ELSEVIER

Theoretical Computer Science 168 (1996) 367–404

Theoretical Computer Science

On optimal solutions to the firing squad synchronization problem

Jacques Mazoyer *

Laboratoire de l'Informatique du Paraélisme, Ecole Normale Supérieure de Lyon, 46 Allée d'Italie, 69364 Lyon Cedex 07, France

1. Introduction and definitions

1.1. History

The one-dimensional *firing squad synchronization problem* (FSSP) is to construct a generic automaton of a one dimensional cellular network made of a segment of n identical machines so that, whatever the length n of the segment is,

1. if, at the starting time ($t = 1$ in the following), all finite automata of the cellular network (called cells) are in a quiescent state L and no meaningful piece of information is exchanged, except the leftmost one, called the "general", which is in a special initial state M,

2. then the evolution of the segment is such that, at some time (the firing time $t(n)$), all automata enter simultaneously and for the very first time the firing state F.

One generally considers that the evolution of each automaton is as follows:

State definition. The state at time $t + 1$ of one automaton depends on its own state and the state of its two neighbours at time t.

In this framework, the problem was stated by Moore [9]. First solutions (using about $3n - 2$ and, more generally, $(2 + \frac{p}{q})n$ time units) were described by Minsky and MacCarthy [8]. Goto first discovered a minimal time solution, using $2n - 2$ time units. Minimal time solutions with a little number of states are due to Waksman [15] (in 1966 with 16 states), Balzer [1] (in 1967 with 8 states) and Mazoyer [5] (in 1986 with 6 states). Observe that Yunes [16] has pointed out a non minimal state solution with 7 states in 1994.

* E-mail: mazoyer@lip.ens-lyon.fr.

PII S0304-3975(96)00084-9

In his paper, Balzer set the following optimality problem:

State optimality. What is the minimal number of states needed to solve the FSSP in minimal time?

First, we observe that the statement of the problem already involves three states (L, M and F). It is easy to convince ourself that there does not exist a 3 states optimal time solution (try to design it!). Balzer and, recently, Yunes have shown that there does not exist a 4 state optimal state solution. It is possible that a computer tries all possible 4 states automata on lines of little length and concludes that none of them is a solution. Unfortunately, the study of all possible 5 states automata is not possible: with a today computer this would take thousands of years. Thus the Balzer's question is now: Does then exist a 5 states minimal time solution? We observe that, if this question clearly has no practical interest, its answer will use new knowledge on how evolves a cellular automaton.

In this paper, we do not study Balzer's question. We only aim to put in light some facts:

1. The set of all solutions (or of all minimal time solutions) of the FSSP is not simple. We prove in 2 that it is not recursively enumerable.

2. It is easy to synchronize with few information. For that we shall distinguish the state of an automaton from the message that it receives from its two neighbours.

In the remaining of this introduction, we set up the basic definitions that we need. In particular, we define various constraints on the information flow.

1.2. Standard definitions

In this section, we give a formal definition of a cellular automaton in case each automaton knows the state of its two neighbours.

Definition 1. (1) A cellular automaton $\mathscr{A}$ is a couple (Q, δ) where Q is a finite set, called the states set of $\mathscr{A}$, and δ is a function from Q^3 into Q. The function δ is the local transition function.

(2) A configuration C of the automaton $\mathscr{A}$ is an application from Z in Q. A configuration C evolves to another configuration $C^\star$ so that

$$C^\star(z) = \delta(C(z-1), C(z), C(z+1)).$$

The application Δ defined by $C^\star = \Delta(C)$ is called the global transition function.

Thus, starting from an initial configuration C_0 (at time 0), the net evolves through configurations $C_t = \Delta^t(C_0)$.

Now we define the usual (linear) FSSP.

Definition 2. The state FSSP is to design a cellular automaton $\mathscr{A} = (Q, \delta)$ with the particular additional syntactical properties:

(i) Four distinguished states $(L, M, F, !)$ belongs to Q.

(ii) State L is the quiescent state. It satisfies $\delta(L,L,L) = L, \delta(L,L,!) = L$ and $\delta(!,L,L) = L$.

(iii) State ! is the *outside* state. It satisfies: $\forall q_1, q_2 \in Q$, $\delta(q_1, !, q_2) = !$.

(iv) State M is the *general* state and state F is the *Fire*.

such that, starting from the initial configuration $C[n]$ (the notation $[n]$ indicates that the significant part of the line has length n) defined by:

(a) $\forall z \leqslant 0$, $C[n](z) = !$,

(b) $\forall z \geqslant n+1$, $C[n](z) = !$,

(c) $C[n](1) = M$,

(d) $\forall z \in \{2,\ldots,n\}$, $C[n](z) = L$.

the evolution of the configuration $C[n]$ is such that, for some time $t(n)$,

(a) $\forall z \in \mathscr{Z}$, $\forall t \in \{1,\ldots,t(n)-1\}$, $C[n]_t(z) \neq F$,

(b) $\forall z \in \{1,\ldots,n\}$, $C[n]_{t(n)}(z) = F$.

We remark that in Definition 2, we have replace a segment of cells by a line and have introduced a new state (namely !) in order to delimit the meaningful segment. We observe also that this new state (!) does not enter in the account of the states, but it enters in the domain of the function δ.

Usually, we represent the evolution of a segment of cells as depicted in Fig. 1: the cells are in abcisses and the time runs up, we do not indicate the state !. Such a representation is called a space–time diagram (by *states*).

1.3. Information flow

Reading the Minsky's paper [8], we see that he wishes to distinguish the number of states from the messages got by cells. Thus, following his point of view, we modify the condition [**State definition**] in order to allow various information flows between automata, either bigger or lesser than to convey full information about the sole states:

Two way information flow definition. The state at time $t+1$ of one automaton depends on its own state and on information sent by its two neighbours at time t.

We do not distinguish the sets of information going from left to right or from right to left. This leads us to the following definitions.

Definition 3. (1) A cellular automaton with information flow $\mathscr{A}$ is a triplet (Q, J, δ), where Q is a finite set, called the states set of $\mathscr{A}$, J is a finite set, called the set of information, and δ is a function from $J \times Q \times J$ into $J \times Q \times J$. The function δ is the *local* transition function.

(2) A configuration C of the automaton $\mathscr{A}$ is an application from $\mathscr{Z}$ in $J \times Q \times J$. A configuration C evolves to another configuration C^* such that

$$C^*(z) = \delta(j_{r,1}, q, j_{l,2})$$

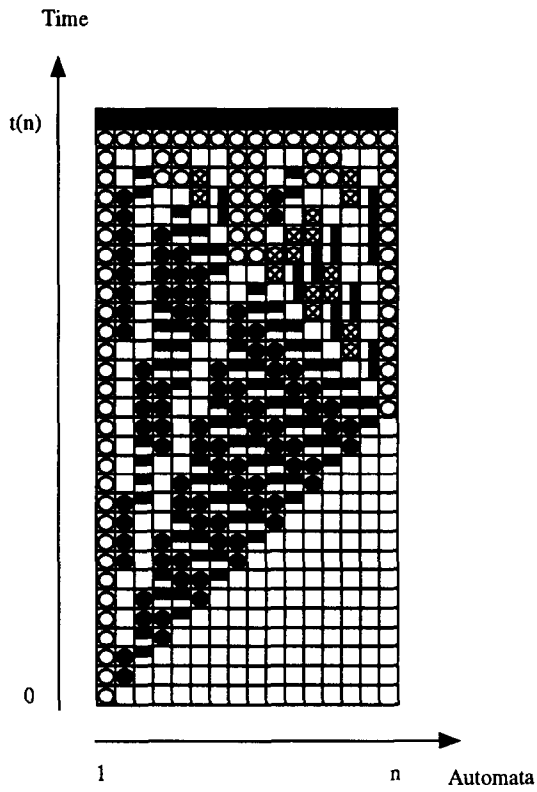


Fig. 1. Space–time diagram of the synchronization of a segment of 16 automata using Balzer's solution.

if $C(z-1)=(j_{l,1},q_1,j_{r,1})$, $C(z)=(j_l,q,j_r)$ and $C(z+1)=(j_{l,2},q_2,j_{r,2})$. The value j_l (j_r) is the information sent to the right (left) neighbour. The application Δ defined by $C^\star=\Delta(C)$ is called the *global* transition function. Thus, starting from an initial configuration C_0 (at time 0), the net evolves through configurations $C_t=\Delta^t(C_0)$.

Definition 4. The *information flow FSSP* is to design a cellular automaton $\mathscr{A}=(Q,J,\delta)$ with the particular additional syntactical properties:

(i) Four distinguished states $(0,\vec{1},F,!)$ belong to Q.

(ii) Three distinguished information $(0,1,!)$ belong to J.

(iii) State 0 is the quiescent state. It satisfies $\delta(0,0,0)=(0,0,0)$, $\delta(0,0,!)=(0,0,0)$ and $\delta(!,0,0)=(0,0,0)$.

(iv) State ! is the outside state. It satisfies: $\forall j_1,j_2\in Q$, $\delta(j_1,!,j_2)=(!,!,!)$.

(v) State $\vec{1}$ is the general state and state F is the *Fire*.

(vi) Information ! is the outside information, 0 is the *null* information and 1 is the *first significant* information.

such that, starting from the initial configuration $C[n]$ (of length n) defined by:

(a) $\forall z\leqslant 0$, $C[n](z)=(!,!,!)$,

(b) $\forall z\geqslant n+1$, $C[n](z)=(!,!,!)$,

(c) $C[n](1) = (1,\vec{1},1)$,
(d) $\forall z \in \{2,\ldots,n\},\ C[n](z) = (0,0,0)$.
the evolution of the configuration $C[n]$ is such that, for some time $t(n)$,
(a) $\forall z \in \mathscr{Z},\ \forall t \in \{1,\ldots,t(n)-1\},\ C[n]_t(z) \neq (0,F,0)$,
(b) $\forall z \in \{1,\ldots,n\},\ C[n]_{t(n)}(z) = (0,F,0)$.

We remark that, as in Section 1.2, we have replaced a segment by a line in Definition 4, using a special state (!) and a special information (!). In the following, we may relax the condition (a) (the synchronization is set up only both by the state and the information of the general) on the initial line, supposing that the value of $C[n](1)$ is either $(0,\vec{1},0)$ (the synchronization is set up only by the state of the general) or $(1,0,1)$ (the synchronization is set up only by the information of the general).

