comput. Sci. **168**(2), 367–404 (1996)

[ML68] Moore, F.R., Langdon, G.G.: A generalized firing squad problem. Information and Control **12**(3), 212–220 (1968)

[Moo64] Moore, E.F. (ed.): Sequential Machines: Selected Papers. Addison-Wesley Longman Ltd., Essex, UK, UK (1964)

[Ros13] Rosenberg, A.L.: Finite-state robots in a warehouse: Achieving linear parallel speedup while rearranging objects. In: Parallel Processing (ICPP), 2013 42nd International Conference on, pp. 379–388. IEEE (2013)

[Ros14] Rosenberg, A.L.: Region management by finite-state robots. The Computer Journal **57**(1), 59–72 (2014)

[San94] Sanders, P.: Massively parallel search for transition-tables of polyautomata. In: Parcella, pp. 99–108 (1994)

[Shi74] Shinahr, I.: Two- and three-dimensional firing-squad synchronization problems. Information and Control **24**(2), 163–180 (1974)

[SW04] Schmid, H., Worsch, T.: The firing squad synchronization problem with many generals for one-dimensional ca. In: Exploring New Frontiers of Theor. Inf., IFIP (TCS2004), pp. 111–124. Kluwer (2004)

[UHS05] Umeo, H., Hisaoka, M., Sogabe, T.: A survey on optimum-time firing squad synchronization algorithms for one-dimensional cellular automata. International Journal of Unconventional Computing **1**(4), 403–426 (2005)

[UK10] Umeo, H., Kubo, K.: A seven-state time-optimum square synchronizer. In: Bandini, S., Manzoni, S., Umeo, H., Vizzari, G. (eds.) ACRI 2010. LNCS, vol. 6350, pp. 219–230. Springer, Heidelberg (2010)

[UK12] Umeo, H., Kubo, K.: Recent developments in constructing square synchronizers. In: Sirakoulis, G.C., Bandini, S. (eds.) ACRI 2012. LNCS, vol. 7495, pp. 171–183. Springer, Heidelberg (2012)

[UKT13] Umeo, H., Kubo, K., Takahashi, Y.: An isotropic optimum-time fssp algorithm for two-dimensional cellular automata. In: Malyshkin, V. (ed.) PaCT 2013. LNCS, vol. 7979, pp. 381–393. Springer, Heidelberg (2013)

[Ume96] Umeo, H.: A note on firing squad synchronization algorithms. In: Worsch, T., Kutrib, M., (ed.), IFIP Cellular Automata Workshop 96, pp. 65 (1996)

[UMK03] Umeo, H., Michisaka, K., Kamikawa, N.: A synchronization problem on 1-bit communication cellular automata. In: Sloot, P.M.A., Abramson, D., Bogdanov, A.V., Gorbachev, Y.E., Dongarra, J., Zomaya, A.Y. (eds.) ICCS 2003, Part I. LNCS, vol. 2657, pp. 492–500. Springer, Heidelberg (2003)

[UNK12] Umeo, H., Nishide, K., Kubo, K.: A simple optimum-time fssp algorithm for multi-dimensional cellular automata. In: 18th international workshop on Cellular Automata and Discrete Complex Systems and 3rd international symposium Journées Automates Cellulaires, AUTOMATA & JAC, volume 90 of EPTCS, pp. 151–165 (2012)

[UUN11] Umeo, H., Uchino, H., Nomura, A.: How to synchronize square arrays in optimum-time - a new square synchronization algorithm. In: International Conference on High Performance Computing & Simulation, HPCS, pp. 801–807. IEEE (2011)

[Wak66] Waksman, A.: An optimum solution to the firing squad synchronization problem. Information and Control **9**(1), 66–78 (1966)