Generalized FSSP on Hexagonal Tiling: Towards Arbitrary Regular Spaces

Luidnel Maignan[1] and Jean-Baptiste Yunès[2(✉)]

[1] LACL, Université Paris-Est-Créteil, Créteil, France
Luidnel.Maignan@u-pec.fr
[2] LIAFA, Université Paris-Diderot, Paris, France
Jean-Baptiste.Yunes@univ-paris-diderot.fr

Abstract. Here we present a solution to the generalized firing squad synchronization problem that works on some class of shapes in the hexagonal tiling of the plane. The solution is obtained from a previous solution which works on grids with either a von Neumann or a Moore neighborhood. Analyzing the construction of this previous solution, we were able to exhibit a parameter that leads us to abstract the solution. First, and for an arbitrary considered neighborhood, we focus our attention on a class of shapes built from this neighborhood, and determine the corresponding parameter value for them. Second, we apply our previous solution with the determined parameter value for the hexagonal neighborhood and show that, indeed, all the considered shapes on the hexagonal tiling synchronizes.

1 Introduction

The Firing Squad Synchronization Problem (FSSP for short) is a very old problem. It has been reported for the first time in 1957 by John Myhill (see [7]). The goal is to design a cellular automaton (CA for short) such that starting from a initial configuration where every cell is inactive except one cell, called the general or the initiator, the dynamics leads to an uniform configuration where all cells are in the same (firing) state that has never been reached before.

There is a lot of solutions to that problem, each focused on some variation of it. One may want to synchronize lines, rectangles (see [2,3]), parallelepipeds (see [14,15,17]), graphs (see [11]), Cayley graphs (see [10]); another may want to be able to start the process at a given special position, at any position (see [8]), start the process at many different places at the same time or not (see [12]), etc. Some others focused their attention on lowering the synchronization time, the set of states (see [1,6,9,13]), or the communication capabilities of cells. Actually, there are probably more than 150 papers on the subject.

But all in all, almost all solutions are recursive as the simple basic idea is to recursively split the space into equals parts, until elementary sub-spaces are obtained that are obvious to synchronize. Most of the solutions split the space into two equals parts (for a more complex scheme see [6]).

This work is partially supported by the French program ANR 12 BS02 007 01.

T. Isokawa et al. (Eds.): AUTOMATA 2014, LNCS 8996, pp. 83–96, 2015.
DOI: 10.1007/978-3-319-18812-6_7

In a previous paper [4], by the use of so-called distance fields we gave a very general scheme that captures the core of many unidimensional solutions. A distance field is an open cellular automaton which ultimately computes on each point of the space its distances to some given set of reference points. We believe that it is a nice way to understand what is behind the scene in many solutions. Although most of them are constructed in an *ad-hoc* way, it seems clear that some distance information, and therefore distance fields, are implicitly used to determine their splitting points.

In [4], we took an explicit approach to build the unidimensional solution. We use distance fields to detect middles as required to split the space into half-spaces, and we compose as many instances of this splitting process as required to split half-spaces into quarter-spaces and so on recursively. Then a reduction to a finite number of states is described that leads to a classical finite cellular automaton. This two-steps approach allows the solution to be correct by construction, the infinities allowing to be abstract, more semantic and clearer. A description with a finite number of states usually leads to less obvious semantics and dynamics of the transition function, especially when minimal synchronization time is aimed.

In [5], we proposed a generalization of this unidimensional solution to handle more different spaces within a single scheme. It was parametrized by some information extracted from the given neighborhood. We showed that it works on the classical and less classical space shapes either with the Moore or the von Neumann neighborhood. We now want to show how all of this can be applied as-is on more general spaces. We tested our construction on various space shapes with an hexagonal neighborhood.

Hexagonal cellular spaces are of special interest because there is no known solution to the generalized FSSP on them up to now. Researchers focused on 1D lines, 2D/3D square grids, some on more general graphs but, to the best of our knowledge, not on hexagonal tilings with arbitrary position of the initiator, or at least not on various shapes on the hexagonal tilings.

In Sect. 2, we give the reader the necessary background on the previous works. The paper is written to ease a global understanding of the key concepts, questions, and answers without having to dig too much in the details of the transition function and the previous results. That section mainly focuses on the parameter of the scheme which allowed to switch from 1D to the various 2D cellular spaces considered in previous works. At that time, some of the values of this parameter were determined in an *ad-hoc* way.

The purpose of Sect. 3 is to extend the work by proposing a procedure to determine the possible values for the parameter. This will also helps to identify a class of synchronizable shapes thus generalizing the set of shapes previously considered.