Now, the question of optimality becomes:

State and information flow optimality. What is the minimal number of states "and" information flow needed to solve the FSSP in minimal time?

In this paper, we present a minimal time solution where J is minimal ($J = \{0,1,!\}$) and Q has 58 states.

Coming back to Minsky's ideas, we observe that he introduced the notion of channels: a channel is the number of digits needed to describe an element of J. More formally, the number of channel is $\lceil \log_2 |J-1| \rceil$. We observe that both Balzer's [1] and Mazoyer's [5] solutions have 3 channels. The solution presented here has only one channel.

Usually, we represent the evolution of a segment of automata when $J = \{0,1,!\}$ as depicted in Fig. 2: the cells are in abcisses and the time runs up, we do not indicate the state and information flow !. Such representation is called a space–time diagram (by *states and information*).

In Section 5, we study the opposite: few states and a large amount of information flow. The result is that synchronization cannot be achieved with two states (the quiescent one and the fire), but it is possible with three states (the quiescent one, the general and the fire).

1.4. Constraints

As there exists a minimal time solution with only one channel, we strengthen the condition in order to get limits of the synchronization process.

Our reinforcement is to allow only one-way channel. Fig. 3 illustrates this notion. The (previous) two-ways channel may be viewed as two electrical wires: one of them carrying electrons from right to left the other one from left to right. We define one-way channel as only one wire carrying electricity in both directions. Thus, if automaton k or $k+1$ has emitted a digit 1, both receive the digit 1. Definition 3 becomes Definition 5 and Definition 4 remains unchanged.

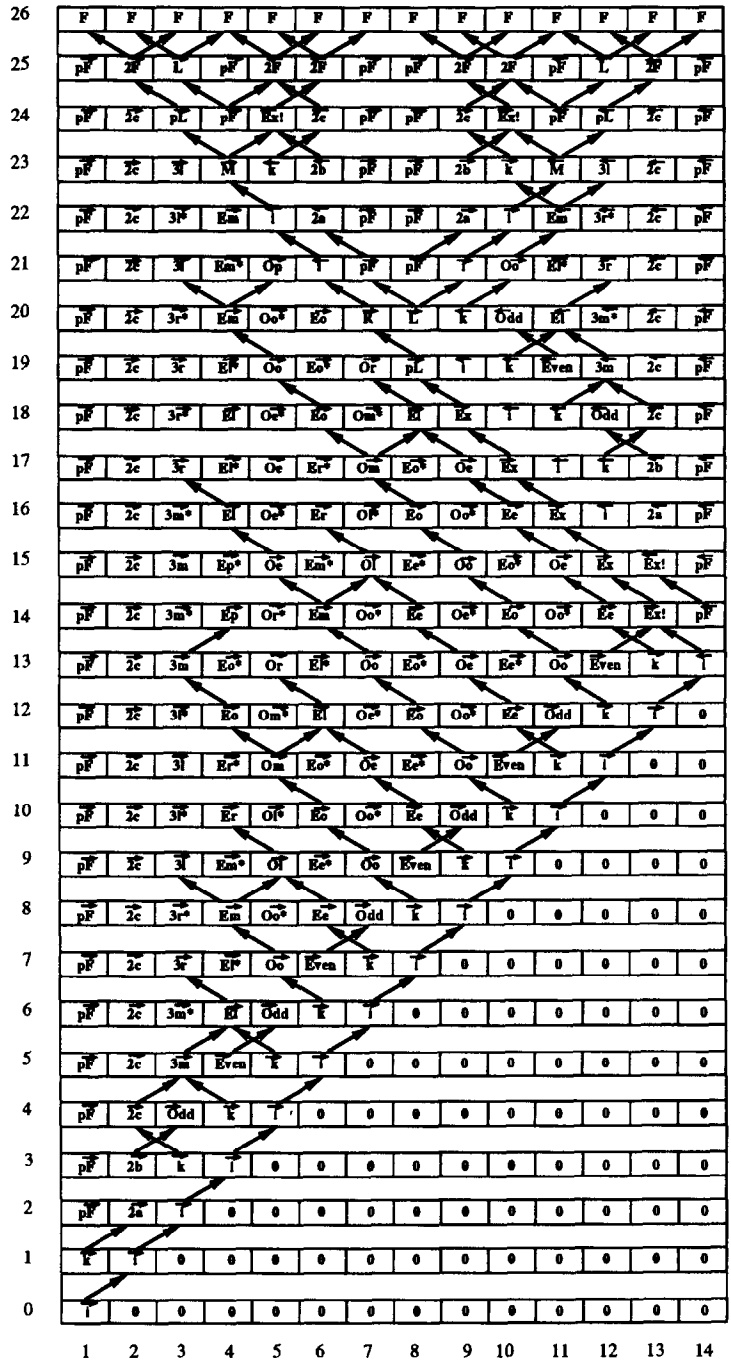


Fig. 2. Space–time diagram of the synchronization of a segment of 14 automata using the solution presented in Section 3.

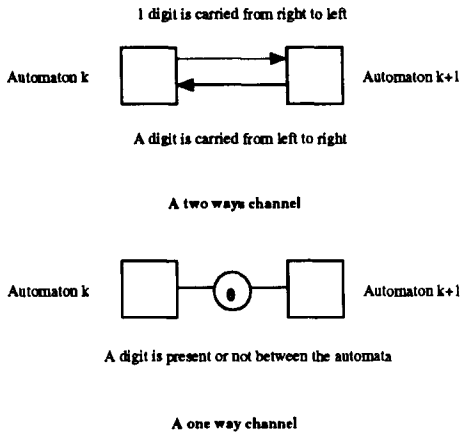


Fig. 3. One-way and two-ways channels.

Definition 5. The same as Definition 3 except the definition of C^* which is now:

$$C^*(z) = \delta(\mathrm{Max}(j_{r,1}, j), q, \mathrm{Max}(j_{l,2}, j))$$

if $C(z-1) = (j_{l,1}, q_1, j_{r,1})$, $C(z) = (j_l, q, j_r)$ and $C(z+1) = (j_{l,2}, q_2, j_{r,2})$.

In Section 4, we shall see that:

- the minimal time is not always $t(n) = 2n - 2$, but it remains $2n - 2$ except for a little finite number of values of n.
- Such a minimal time solution exists with 230 states. Its synchronization time is: $t(2) = 3$, $t(3) = 6$, $t(4) = 8$, $t(5) = 8$, $t(6) = 12$ and, for $n \geqslant 7$, $t(n) = 2n - 2$.

2. Structure of the set of solutions

First, we prove that the set of the solutions (in minimal time or not) to the FSSP is not recursively enumerable. Thus, the questions of optimality are not – a priori – obvious.

Theorem 1. *The sets of the solutions and of minimal time solutions to the FSSP are not recursively enumerable.*

Proof. The idea of the proof is very simple: we suppose that the set of solutions (in minimal time or not) is recursively enumerable; we observe that the set of nonsolutions is obviously recursively enumerable; and we deduce that the set of the nonsolutions is recursive. Under this assumption, we solve the halting problem. Let $\mathscr{A}$ be one cellular automaton solution (in minimal time or not) to the FSSP, we define a family $\{\mathscr{A}\}_i$ of cellular automata such that $\{\mathscr{A}\}_i$ is a solution for the FSSP if and only if the ith

Turing machine halts on the ith input string. In this case, we get an algorithm which solves the halting problem, thus the contradiction.

To construct the family $\{\mathscr{A}\}_i$ uses Smith's simulation [12] of a Turing machine by a one-dimensional cellular automaton. On input a segment of length n (initial configuration ${}^{\omega}!M\ \underbrace{L\ldots L}_{n-1 \text{ times}}!^{\omega}$), automata $\{\mathscr{A}\}_i$ have two concurrent behaviours: they synchronize the segment of length n and simulate the ith Turing machine on the ith input.

1. If the simulation takes more than n cells of the tape, the simulation stops.

2. If the Turing machine halts, then the first behaviour (synchronization of the segment) stops.

3. When the synchronization is obtained, it stops the simulation of the Turing machine. By this way, if the ith Turing machine halts on the ith input, it halts at some time $\theta(i)$ and a line of length greater than $\theta(i)$ is not synchronized (2); but if the Turing machine does not halt, then synchronization is obtained whatever is the length of the segment.

4. And the contradiction is got.

It only remains to describe $\{\mathscr{A}\}_i$ in details.

(i) *Smith's simulation*

Fig. 4 illustrates this simulation. The automaton $\mathscr{A}_M$ which simulates the Turing machine $\mathscr{M}$ of alphabet A and states Q has $A\times(Q\cup\{\emptyset\})$ as set of states. Its state function δ^* is defined by (δ is the transition function of M):

- $\delta^*((a_\alpha,\emptyset),(a_\beta,\emptyset),(a_\gamma,\emptyset)) = (a_\beta,\emptyset)$ (*the read write head is not in the neighbour of the cell and no simulation is performed*),
- $\delta^*((a_\alpha,\emptyset),(a_\beta,q_\ell),(a_\gamma,\emptyset)) = (a_\beta^\sharp,q_\ell^\sharp)$ if $\delta(q_\ell,a_\beta) = (q_\ell^\sharp,a_\beta^\sharp,\mathrm{St})$ (*the read write head is on the cell and it does not move; thus the cellular automaton changes the value of the letter and updates the value of the state of M*),
- $\delta^*((a_\alpha,\emptyset),(a_\beta,q_\ell),(a_\gamma,\emptyset))=(a_\beta^\sharp,\emptyset)$ if $\delta(q_\ell,a_\beta)=(q_\ell^\sharp,a_\beta^\sharp,\mathrm{Le})$ or $\delta(q_\ell,a_\beta)=(q_\ell^\sharp,a_\beta^\sharp,\mathrm{Ri})$ (*the read write head is on the cell and it moves; thus the cellular automaton only changes the value of the letter*),

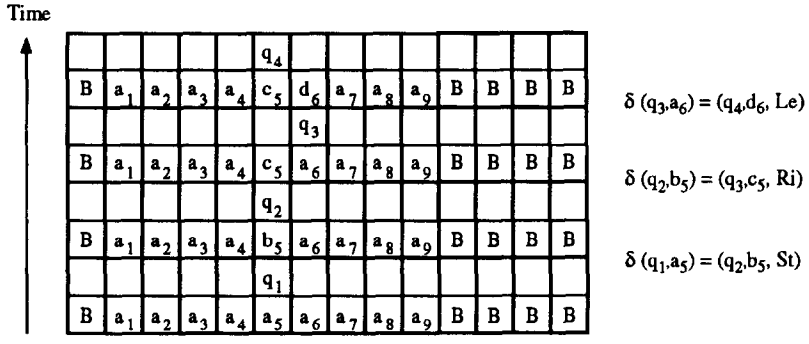


Fig. 4. Smith's simulation of a Turing machine by a cellular automaton.