In Sects. 4 and 5 we proceed with some arguments and experimental results to show and explain how the synchronization is achieved.

In Sect. 7, we finally summarize the paper and discuss interesting future directions of work.

2 Summary of Previous Solutions

As already explained, this work is based on two previous works. The first one [4] gives a very general solution for the 1D FSSP. The second one [5] extends this solution to various 2D space shapes on the Moore and the von Neumann neighborhood.

In this section, we do not give all the details of the internals of the CA as they are not relevant here. Instead, we only introduce the leading concepts of interest for the remaining. The reader is therefore referred to the cited papers for more details.

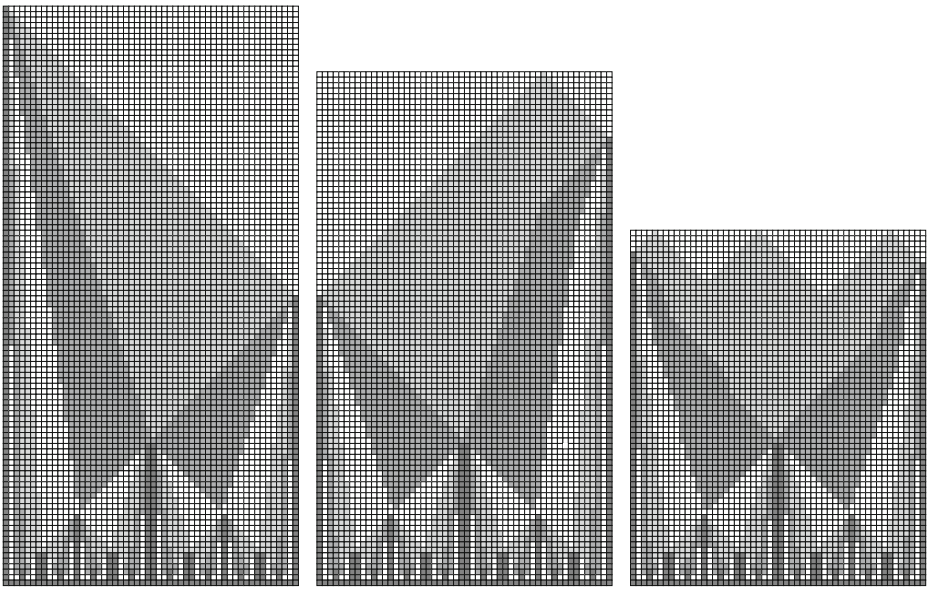


Fig. 1. Evolutions of our 1D algorithm with different sets of generals. The reader must be aware that this does not show the states of the CA but only some "interesting" information extracted from.

Our unidimensional solution was designed from the simple key idea that middles of a space are characterized by their distance to the borders of the space, and that many layers of middles detection have to be stacked to obtain the recursive computation of the synchronization. Figure 1 provides a summarized view of the stack of layers for different sets of initiators, and the space-time diagrams looks a lot like many other optimal time solutions. This is exactly why we claim that our solution captures many classical solutions.

In order to generalize this solution to the case of the classical rectangle, the starting idea was to split this rectangle into four quarter-rectangles and so on recursively. This can be easily achieved by superposing two unidimensional splitting processes: one along the horizontal axis and another one along the vertical axis.

Table 1. Values of parameter ν used for borders

Moore	von Neumann
X-axis	X-axis
$\nu_X^{-1} = \left\{ \binom{-1}{-1}, \binom{-1}{0}, \binom{-1}{+1} \right\}$	$\nu_X^{-1} = \left\{ \binom{-1}{0}, \binom{0}{+1} \right\}$
$\nu_X^{+1} = \left\{ \binom{+1}{-1}, \binom{+1}{0}, \binom{+1}{+1} \right\}$	$\nu_X^{+1} = \left\{ \binom{+1}{0}, \binom{0}{-1} \right\}$
Y-axis	Y-axis
$\nu_Y^{-1} = \left\{ \binom{-1}{-1}, \binom{0}{-1}, \binom{+1}{-1} \right\}$	$\nu_Y^{-1} = \left\{ \binom{-1}{0}, \binom{0}{-1} \right\}$
$\nu_Y^{+1} = \left\{ \binom{-1}{+1}, \binom{0}{+1}, \binom{+1}{+1} \right\}$	$\nu_Y^{+1} = \left\{ \binom{+1}{0}, \binom{0}{+1} \right\}$

The study of this superposition process led us to the introduction of a parameter ν that describes the different axes on which a 1D solution have to be executed. For the Moore neighborhood, the axes that make the synchronization work are the horizontal and vertical ones. For the von Neumann neighborhood, the axes are the north-west/south-east and south-west/north-east axes. This is formally described in Table 1 in terms of ν's. For simplicity, the axes indexes are $D = \{X, Y\}$ in both cases, although this might be slightly misleading. So, for example, ν_X^{-1} can be read as "the subset of neighbors that all contribute to the left-neighborhood along the X axis" and similarly ν_Y^{+1} as "the subset of neighbors that all contribute to the right-neighborhood along the Y axis".