- $\delta^*((a_\alpha,\emptyset),(a_\beta,\emptyset),(a_\gamma,q_\ell)) = (a_\beta,q_\ell^\sharp)$ if $\delta(q_\ell,a_\gamma) = (q_\ell^\sharp,a_\gamma^\sharp,\mathrm{Le})$ (*the read write head is on the right neighbour cell and it moves to the left; thus the cellular automaton takes it with the new state of M*),
- $\delta^*((a_\alpha,q_\ell),(a_\beta,\emptyset),(a_\gamma,\emptyset)) = (a_\beta,q_\ell^\sharp)$ if $\delta(q_\ell,a_\alpha) = (q_\ell^\sharp,a_\alpha^\sharp,\mathrm{Ri})$ (*the read write head is on the left neighbour cell and it moves to the right; thus the cellular automaton takes it with the new state of M*).

(ii) *Setting the input of the Turing machine*

Let A be the alphabet on which work our Turing machine and $a_{i(1)}\dots a_{i(\ell(i))}$ be the ith input string, we define a cellular automaton $\mathscr{A}_{\mathrm{input}(i)}$ of set of states $A\cup\{L,M,!\}$ ($!,M$ and L have the same meaning as in the FSSP) defined by:

- $\delta(!,M,L) = a_{i(1)}$,
- $\forall j\in\{1,\dots,i(\ell(i))-1\},\ \delta(a_{i(j)},L,L) = a_{i(j+1)}$,
- $\forall j\in\{1,\dots,i(\ell(i))-1\},\ \delta(a_{i(j)},L,!) = a_{i(j+1)}$,
- $\forall j\in\{1,\dots,i(\ell(i))-1\},\ \delta(L,a_{i(j)},L) = a_{i(j)}$,
- $\delta(a_{i(\ell(i))},L,L) = M$,
- for all other cases, $\delta(\alpha,\beta,\gamma) = \beta$.

The automaton $\mathscr{A}_{\mathrm{input}(i)}$ has on any initial configuration of the form ${}^\omega!M\underbrace{L\dots L}_{n-1\text{ times}}!^\omega$, the following behaviour: it sets up the state of the jth automaton to $a_{i(j)}$ at time j. Thus:

- if $\ell(i)\leqslant(n+1)$, the configuration obtained at time $\ell(i)$ is

$$ {}^\omega!a_{i(1)}\dots a_{i(\ell(i))}M\underbrace{L\dots L}_{n-\ell(i)-1\text{ times}}!^\omega $$

- if $\ell(i)>(n+1)$, the configuration obtained at time n is ${}^\omega!_{a_i(1)}\dots a_{i(n)}!^\omega$

(iii) *Simulation of the ith Turing machine on the ith input string*

Let $M(i)$ be the ith Turing machine. We define a cellular automaton $\mathscr{S}_i$ by:

- The set of states of $\mathscr{S}_i$ is $Q_1\times Q_2\times Q_3$ where:
  Q_1 is the set of states of $\mathscr{A}_{M(i)}$ (point i)
  Q_2 is the set of states of $\mathscr{A}_{\mathrm{input}(i)}$ (point ii)
  Q_3 is the set of states of a minimal time solution for the FSSP.
- The behaviour of the transition function of $\mathscr{S}_i$ is the following:

– If the component in Q_3 is not F (the Fire), the behaviour is the one of $\mathscr{A}_{\mathrm{input}(i)}$ on the second component, in order to build the entry for $M(i)$.
– If the component in Q_3 is the Fire, the behaviour is the one of $\mathscr{A}_{M(i)}$ on the first component (simulation of $M(i)$).
– On the third component, the synchronization is obtained by this way: the state M of the second component is understood as the outside. If the state M is never seen, then the synchronization never occurs (all the segments enter a new state N when reaching !).
– In addition when on the third component the triplet of states $(!,F,F)$ appears, the second component of the first component becomes the initial state of $M(i)$.

By this way, on the initial configuration of length n

$$^{\omega}(!,!,!)((\emptyset,\emptyset),L,M)\underbrace{((\emptyset,\emptyset),L,L)\ldots((\emptyset,\emptyset),L,L)}_{n-1 \text{ times}}(!,!,!)^{\omega}$$

at time $2n-2$:

- if $\ell(i)\geqslant(n+1)$, at time $2n-2$, the following configuration is obtained

$$^{\omega}(!,!,!)((a_{i(1)},q_0),a_{i(1)},F)((a_{i(2)},\emptyset),a_{i(2)},F)\ldots((a_{i(\ell(i))},\emptyset),a_{i(\ell(i))},F)$$
$$((\emptyset,\emptyset),M,L),\underbrace{((\emptyset,\emptyset),L,L)\ldots((\emptyset,\emptyset),L,L)}_{n-\ell(i)-1 \text{ times}}(!,!,!)^{\omega}$$

  and then $\mathcal{A}_{M(i)}$ simulates the ith Turing machine on the ith input string.
- if not, at time $2n-2$, the following configuration is obtained:

$$^{\omega}(!,!,!)(a_{i(1)},\emptyset),(a_{i(1)},N)\ldots((a_{i(\ell(i))},\emptyset),a_{i(\ell(i))},N)(!,!,!)^{\omega}$$

and no simulation of $M(i)$ is performed.

(iv) Finally, the construction of $\{\mathcal{A}\}_i$ is easy to complete. Its states have two components: one is state of a solution for the FSSP, the second one is a state of $\mathcal{S}_i$ (point iii)). Its transition function works as previously mentioned. □

Thus, it is impossible to describe all the minimal time solutions to the FSSP with a finite number of words. In fact, in [7], we have described a lot of solutions. Three main features arise:

1. All solutions use a "divide and conquer" strategy. Only the ratio in which the segment is cut changes (it may be any ratio in $[\frac{1}{2},1[$).
2. It is possible to obtain minimal time solution using all the transitions of a non-minimal time solution.
3. Non minimal synchronization times have some closure properties.

3. A minimal time solution with a minimal number of two-way channels

The following described automaton has been tested on a computer for segments from length 2 up to 1000. It is possible to prove that it is sufficient to test it for segments from 2 up to 300. We do not give this (formal and tedious) proof.

3.1. General strategy

The general strategy to obtain a solution of the FSSP is to break the line at its $\frac{p}{q},(\frac{p}{q})^2,(\frac{p}{q})^3,\ldots$ (with $\frac{p}{q}\in[\frac{1}{2},1]\cap\mathcal{Q}$) and then to synchronize each new created subline by a recursive call to one solution (the expected solution itself or another one). In [5], the value of $\frac{p}{q}$ is $\frac{2}{3}$ and each subline is synchronized by a recursive call to the constructed solution. In [1], the value of $\frac{p}{q}$ is $\frac{1}{2}$ and each subline is synchronized

by a recursive call to the "image" solution (the solution itself interchanging roles played by the left and the right in order to get the synchronization from the right end cell). Here, we aim to minimize the information flow and observing that in the Mazoyer's strategy [5] each cell must know the remainder of its location by 3 and that we need 2 information to give it this knowledge, we choose the Balzer's strategy [1]. If Balzer's strategy increases the number of states (by the "image" solution), we do not mind here because we are only interested in the information flow. Thus, we begin to describe the Balzer's strategy in some details. Fig. 1 shows the synchronization of a short segment and Fig. 5 illustrates his solution when the number of cells is so large that we may identify $\mathscr{Z}^2$ to $\mathscr{R}^2$.

1. **Breaking the segment at its** $\frac{1}{2}, (\frac{1}{2})^2, \ldots$. At time 0, the general sends a signal at maximal speed (one cell per unit of time), called the "*initial wave*", this signal is reflected by the right end and comes back at maximal speed; it reaches the general at $2n-2$ (the minimal synchronization time). A family of signals ("*break signals*" in Fig. 5), appearing as connected waves of white squares on the Fig. 1, is set up. The slopes of these break signals are $2, 2^2, \ldots$ and they are generated on the second cell. To set up such a family of signals is not possible by a finite automaton; they are set up by the whole segment. Every time the break signal of level j (at slope 2^j) moves to the right, it sends a signal (at maximal speed) to the left, called an "*auxiliary signal*".

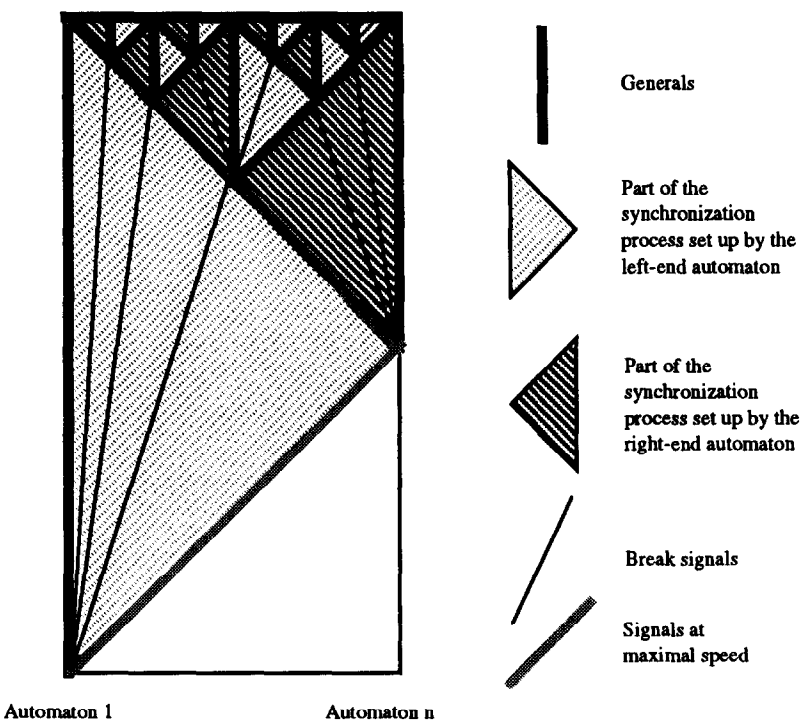


Fig. 5. The Balzer's strategy.