Here "left", "right", "-1" and "$+1$" are purely a matter of convention, since the unidimensional solution is symmetric. The ν's map the given topological neighborhood into many appropriate unidimensional neighborhoods, one per axis.

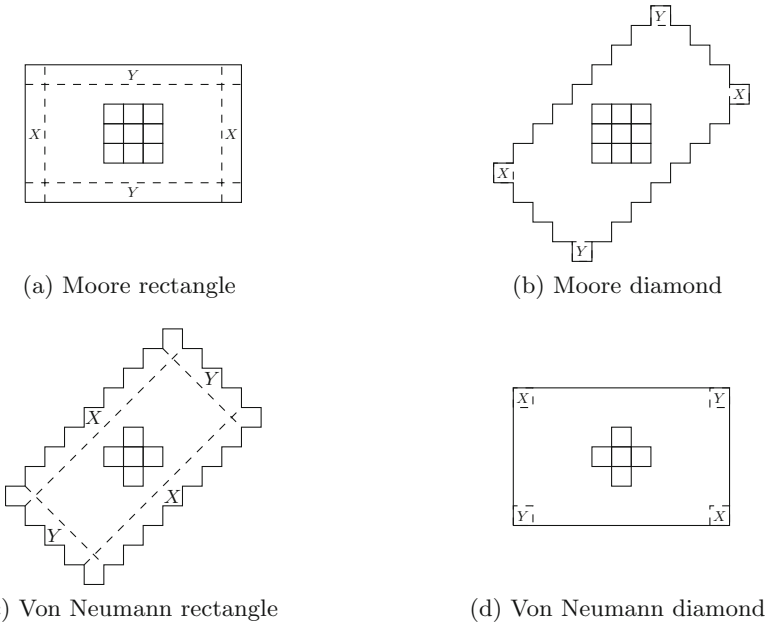
The notion of border is centric to the unidimensional solution. As a consequence we need to find an equivalent counterpart in our cases. Borders are determined for each axis. For any axis $d \in D$, they are determined by the use of the following predicate excerpted from the solution described in [5]:

$$\text{border}_{t+1}^{0,d}(c) = \text{input}_{t+1}(c) \wedge \exists i \in \text{I};\ \forall \delta \in \nu_d^i;\ c + \delta \notin \text{S} \tag{1}$$

This equation defines the borders of the space S along the axis d and at time $t+1$ as the cells c that are activated ($\text{input}_{t+1}(c)$) and have no neighbors in at least one direction $i \in I = \{-1, +1\}$ along the axis, *i.e.* no neighbors "on the left", or no neighbors "on the right".

With these definitions of axes and borders, we showed that the solution synchronizes what we called *rectangles* and *diamonds* for any location of the initiator. These spaces are illustrated in Fig. 2. For a given neighborhood, the rectangle has border cells for one or the other axis all along the boundary of the space while the diamond only have isolated border cells according to Eq. (1).

In [5], the values of the ν's for the Moore and von Neumann neighborhoods were obtained in two distinct ways. Now we will show that a common procedure allows to consider more complex shapes, and in particular hexagonal spaces.



(a) Moore rectangle (b) Moore diamond

(c) Von Neumann rectangle (d) Von Neumann diamond

Fig. 2. Synchronized shapes and their respective borders

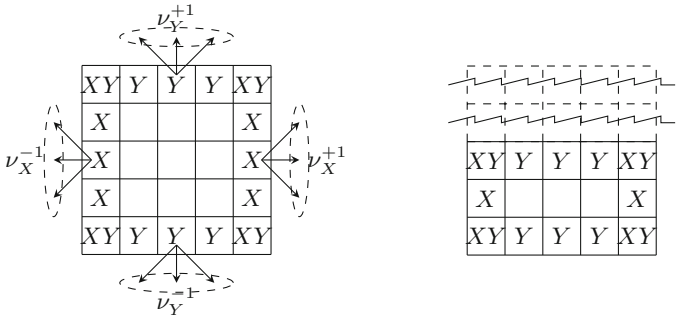
3 Determination of the Axes, Borders and Shapes

In the literature, people are usually interested in the synchronization of classical rectangles like Fig. 2a and d. It can be argued that the most natural neighborhood to consider first on a rectangle is the Moore neighborhood as the Moore rectangle exactly corresponds to what one probably think at first about a classical rectangle is (Fig. 2a). So let us describe our general procedure by examining this case first.