The signal of level $j+1$ moves to the right one time out of two when it receives such an auxiliary signal. By this way, when break signals meet the reflection of the initial wave, the line is cut. But, due to the discrete nature of the problem, a segment of length n is cut on the automaton $\lfloor \frac{n}{2} \rfloor + 1$. If n is odd, this value corresponds to the middle of the segment. If n is even, observing that selecting cell $\lfloor \frac{n}{2} \rfloor + 1$ is to select cell $\lfloor \frac{n}{2} \rfloor$ one time later, we have selected the two cells near of the middle of the segment.

2. **Synchronization of the sublines.** Selecting cells on the path of the reflection of the initial wave in the space time diagram creates new sublines. More precisely, if the initial segment has length n, the first subline is made of cells $\lfloor \frac{n}{2} \rfloor$ to n (if n is even) or $\lfloor \frac{n}{2} \rfloor + 1$ to n (if n is odd). The $(j+1)$th subline is similarly obtained from the jth subline. Then the synchronization of a subline is initiated from its right end: on the path of the reflection of the initial wave if the length of this subline is odd or with a delay of one unit of time if its length is even. We observe that when the synchronization of a new subline is initiated, the left end cell of the subline is not set up, but its right end can know the parity of the (future) subline. This synchronization from the right end is achieved by an image solution.

3. **Completing the synchronization.** The previous process is iterated until all sublines have length 2.

3.2. Breaking the segment

In this section, we set up the exchange of information needed to give to cells the ability to recognize and answer to the reflection of the initial wave. This exchange is described in Fig. 6 in which we present the beginning of segment so much long that we do not see the reflection of the initial wave.

(a) **Recognition of the reflection of the initial wave**

As the initial wave is reflected as soon as possible (minimal time solution), we observe that the cell $n-j$ (of a segment of length n) receives the initial wave at time $j-1$ and its reflection at time $n-1+n-j$. Thus, the number of time units between the arrival of the initial wave and of its reflection is always even $(2(n-j))$. Thus, we choose that any digit 1 reaching a cell before the reflection of the initial wave, reaches it an odd number of times after the initial wave itself. By this way, counting times since the initial wave modulo 2, any cell can recognize the reflection of this initial wave. In the following, we shall mark the states on which the reflection of the initial wave cannot occur by a $\star$: a $\star$-state receiving 1 from its right neighbour does not understand it as the reflection of the initial wave. We call *site* a point of the space–time diagram.

(b) **Eventual break sites**

We list possible knowledges that cells must have in mind if they receive the reflection of the initial wave. According to the previous Section 3, the three following knowledges are needed:

eventual ($\star$) left-end sites. *If I have a not $\star$ state and I receive digit* 1 *from my right neighbour, I know that the broken segment is even and I become the right end*

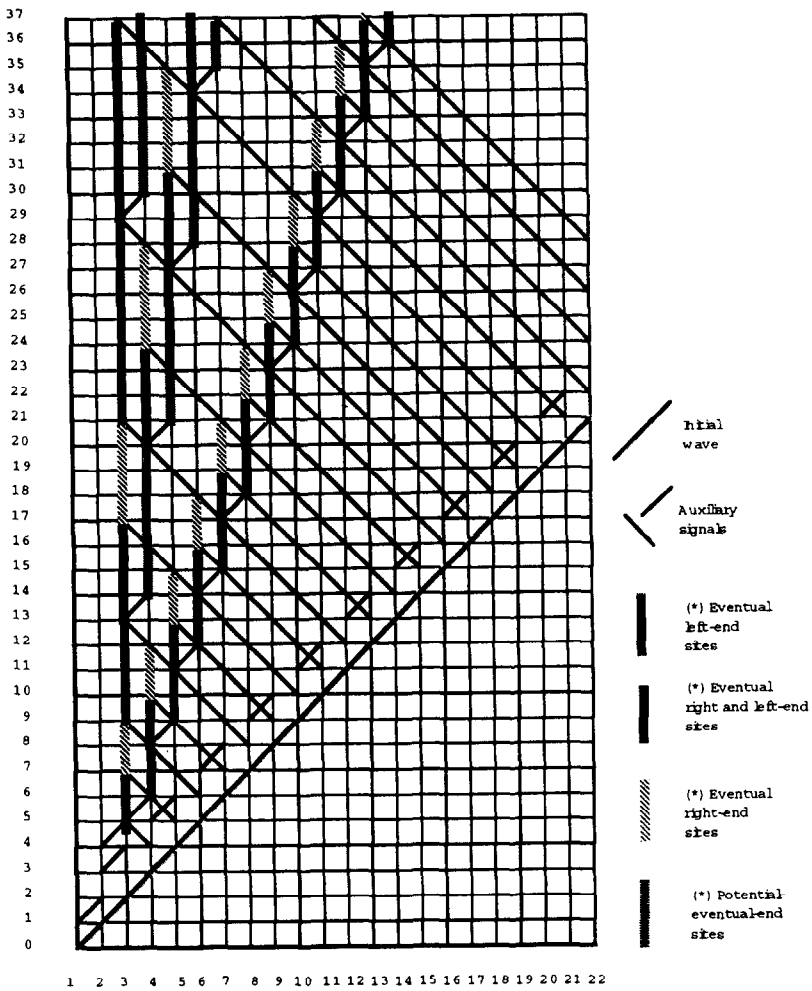


Fig. 6. The information flow setting up breaks of the segment.

of the left new subline. In this case, the cell becomes the general (at the right end) of the new created subline.

eventual (⋆) right and left-end sites. *If I have not a ⋆ state and I receive digit* 1 *from my right neighbour, I know that the broken segment is odd and I become the right end of the left new subline and the left end of the right subline.* In this case, the cell becomes the general (at the right end) of the new created subline and it also becomes the left end of the previous created subline.

eventual (⋆) right-end sites. *If I have not a ⋆ state and I receive digit* 1 *from my right neighbour, I know that the broken segment is even and I become the left end of the right subline.* In this case, the cell becomes the left end of the previous created subline.

All these knowledges are set up in the states of the automata. We observe easily that any cell enters eventual ($\star$) right-end sites (eventual ($\star$) right and left-end sites) when it leaves ($\star$) right and left-end sites (eventual ($\star$) left-end sites). A cell changes its knowledge when it receives digit 1 from its right neighbour on a $\star$ site (an odd number of time after the initial wave). These digits 1 correspond to the auxiliary signals of Section 3. But they cannot be sent by a cell when it receives the initial wave in order to keep the parity of point a). We choose that a cell emits such a signal one unit of time after it receives the initial wave.

It remains the problem to know when receiving such a digit 1 on a $\star$ site, a cell enters eventual ($\star$) left-end sites. This is not quite obvious since sometimes receiving such a digit it must enter these sites and sometimes not. Observing that the same digit 1 coming from the right must put cell j in eventual ($\star$) right-end sites and cell $j+1$ in eventual ($\star$) left-end sites, we choose that, when cell j enters eventual ($\star$) right and left-end sites, it sends a digit 1 to its right neighbour. Conversely, when some cell receives a digit 1 from its left neighbour, it knows that, if it receives digit 1 from its right neighbour on a not $\star$ state, it enters an eventual ($\star$) left-end site. This new knowledge is:

($\star$) Potential-eventual-end sites. *Receiving digit* 1 *from my right neighbour on a* $\star$ *site, I enter eventual* ($\star$) *left-end sites.*

(c) Now we list the knowledges that some cell must have in mind to set up the delay of 1 unit of time if the segment to be broken is even. We observe that the only knowledge needed for the right-end cell is its parity. We also observe that it may know its parity if all cells know their own parity. Thus, we introduce a new needed knowledge:

Parity. *I know the parity of my location in the segment.* If the cell is the right-end of the segment, it sets up a delay of 1 unit of time if and only if its parity is even.

For other automata, we observe that reflections of initial waves corresponding to even (odd) lines may reach them every 4 units of time. Thus, the delay to be set up depends only of the parity of the cell and of the remainder by 4 of the number of times elapsed since the initial wave has reached it. This remainder is known if each cell counts the elapsed time modulo 4. We observe that leaving an eventual ($\star$) right-end site, any cell resets this counter because it concerns a new created subline. Thus we introduce this new knowledge:

Remainder modulo 4. *I know the remainder modulo* 4 *of the time elapsed since I received the initial wave.* This remainder is used both to choose between 1 digits coming from the right and the one understood as the reflection of the initial wave and to select the delay used in the synchronization of the new created subline.

(d) As indicated in Section 3, we must distinguish the second cell on which the break signals are created. We observe that in order to achieve synchronization of segment of length 3, the second cell must know its number before time 3. Thus, we choose to introduce the following knowledge:

Number 2. *I am the second cell in the segment* and to set up it in the following way: The left-end cell (which knows its location receiving ! of its left neighbour in the

outside) sends digit 1 to its right neighbour one unit of time after it sends the initial wave.

Now we must indicate to any cell its parity. To give this knowledge, the second cell sends to its right neighbour digit 1 as soon as possible: at time 3 because if it sends this digit at time 2, automaton 2 would believe that it is the second. We iterate the process. Any cell, receiving digit 1 from its left neighbour 2 times after the initial wave knows that its parity is odd and sends digit 0 to its right neighbour. Similarly, any cell, receiving digit 0 from its left neighbour 2 times after the initial wave knows that its parity is even and sends digit 1 to its right neighbour.

(e) We observe that the second cell does not follow the process described in (b). Receiving the reflection of the initial wave, it always becomes the general (at the right end) of a new created subline of length 2. Thus we choose to initiate the process of (b) on the third cell.

Now, we study the behaviour of the third cell.

- If the segment has length 4, the third cell becomes the left-end of a new subline of length 2 and it does not become the right-end general of a new subline.
- If the segment has length 5 the third cell becomes the right-end general of a new subline of length 3 and the left-end of a new subline.
- If the segment has length 6, the third cell becomes the right-end general of a new subline of length 2 and it does not become the left-end of a new subline.
- If the segment has length 7, is created a new subline of length 4.
- If the segment has a length greater than 7, a new subline of length 4 or 5 or 6 is created.

Thus, the third cell is always in a eventual ($\star$) end site. It must enter eventual ($\star$) right and left-end sites at time 4 and, then, receiving digit 1 from its right neighbour it enters the following eventual ($\star$) sites in the order of point (b). Thus, we need the following information:

Number 3. *I am the third cell.* At time 4, the third cell enters eventual ($\star$) right and left-end sites and, then, leaving eventual ($\star$) right-end sites, it enters eventual ($\star$) left-end sites.

We observe that the third cell knows its parity at time 3. We choose to give it its location by: the second cell (which knows its location at time 2 and sends (at time 3) to its right neighbour a "parity" digit 1) sends digit 1 to its right neighbour at time 4 (as soon as possible). Thus, the third cell is the only one which receives digits 1 two and three times after the initial wave. Finally, we observe that distinguishing what happens 2 times after the initial wave, receiving a digit 1 from its left neighbour may be understood as knowledges of points (e) or (b).

3.3. States of the break process

In this section, we describe the states used to set up the general and the break process described in Section 3.2. We use two conventions: to mark by a $\star$ states on

which the reflection of the initial wave cannot occur (see Section 3.2) and to mark all states by an arrow $\rightarrow$, indicating that the initial wave runs from left to right.

States of the general are shown in Fig. 7. The general always receives ! from its left neighbour (in the outside). Starting from state $\vec{\imath}$, emitting 1 to the right (initial wave), it enters state $\vec{k}$, emiting 1 to the right (marking the second automaton), and, then, it enters and remains in state $p\vec{F}$ until it receives the reflection of the initial wave in order to be put in F. We observe that the general does not need to count modulo 4.

The states involved in Section 3.2 are shown in Fig. 8. In this figure, we also have indicated when the reflection of the initial wave can occur; and we have distinguished when, in this case, the cell becomes a right general (set of states R), a left-end (set of states L) or the both (set of states M).

The two time units after the initial wave (states $\vec{\imath}$ and $\vec{k}$) are used to set up the second cell and the parity of the cell (points c) and d) of Section 3.2). Thus, three times after the initial wave, we introduce the states $2\vec{a}$, $\vec{O}$dd and $\vec{E}$ven having these knowledges.

The states of the second cell begin by the letter "2". As mentioned in point (c) of 3.2, it sets up the third cell and then waits the reflection of the initial wave. We do not have to know where this reflection occurs because the second cell always becomes a general (point (e) of 3.2). Only 3 states are necessary: $2\vec{a}, 2\vec{b}$ and $2\vec{c}$.

The third cell in state $\vec{O}$dd, receives digit 1 from the second; at this time it enters one state among states marked by the letter "3". This third cell follows the process of

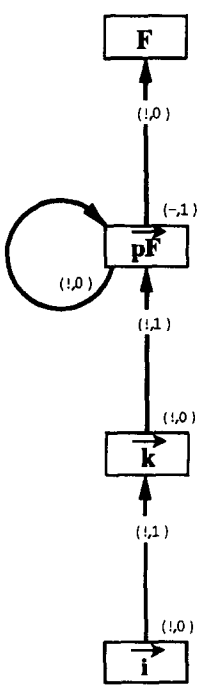


Fig. 7. States of the general.

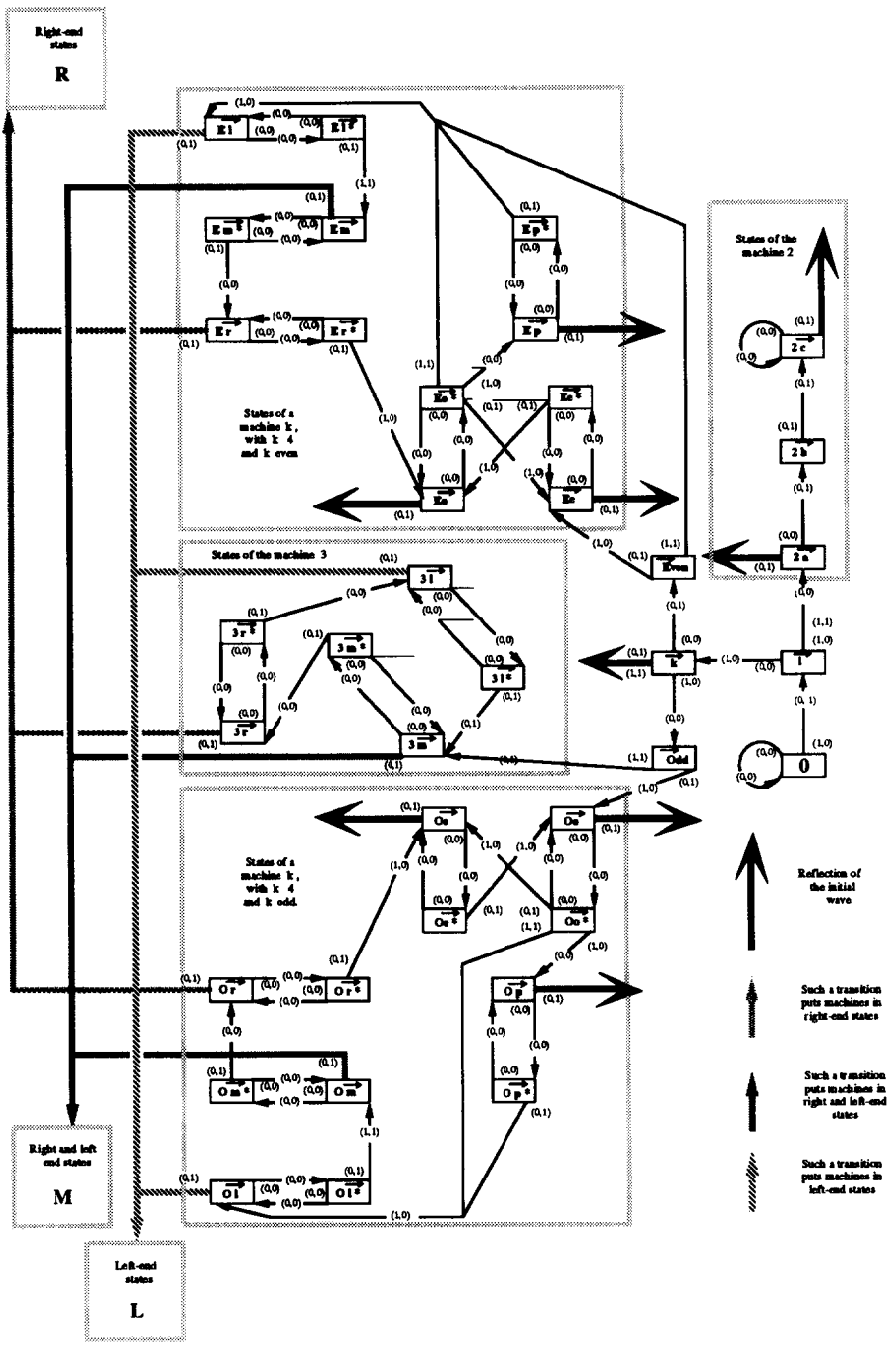


Fig. 8. States of the break process.

point (b) modified by point (e). 6 states are used:

$3\vec{m}$ and $3\vec{m}^{\star}$ (eventual ($\star$) right and left-end states),

$3\vec{r}$ and $3\vec{r}^{\star}$ (eventual ($\star$) right-end states),

$3\vec{l}$ and $3\vec{l}^{\star}$ (eventual ($\star$) left-end states).

States of an even (odd) cell are marked by "*E*". ("*O*"). The process of point b) of 3.2 (eventual and potential-eventual end sites) is set up using the following 2×8 states:

$E\vec{p}$ and $E\vec{p}^{\star}$; $O\vec{p}$ and $O\vec{p}^{\star}$ (($\star$) potential-eventual-end states),

$E\vec{l}$ and $E\vec{l}^{\star}$; $O\vec{l}$ and $O\vec{l}^{\star}$ (eventual ($\star$) left-end states),

$E\vec{m}$ and $E\vec{m}^{\star}$; $O\vec{m}$ and $O\vec{m}^{\star}$ (eventual ($\star$) right and left-end states),

$E\vec{r}$ and $E\vec{r}^{\star}$; $O\vec{r}$ and $O\vec{r}^{\star}$ (eventual ($\star$) right-end states).

These states mark the auxiliary signals of point 1) of Section 3.1. Between two such signals and only in this case, one cell needs to know is remainder modulo 4. This is due to the fact that receiving in an potential-eventual-end state the reflection of the initial wave, one cell knows that the delay it has to bring up is null because its left neighbour becomes both a general and a left-end indicating that the segment is odd. This remainder is set up using 2×4 states:

$E\vec{o}$ and $E\vec{o}^{\star}$; $O\vec{o}$ and $O\vec{o}^{\star}$ when the time since the initial wave is even,

$E\vec{e}$ and $E\vec{e}^{\star}$; $O\vec{e}$ and $O\vec{e}^{\star}$ when the time since the initial wave is odd.

Thus all the process of Section 3.2 is set up with 37 states.

3.4. Completing the synchronization

Now, we study what happens when the reflection of the initial wave reaches a cell.

Fig. 9 is Fig. 8 on which we have added the states corresponding to the end cell.

- When the segment to cut is even and when some cell in an eventual right-end state receives the reflected initial wave, it becomes the general (at the right) of a new created subline. Thus it enters a new state ($\vec{R}$), sending the reflected initial wave (by digit 1) to its left neighbour and nothing (digit 0) to its right neighbour. In state $\vec{R}$, it sends to its left neighbour digit 1, indicating to it that it is the second cell of the new created subline, and enters state $p\vec{F}$. In state $p\vec{F}$, it waits until it receives the reflection of the initial wave of the new subline (at its left) and, then, it enters the Fire.
- When the segment to cut is odd and when some cell in an eventual right- and left-end state receives the reflected initial wave, it becomes the general (at the right) of a new created subline and the left-end automaton of the new subline created at its right. But this new subline (at its right) is synchronized with a null delay (the segment is odd); thus, our cell will also become the general (at the left) of the first subline created during the synchronization of its right subline. And, our cell must act as if it was a general for its both sublines (at its right and at its left). It acts as previously sending digits 1 both at its right and at its left. This is achieved by the new state $\vec{M}$ (state $p\vec{F}$ is identified with the previous case).