3.1 The Moore Case

With the Moore neighborhood a rectangle can easily be considered as a generalization of a square. A square of side length $2r$ is simply a Moore ball B_r^M of radius r. Note that in this paper, all lengths are given in number of hops which corresponds to a distance, and not in terms of numbers of cells which would add many annoying "+1" in the expressions. By observing what characterizes the borders of a Moore's ball, we can obtain the values of the parameter ν and make explicit the relation with the rectangle as illustrated in Fig. 3b.

On a Moore's ball, one can identify four kinds of border cells. These types are characterized by the set of their missing neighbors. These sets can be paired by symmetry and this pairing naturally corresponds to the concept of axis. Here there are two axes X and Y, each of them having two symmetric sets ν_d^{-1}, ν_d^{+1} for $d \in D = \{X, Y\}$.



(a) B_2^M (b) $R_{4\times 2}^M$

Fig. 3. Moore's balls and rectangles

In this respect, rectangles are very similar to squares. Their borders are also classified into these same four types. Moreover, any rectangle can be obtained by cutting down all cells of a given type. This operation preserves the classification, as the trimmed cells become the missing neighbors of some other cells, the laters becoming the new border cells of the considered type. This is illustrated in Fig. 3b where a rectangle $R_{4\times 2}^M$ is obtained by two removals of border cells of type ν_Y^{+1} from the ball B_2^M. Let us make two important observations.

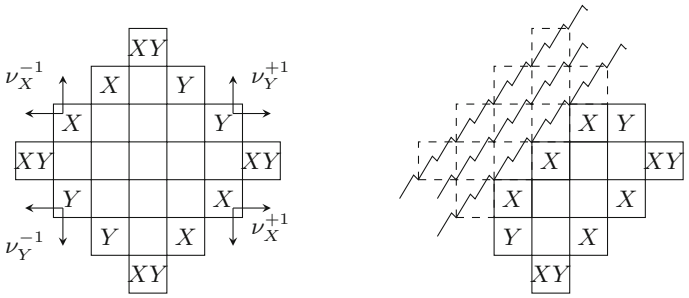
First, the borders of squares are parallel by construction and so are the borders of rectangles since the cutting down operation obviously preserves this property.

Second, the sets ν_D^I thus constructed correspond exactly to the values presented in Table 1 and that allowed the synchronization.

Although, these constructions and properties seem to be obvious in the Moore case, things are slightly subtler in the following cases.

3.2 The von Neumann Case

With the von Neumann topology, one can apply exactly the same process: take a ball, identify the sets of missing neighbors, pair them by symmetry, characterize the axes, and cut down some borders. As for the Moore case, the resulting sets of missing neighbors correspond to the values of Table 1. The construction is illustrated in Fig. 4 where a rectangle $R_{3\times 6}^V$ is obtained by three removals of borders of type ν_X^{-1} from a ball B_3^V. However, if instead of removing border of type ν_X^{-1}, we choose to do three removals of border of type ν_X^{+1}, we obtain a different but symmetric instance of a rectangle $R_{3\times 6}^V$. It might seem completely different from the Moore case but it is in fact very similar if we restrict our attention to the lengths: balls B_r^V are rectangles $R_{2r\times 2r}^V$, so cutting down an X (resp. Y) border reduces the length of the shape along the X-axis (resp. Y-axis). Later we will see a more convincing argument about this, but for now it is sufficient to remark that borders on a given axis remains geometrically parallel.



(a) B_3^V (b) $R_{3\times 6}^V$

Fig. 4. Von Neumann balls and rectangles

It is not common to call such shapes "rectangles", this is why we called them von Neumann rectangles. In the literature, the space considered is almost always the "classical" rectangle that we first considered in the Moore case. One can remark that these "classical" rectangles can be considered (see [16] for example) with the von Neumann neighborhood, but the borders are not where one might expect. If one agrees on our process to identify the borders then the border cells of the "classical" rectangle in the case of the von Neumann neighborhood are only the cells at corners. This is why we called this shape the von Neumann diamond, as the axes we identified are (roughly) the diagonals as illustrated in Fig. 2d.

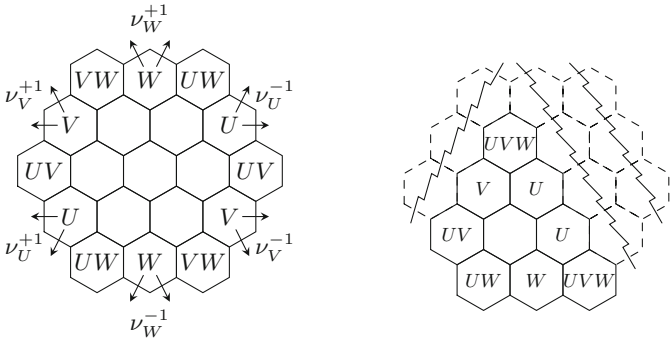
Although this might be surprising, these diamonds have all the good properties to be synchronizable: the borders along a given axis are parallel, and we can remove some borders as described before and preserve the parallelism of the borders. This gives rise to some additional synchronizable shapes, but more is said about this in Sect. 4. As a final note about diamonds, note that the distance between the X borders is the same that the one between the Y borders. In fact, the two axes are totally symmetric in diamonds. Actually, this can be viewed as the reason why a single axis is used in the work presented in [16] which is restricted to von Neumann diamonds.

Similarly the Moore neighborhood can be used on the von Neumann rectangle (this is called a Moore diamond) with the same peculiarities and properties as shown in Fig. 2b.

3.3 The Hexagonal Case

We can now apply the same procedure in the hexagonal case and see what happens. In the hexagonal ball B_r^H of radius r, we can identify six types of borders paired into three axes that we called U, V, and W, as illustrated in Fig. 5a. The values of sets ν_d^{-1}, ν_d^{+1} for $d \in D = \{U, V, W\}$ are shown in Table 2.

Now we cut-down some borders. What is obtained is an object that as three lengths that represents, on each axis, the distance in between parallel borders.



(a) B_2^H (b) $R_{2\times3\times3}^H$

Fig. 5. Hexagonal balls and rectangles

Table 2. Values of parameter ν for the hexagonal case

Hexagonal		
U-axis	V-axis	W-axis
$\nu_U^{-1} = \left\{\binom{+2}{0}, \binom{+1}{+1}\right\}$	$\nu_V^{-1} = \left\{\binom{+2}{0}, \binom{+1}{-1}\right\}$	$\nu_W^{-1} = \left\{\binom{-1}{-1}, \binom{+1}{-1}\right\}$
$\nu_U^{+1} = \left\{\binom{-2}{0}, \binom{-1}{-1}\right\}$	$\nu_V^{+1} = \left\{\binom{-2}{0}, \binom{-1}{+1}\right\}$	$\nu_W^{+1} = \left\{\binom{+1}{+1}, \binom{-1}{+1}\right\}$

By analogy with the previous cases such shapes are called Hexagonal rectangles. This is illustrated in Fig. 5b, where from B_2^H, by removing a border of type ν_V^{+1} and then two borders of type ν_U^{-1} we obtained $R_{2\times3\times3}^H$.

The reader can note some differences with the Moore and von Neumann cases. First in the hexagonal case it possible to remove a border of some type without removing it explicitly. For example, in the figure we removed two U's and one V borders and as a side effect a border of type W also disappeared. Second, our notation $R_{l\times m\times n}^H$ does not represent a single shape as in the Moore case, nor even a class of symmetric shapes as in the von Neumann case, but a class of different shapes (see Fig. 6). But this is really not important for our discussion, as the numbers l, n, m represents the data that really matter for the synchronization, namely the lengths along each axis. Therefore, all shapes in the same class are equivalent for our discussion.

We remind that the important thing is the concept of parallel borders, and this is why we were able to synchronize all those shapes with our algorithm using the right ν's. Now, let us say more about why and how all of this works.

4 Sketch of the Synchronization Process

As we said, the synchronization of the whole space is obtained by superposing independent unidimensional solutions along the identified axis. We therefore

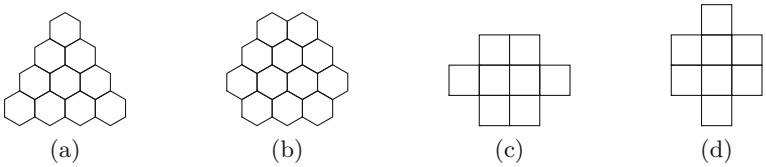


Fig. 6. Two different shapes for $R^H_{3\times3\times3}$ and for $R^V_{3\times3}$

need to explain how the whole is built from the pieces, and then explain what happens along each axis for the whole space.