- When the segment to cut is even and when some cell in an eventual left-end state receives the reflected initial wave, it becomes the left-end cell of the subline at its right. It enters state $p\vec{L}$ transmitting the reflected initial wave to its left neighbour and digit 0 to its right one. As described previously, at the next time, it acts as a general for its right subline. This is done with a new state $\vec{L}$.
- The behaviour of the third cell follows the previous rules. But the second one has a special behaviour. As said in point (d) of 3.2, the second cell, receiving the reflected initial wave, will be in Fire after one unit of time. Thus, it waits one time in state $2\vec{F}$ and also acts as the middle automaton of an odd segment, sending digit 1 to its both neighbours.

Fig. 10 depicts the states of the right-end cell. It only reflects the initial wave and then waits the reflection of the reflected initial wave to enter the Fire.

In Fig. 9, we have also indicated the knowledge of the parity of the length of the segment, the cell becomes an end of which when it receives the reflected initial wave. In states $O\vec{o}$, $O\vec{p}$, $E\vec{o}$ and $E\vec{p}$, it knows that the segment is odd and that no delay must be set up. In states $O\vec{e}$ and $E\vec{e}$, it knows that the segment is even and that a delay of 1 unit of time must be set up. In state $\vec{k}$, it knows that the segment is even and has possibly length 2.

Fig. 11 shows states involved to set up the delay. In the general case (the segment is even but has not length 2) this delay is set up using the states $E\overleftarrow{x}!$ (by the cell number 2, corresponding to an cell reached by the reflection of the initial wave in state $\vec{k}$) and $E\overleftarrow{x}$ by the others.

Before to set up new created sublines of length 2, we observe that to obtain our final automaton, we must duplicate all states according to the fact that the synchronization is initiated from the left to the right (see point (2) of 3.1).

When a cell is in state $E\overleftarrow{x}!$, if the new subline has a length greater than 2, it receives digits 0 from its two neighbours and enters state $2\vec{a}$. If the new subline has length 2, in state $E\overleftarrow{x}$, it receives digit 1 from its left neighbour and the following time it will receive digits 1 from its both neighbours, entering the fire the next time. Thus, receiving 1 from its left neighbour, it enters $2\overleftarrow{S}$ and then $2\vec{F}$. We observe that we may identify states $E\vec{x}$ and $2\overleftarrow{S}$.

Finally, we obtain the automaton of Fig. 12. This automaton has 92 states. Its evolution on a segment of 14 automata is shown in Fig. 2.

3.5. *Comments*

The automaton, previously described and depicted in Fig. 12, uses a minimal amount of information flow but a large amount of states. Can we reduce its number of states? First of all, we observe that it is incompletely specified and when no transition is indicated this means that any transition may occur. Thus to minimize it is $\mathcal{NP}$-complete. In the following, we present a possible minimization obtaining a final automaton with (only) 58 states (shown in Fig. 13).

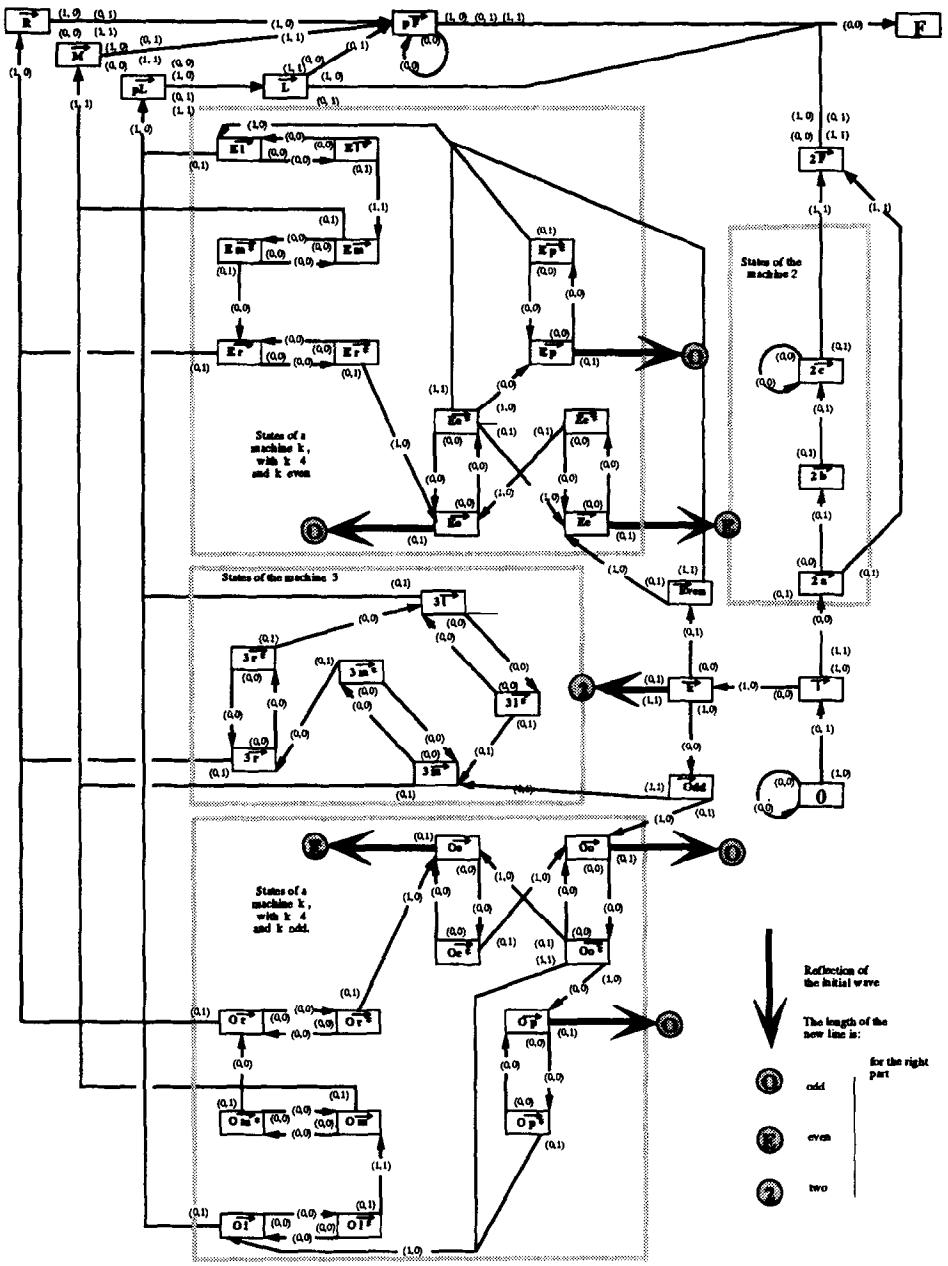


Fig. 9. States of the break process and of the end automata.

1. The two possible second cells (when the synchronization is initiated from the left or the right) do not need to know from which end the synchronization started; they see from which side arises the reflected initial wave. This leads us to identify states $2\vec{b}$ and $2\overleftarrow{b}$, $2\vec{c}$ and $2\overleftarrow{c}$, $2\vec{F}$ and $2\overleftarrow{F}$.

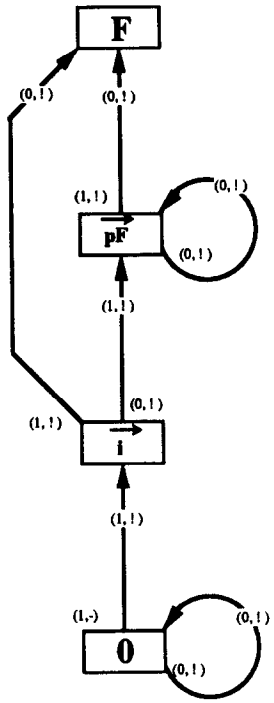


Fig. 10. States of the right end automaton.

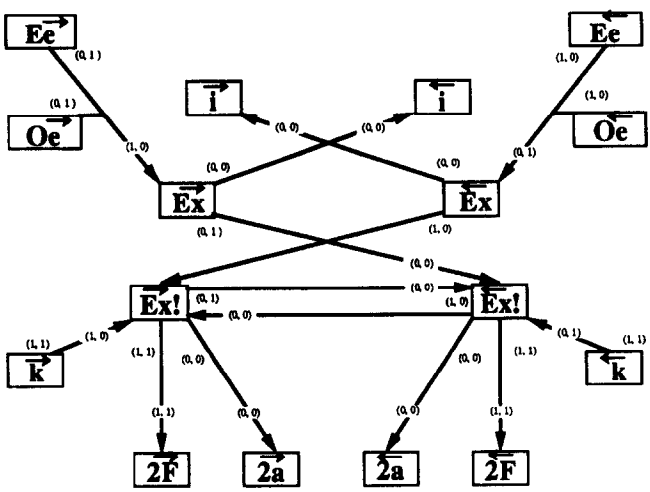


Fig. 11. Setting up the delay.

2. The same argument holds for the two third cells. This leads us to identify states: $3\vec{r}$ and $3\overleftarrow{r}$, $3\vec{r}^{\star}$ and $3\overleftarrow{r}^{\star}$, $3\vec{l}$ and $3\overleftarrow{l}$, $3\vec{l}^{\star}$ and $3\overleftarrow{l}^{\star}$, $3\vec{m}$ and $3\overleftarrow{m}$, $3\vec{m}^{\star}$ and $3\overleftarrow{m}^{\star}$.