At the initial configuration, only one cell, the initiator, is active. The first thing to known is that a cell becomes active as soon as one of its neighbors is active, independently of the axes. Once active, a cell participates in all superposed synchronizations, *i.e.* two for the Moore and von Neumann neighborhood, and three for the hexagonal one. For each axis d, the cell uses the corresponding ν_d's to give a real meaning to "left" and "right" in the corresponding unidimensional synchronization. Each instance being independent, it might reach its fire state at a time different from the others. A cell finally fires exactly when *all* the instances have fired. This means that it has to wait for the latest synchronization to effectively fire at the same time.

To have a global meaning for this local behavior, we need to understand how things happen globally for an arbitrary axis, and then see how the independent axes synchronization signals give rise to a coherent compound synchronization signal.

So let us now consider an arbitrary axis d. The first thing to clarify is how the fact that there are many "left" and "right" neighbors (according to the ν_d's) comply with the fact that we execute a single unidimensional synchronization of the axes. This is the reason why we insisted on the parallelism of borders along axes in all considered shapes. This parallelism property means more precisely that for all cells, all unidimensional lines built from them that reach a "left" border using the ν_d^{-1} and that reach a "right" border using the ν_d^{+1} have the same length. Such lines are called *lines along the axis* (see Fig. 7) and their lengths are exactly the ones used in the notation $R_{l\times m}$ and that are given in the figures.

We remind to the reader that our unidimensional solution is based on the notion of distance to the borders, so this parallelism implies that all those lines are equivalent with respect to this notion. When a splitting occurs and new borders are added, we known that the parallelism property still holds for each resulting half-spaces. Indeed the splittings can be described in terms of borders removal, and we showed that this operation preserves the parallelism property. Thanks to this parallelism, the fact that a cell belongs to many lines along a given axis is not a problem. Its goal for all these lines is the same since it has the same distance to the borders for all of them. Also, the unidimensional solution is therefore able to mix distance coming from different "left" neighbors in a

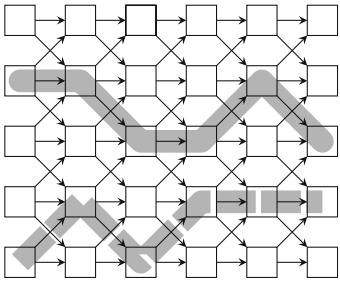


Fig. 7. Two lines along the X axis in the Moore rectangle

single "left" information, and similarly for the right direction. This gives good properties at the global level.

These good properties can be clearly stated in the Moore and von Neumann cases. For any axis, and when we restrict our attention on the activated part of the space, all the lines along the considered axis have exactly the same unidimensional configuration. In particular, when one active cell fires along this axis, all the other active cells fire. So for a given axis either it fires before or after all the cells are active. If it fires after the full activation of the space, its synchronization signal is global. If it fires before the full activation of the space, as illustrated in Fig. 8, then each cell that become active after also fires. The picture is completed by the fact that the last axis to fire necessarily fire after the full activation of the space, and since it is the last to fire, it is the one that determines to complete compound fire.

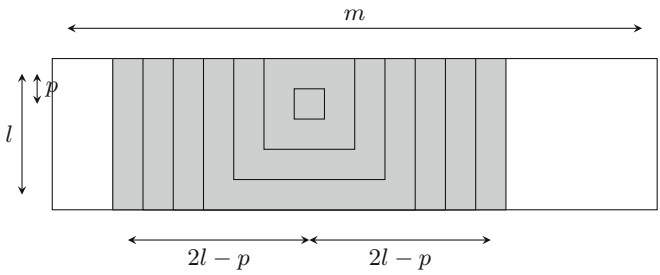


Fig. 8. A Y-axis synchronization on a large rectangle in Moore

For the hexagonal case, it is not the case that all the lines in the active part of the space have the same unidimensional configuration, so a more general argument is needed. The important fact to establish is that when the last axis to fire do so, the axes which were not able to fire globally have already finished to fire all the cells of the space. Instead of digging more into such an argument we

provide some executions on the hexagonal case that allows to verify the property in some elementary and understandable cases.