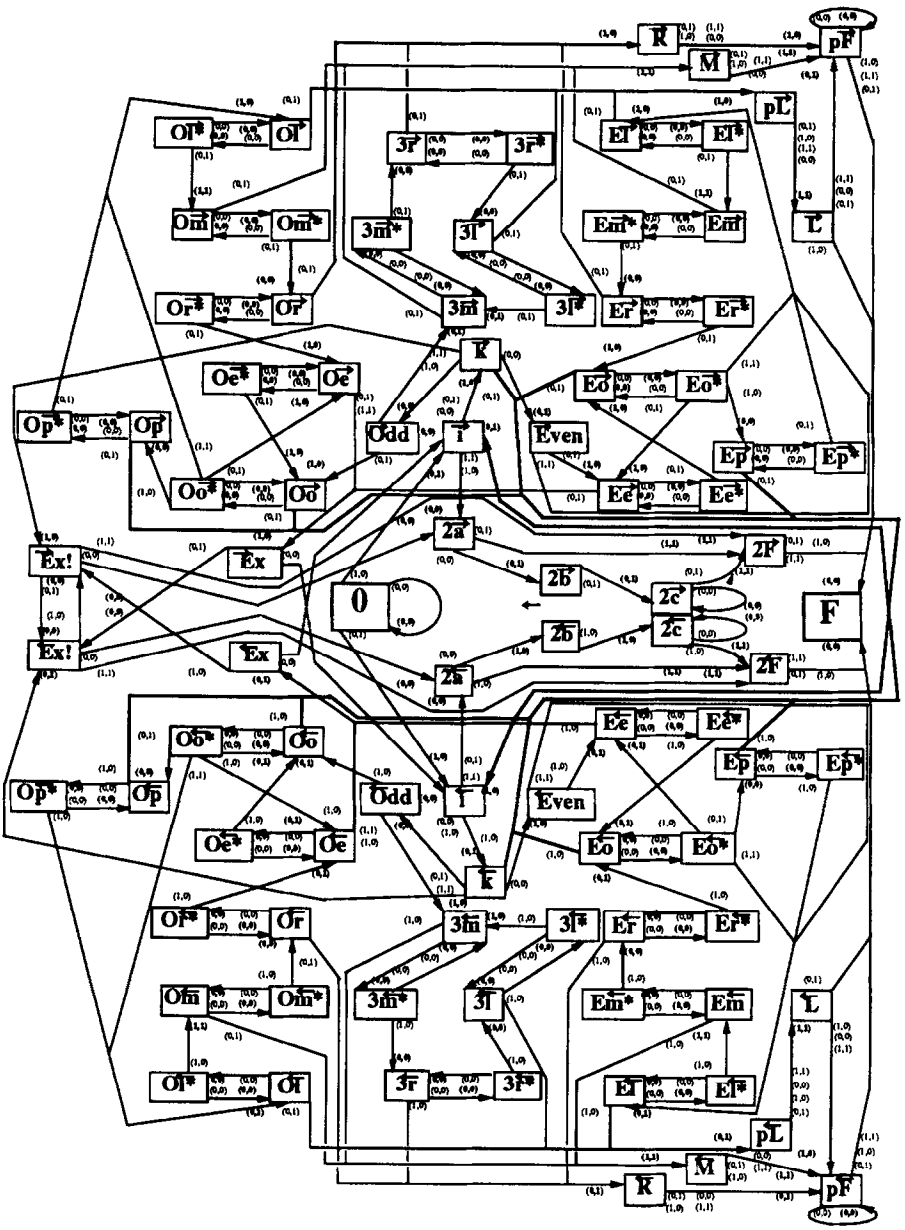


Fig. 12. States of the generic automaton.

3. When a cell is an "even" state ($X\vec{e}$ or $X\overleftarrow{e}$ where X is O or E) the side from which digit 1 may arise depends on the direction of the initial wave. Thus, we identify $E\vec{e}$ and $E\overleftarrow{e}$, $E\vec{e}^{\star}$ and $E\overleftarrow{e}^{\star}$, $O\vec{e}$ and $O\overleftarrow{e}$, $O\vec{e}^{\star}$ and $O\overleftarrow{e}^{\star}$.

4. The same remark holds for states $X\vec{p}$, leading us to identify $E\vec{p}$ and $E\overleftarrow{p}$, $E\vec{p}^{\star}$ and $E\overleftarrow{p}^{\star}$, $O\vec{p}$ and $O\overleftarrow{p}$, $O\vec{p}^{\star}$ and $O\overleftarrow{p}^{\star}$.

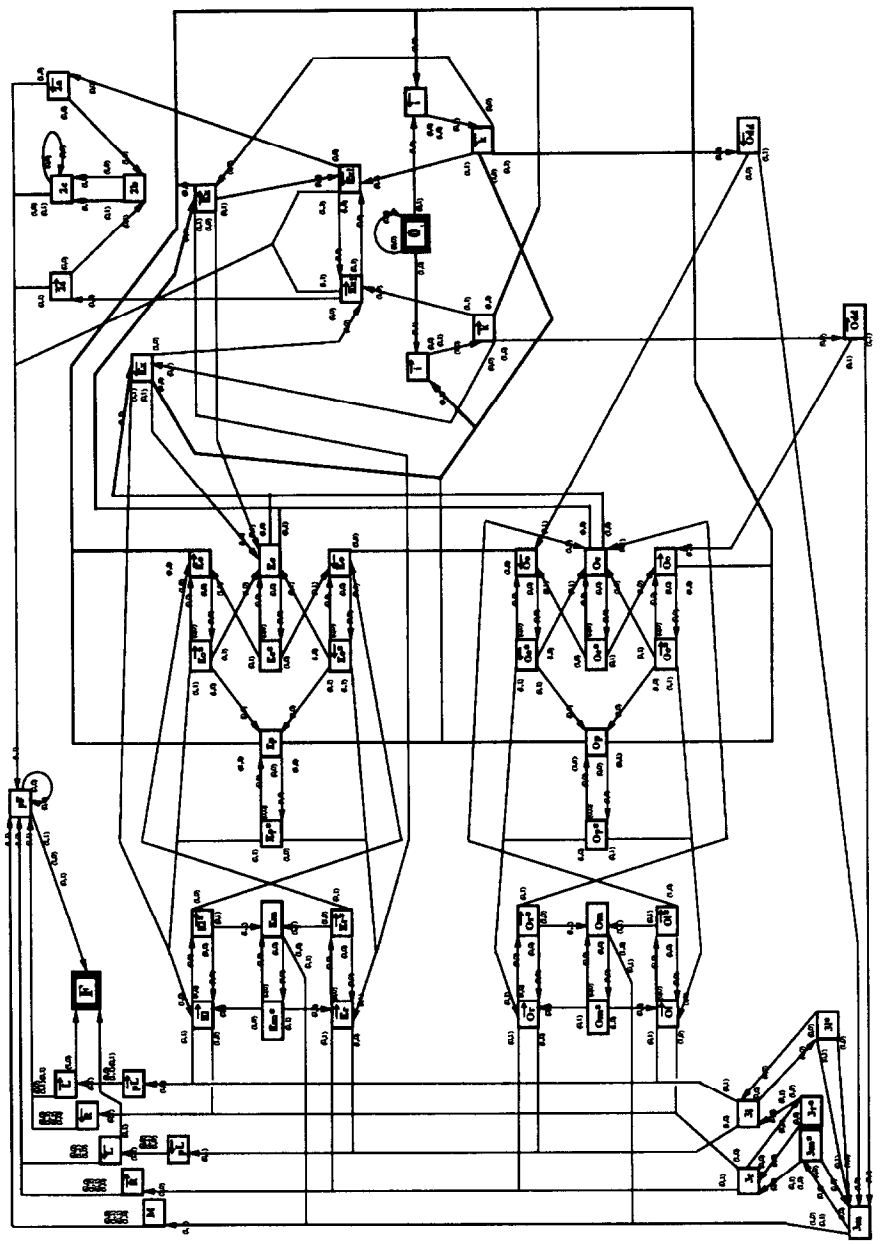


Fig. 13. States of the generic automaton with 58 states.

5. The same remark also holds for states $X\vec{m}$, leading us to identify $E\vec{m}$ and $E\overleftarrow{m}$, $E\vec{m}^{\star}$ and $E\overleftarrow{m}^{\star}$, $O\vec{m}$ and $O\overleftarrow{m}$, $O\vec{m}^{\star}$ and $O\overleftarrow{m}^{\star}$.

6. But the previous remark does not hold for states $E\vec{l}$ and $E\overleftarrow{l}$. Nevertheless, we may identify $E\vec{l}$ and $E\overleftarrow{r}$ and, similarly, $E\vec{r}$ and $E\overleftarrow{l}$, $O\vec{l}$ and $O\overleftarrow{r}$, $O\vec{r}$ and $O\overleftarrow{l}$, $E\vec{r}^{\star}$ and $E\overleftarrow{r}^{\star}$, $E\vec{r}^{\star}$ and $E\overleftarrow{l}^{\star}$, $O\vec{l}^{\star}$ and $O\overleftarrow{r}^{\star}$, $O\vec{r}^{\star}$ and $O\overleftarrow{l}^{\star}$.

7. Finally, we observe that we may identify $\vec{M}$ and $\overleftarrow{M}$, $p\vec{F}$ and $p\overleftarrow{F}$, $\vec{E}$ven and $\overleftarrow{E}$, $\overleftarrow{E}$ven and $\vec{E}$x.

Clearly, it is easy to reduce this number of states of one or two units modifying some chooses. We do not know the minimal number of states needed for a synchronizing automaton with a two-ways channel. Our solution achieves synchronization in time $2n-2$ which is the minimal time. Thus, we may state the theorem:

Theorem 2. *There exists a minimal time solution to the FSSP with only one digit of information exchanged in both directions and with* 58 *states.*

4. A minimal time solution with a minimal number of one-way channels

In this section, we study one-way channel solutions. First, in Section 4.1, we observe that we do not need to construct directly one-way channel solutions but we can adapt the two-ways solution of Section 3 excluding all crosses. Then, we give in Section 4.2 some indications on what are optimal time solutions in the context of the one-way channel constraint. Finally, we adapt the solution of Section 3, obtaining a solution without crosses with 230 states and only one one-way channel. This solution has been tested by computer for segment from 2 to 1000. It induces a one-way solution with, at most, 920 states.

4.1. Excluding crosses

We do not construct a one-way solution in the sense of Definition 5. We denote in the space-diagram of a two-ways channel automaton (such as the one depicted in Fig. 2, by $\gg k,t \gg$ ($\ll k,t \ll$) the digit sent by the cell k to its right (left) neighbour at time t. Now we define a two-way solution to the FSSP *excluding crosses.*

Definition 6. A two-way solution in time $t(n)$ to the FSSP excludes crosses if in any space–time diagram of the behaviour of the solution on any initial configuration $C[n]$ of Definition 3 the situation

$$\gg k,t \gg = 1 \quad \text{and} \quad \ll k+1,t \ll = 1$$

never occurs for all k in $\{1,\ldots,n\}$ and t in $\{0,\ldots,t(n)\}$.