5 Executions of the Algorithm in the Hexagonal Case

In the hexagonal case, we determined that three axes exist. Thus three synchronizations have to be superposed. This represents a lot of information per cell but one can have a good insight of how the synchronization occurs by simply observing how and when the splittings occur.

Figure 9 illustrates the case of a hexagonal ball of radius 15 with the initiator at the center. Looking at the time of apparition, one can see the classical logarithmic behavior of the divide and conquer scheme used: it takes 15 transitions to activate the borders, then 15 more transitions to get the first splittings. The subsequent splittings occur after 8, then 4, and then 2 transitions at which point for each cell its neighborhood is full of borders of all kind. This event triggers the global synchronization signal. One can also note that the hexagon splits into 6 triangles, and that here after each one splits into 4 triangles. In this example, all the axes act symmetrically since the initiator is exactly at the center of the space.

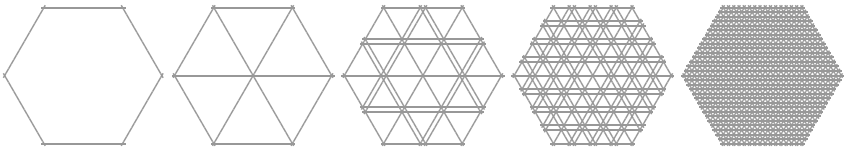


Fig. 9. Splitting of an hexagonal ball of radius 15 with the initiator at the center

In Fig. 10, the independence of the axes and the fact that early splittings end before the final one for each given recursive level can be observed. For time 33 to 39, the axis U starts its splitting, then the axis W, but both of them finished before the axis V finished its first splitting. For the second level of splittings, the same thing can be observed from time 42 to 47, this is harder to observe because the axis U starts its third level of splitting at time 46, making even more explicit the independence of the axes. All in all, V is the last axis to complete the synchronization at all levels, and this determines the global synchronization time that happens at step 54, just after that the whole neighborhood is full of borders of all kind.

In Fig. 11 an even more complex situation is illustrated with splittings occurring during the synchronization of an asymmetric Hexagonal rectangle.

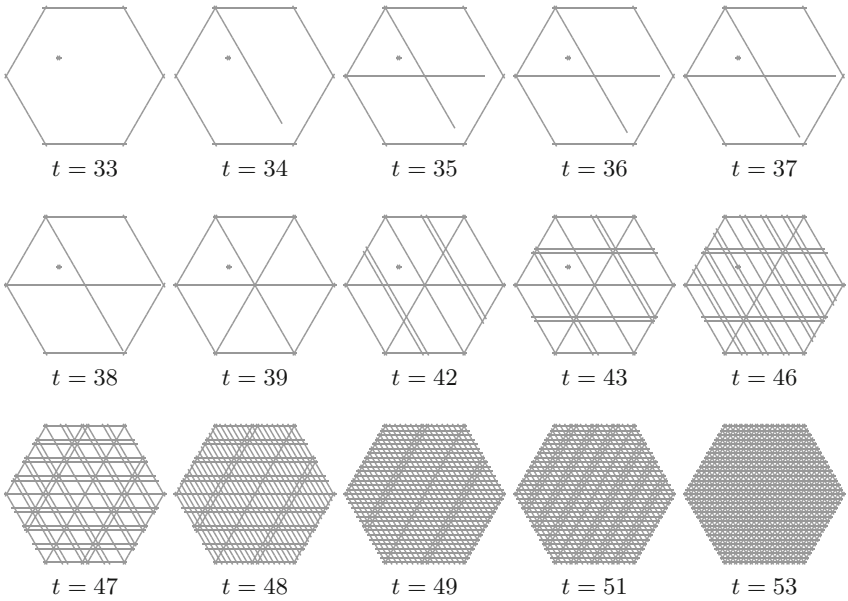


Fig. 10. Splittings of an hexagonal ball of radius 15 with the initiator's position indicated by the isolated point. All configurations where borders appear are shown. The independence of the axes is even clearer.

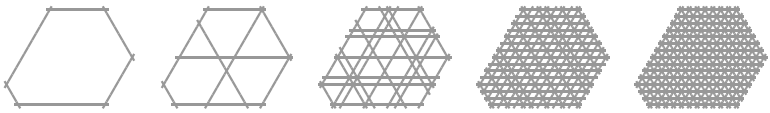


Fig. 11. Splittings of a trimmed hexagonal $R^H_{18\times 14\times 14}$

6 Synchronization Time

An important thing to discuss is the synchronization time. In the unidimensional case, if we denote by l the length of the line (in number of hops, not in number of cells), and we denote by p the distance of initiator to the nearest border, the synchronization occurs after $2l - p + 1$ transitions. Here, we superposed many axis synchronizations. Reminding that the global synchronization occurs when the latest axis synchronizes, we obviously obtain the following formula:

$$T_s = \max_{d\in D}\{2l_d - p_d + 1\}. \tag{2}$$

Then we recover the minimal synchronization time in all the considered case, *i.e.* Moore rectangle, and von Neumann diamond being the most known for the bidimensional case.

7 Conclusion

In [4,5] we presented an algorithm that solves the G-FSSP on various shapes in dimension 1 and 2. Even if it was easy to understand how to extend it to higher dimensions, we asked if our solution was usable to solve the G-FSSP on 2D shapes in the hexagonal topology, which is *a priori* less obvious (remind that no one had proposed a solution up to now). This was obtained by the identification of some properties of the synchronized shapes that permits to determine the right values of some parameters to the algorithm that are related to the topology. We do not have a formal proof of our claims yet, but we experimented successfully everything that is presented here. All the characterizations and properties we talked about can be proved, but this will be the main thread of a work to come. We claim that we have a very generic solution that is able to synchronize many regular shapes of any dimensions with various topologies in minimal-time. Characterizing all the shapes that our solution captures is now a challenge. We also think that it is possible to decorrelate in some way the axes to the neighborhood and to "choose" more independently the axes. Of course, this will necessitate to understand well the relations in between the ν's and the neighborhood.

References

1. Balzer, R.: An 8-state minimal time solution to the firing squad synchronization problem. Inf. Control **10**, 22–42 (1967)
2. Grasselli, A.: Synchronization of cellular arrays: the firing squad problem in two dimensions. Inf. Control **28**, 113–124 (1975)
3. Kobayashi, K.: The firing squad synchronization problem for two-dimensional arrays. Inf. Control **34**, 177–197 (1977)
4. Maignan, L., Yunès, J.-B.: A spatio-temporal algorithmic point of view on firing squad synchronisation problem. In: Sirakoulis, G.C., Bandini, S. (eds.) ACRI 2012. LNCS, vol. 7495, pp. 101–110. Springer, Heidelberg (2012)
5. Maignan, L., Yunès, J.B.: Moore and von Neumann neighborhood n-dimensional generalized firing squad solutions using fields. In: AFCA 2013 Workshop, CANDAR 2013 Conference, Matsuyama, Japan, 4–6 December 2013
6. Mazoyer, J.: A six-state minimal time solution to the firing squad synchronization problem. Theoret. Comput. Sci. **50**, 183–238 (1987)
7. Moore, E.E.: Sequential Machines, Selected Papers, pp. 213–214. Addison-Wesley, Reading (1964)
8. Moore, E.E., Langdon, G.: A generalized firing squad problem. Inf. Control **12**, 212–220 (1968)
9. Noguchi, K.: Simple 8-state minimal time solution to the firing squad synchronization problem. Theoret. Comput. Sci. **314**(3), 303–334 (2004)
10. Róka, Z.: The firing squad synchronization problem on Cayley graphs. In: Hájek, Petr, Wiedermann, Jiří (eds.) MFCS 1995. LNCS, vol. 969, pp. 402–411. Springer, Heidelberg (1995)
11. Romani, F.: Cellular automata synchronization. Inf. Sci. **10**, 299–318 (1976)

12. Schmidt, H., Worsch, T.: The firing squad synchronization problem with many generals for one-dimensional ca. In: Levy, J.J., Mayr, E.W., Mitchell, J.C. (eds.) TCS 2004. IFIP, vol. 155, pp. 111–124. Springer, Heidelberg (2004)
13. Settle, A., Simon, J.: Smaller solutions for the firing squad. Theoret. Comput. Sci. **276**(1), 83–109 (2002)
14. Shinahr, I.: Two- and three-dimensional firing-squad synchronization problems. Inf. Control **24**, 163–180 (1974)
15. Szwerinski, H.: Time-optimum solution of the firing-squad-synchronization-problem for n-dimensional rectangles with the general at an arbitrary position. Theoret. Comput. Sci. **19**, 305–320 (1982)
16. Umeo, H.: Recent developments in firing squad synchronization algorithms for two-dimensional cellular automata and their state-efficient implementations. In: AFL, pp. 368–387 (2011)
17. Yamakawi, T., Amesara, T., Umeo, H.: A note on three-dimensional firing squad synchronization algorithm. In: ITC-CSCC, pp. 773–776 (2008)