In other words, no picture (cross) of the form

appears in any significant space–time diagram of the two-way solution.

Proposition 1. *Let* $\mathscr{A}$ $((Q,\{!,0,1\},\delta))$ *be a two-way solution in time* $t(n)$ *to the FSSP excluding crosses with* $|Q|$ *states, then there exists a one-way solution* $\mathscr{A}^\star$ $(Q^\star, \{!,0,1\},\delta^\star)$ *in time* $t(n)$ *to the FSSP where* $|Q^\star| = 4 \times |Q|$.

Proof. The formal proof is tedious but the idea is very simple. The set of states of $\mathscr{A}^\star$ is given by: $Q^\star = Q \times \{0,1\}^2$. If in the space–time diagram of $\mathscr{A}^\star$, a cell k is, at time t, in state (q,ε,η), it understands this fact as: *In the space–time diagram of* $\mathscr{A}$, *the cell* k *is, at time* t, *in state* q *and has emitted* ε *to its left neighbour and* η *to its right neighbour.* Thus, the next time, receiving from its right the digit 1, it understands it as sent by its left neighbour (case $\eta = 0$) or not (case $\eta = 1$).

Now we give $\delta^\star$.

Case $\varepsilon = 1$ *and* $\eta = 1$

$$\delta^\star(j_{r,1},(q,1,1),j_{l,2}) = \delta(0,q,0).$$

Case $\varepsilon = 0$ *and* $\eta = 1$

$$\delta^\star(j_{r,1},(q,0,1),j_{l,2}) = (j_{r,1},q,0).$$

Case $\varepsilon = 1$ *and* $\eta = 0$

$$\delta^\star(j_{r,1},(q,1,0),j_{l,2}) = \delta(0,q,j_{l,2}).$$

Case $\varepsilon = 0$ *and* $\eta = 0$

$$\delta^\star(j_{r,1},(q,0,0),j_{l,2}) = \delta(j_{r,1},q,j_{l,2}).$$

The proof of the simulation of $\mathscr{A}$ by $\mathscr{A}^\star$ is long and tedious. It is easy to show by induction on the time that the evolutions are what we have in mind. □

Thus, in the following, we shall only consider two-way solutions to the FSSP excluding crosses.

4.2. *One-way optimal time solutions*

If in the case of two-way solutions to the FSSP, the minimal time remains $t(n) = 2n-2$ as in the standard case, this point is no more true in the one-way channel case. This results from the following proposition:

Proposition 2. *For any segment of length* n $(n \geqslant 2)$, *the evolution of a one-way minimal solution to the FSSP is such that*:

1. $\gg 1,0 \gg = 1$.
2. *If the minimal time is asymptotically* $2n$, *then, for* k *in* $\{1,\ldots,n-1\}$, $\gg k, k-1 \gg = 1$.
3. *If the minimal time is ultimately* $2n-2$, *then, for some integer* n_0, $t(n_0) \geqslant 2n_0 - 1$.

Proof. (1) There exists some time τ $(\tau \geqslant 0)$ for which $\gg 1,\tau \gg = 1$: else, all cells (except the general) will stay in the quiescent state L and no synchronization will occur.

Let $\mathscr{A}$ be a one-way minimal solution to the FSSP in time $t(n)$. If the value of $\tau_{\mathscr{A}}$ corresponding to $\mathscr{A}$ is greater than 0, then we define one automaton $\mathscr{B}$ with the same states and transition function than $\mathscr{A}$ except that it synchronizes initial configuration in which the general is in the state of $\mathscr{A}$ at time $\tau_{\mathscr{A}}$. Automaton $\mathscr{B}$ is a one-way minimal solution to the FSSP in time $t(n) - \tau_{\mathscr{A}}$. Contradiction.

2. Let $\mathscr{A}$ be a one-way minimal solution to the FSSP in asymptotical time $2n$. We consider a segment of length 3. There exists some integer τ such that $\gg 2,1+r \ggg= 1$ (else the third automaton will stay in the quiescent state). If we suppose that $\tau > 0$, then for a segment of length n, we have, for k in $\{2,\ldots,n-1\}$:

- $\forall \alpha \in \{0,\ldots,(k-1)\tau\}$, $\gg k,k-1+\alpha \ggg= 0$,
- and $\gg k,k+(k-1)\tau \ggg= 1$.

Thus, the time of synchronization is, at best, $2n+(n-1)\tau$ which is asymptotically greater than $2n$. Thus, by contradiction, $\tau = 0$.

3. Let $\mathscr{A}$ be a one-way minimal solution of the FSSP in time $2n-2$. Let us recall that there does not exist solution to the FSSP which synchronizes some segment of length n_1 in a time less than $2n_1-2$. For contradiction, we suppose that $t(2) = 2$. For a segment of length 2, 4 cases are possible (see Fig. 14). In the four cases, by the point 1) and the definition of the initial configuration, we have $\gg 1,0 \ggg= 1$, $\ll 1,0 \lll= 0$. We study these cases:

Case 1: $\gg 1,1 \ggg= 1$, $\ll 2,1 \lll= 0$.

If the segment has length 3, the first cell receives digit 1 from its right and, as for a segment of length 2, it enters the Fire at time 2; and the synchronization of a segment of length 3 would be achieved before time 4 which is impossible.

Case 2: $\gg 1,1 \ggg= 1$, $\ll 2,1 \lll= 1$.

Similar to case 1.

Case 3: $\gg 1,1 \ggg= 0$, $\ll 2,1 \lll= 1$.

If the segment has length 3, then $\gg 2,1 \ggg= 1$ (by point 2), $\ll 3,2 \lll= 1$ (since $\ll 2,1 \lll= 1$ in our hypothesis). Whatever the value of $\gg 2,2 \ggg$ is, at time 3, the third cell receives digit 1 from its left. But synchronization of a segment of length 2 in case 3, implies that it enters the Fire at time 3; thus before time 4 which is impossible.

Case 4: $\gg 1,1 \ggg= 0$, $\ll 2,1 \lll= 0$.

First we consider the synchronization of a segment of length 3. We have:

- $\gg 2,1 \ggg= 1$ (point 2);
- $\ll 2,1 \lll= 1$ (else, as we are in case 4, the first cell would enter Fire at time 3 which is impossible);
- $\ll 3,2 \lll= 0$ (in case 4, we have $\ll 2,1 \lll= 0$);
- $\ll 2,2 \lll= 1$ (else, as we are in case 4, the third cell would enter Fire at time 3 which is impossible);
- Between the first and the second cell is digit 1 at time 2 (it is to say $\gg 1,2 \ggg= 1$ or $\ll 2,2 \lll= 1$) because, else, as we are in case 4, the first cell would enter Fire at time 3 which is impossible.

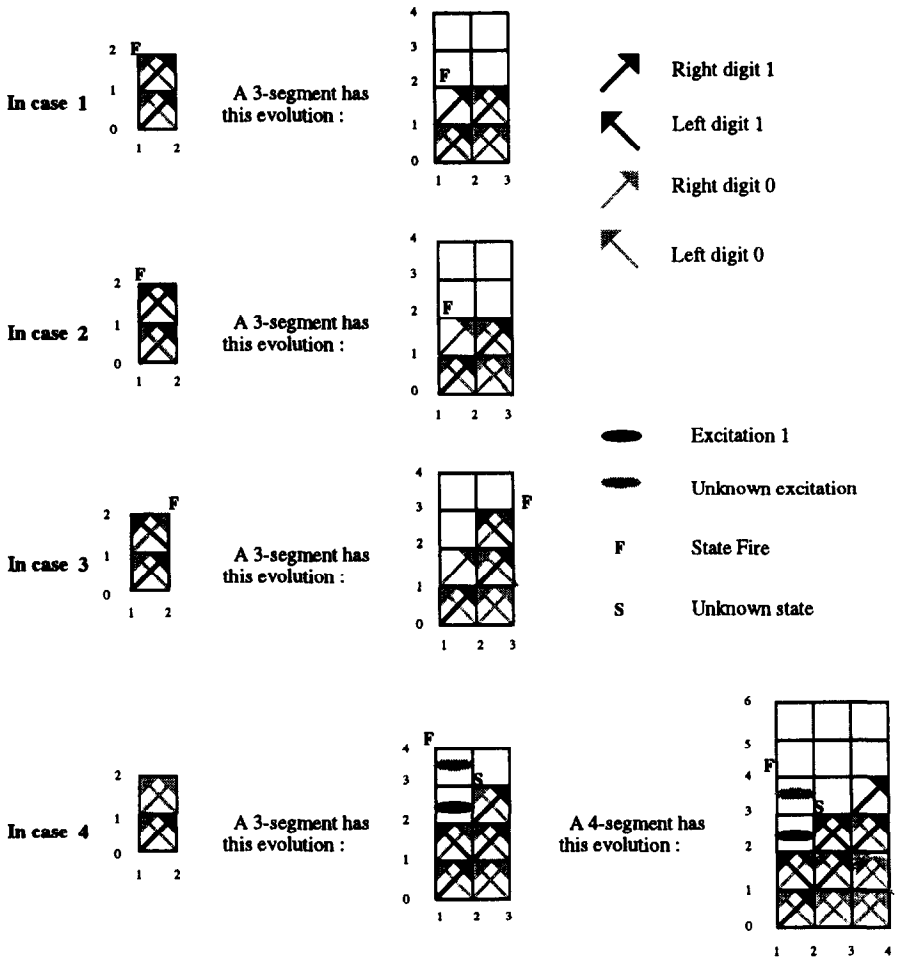


Fig. 14. Cases appearing in the proof of Proposition 2.

- We observe that if the synchronization is achieved at time 4 then $\gg 1,3 \gg = 0$ or $\ll 2,3 \ll = 0$.

Now, we consider a segment of length 4, digits emitted for the first and the second cell are the same than in case of length 3 from time 0 up to 2. In addition, the third cell acting as the second one emits digit 1 to its left at time 1, and to its right at time 2. The second cell is, at time 2 in both cases (lengths 3 and 4) in the same state and emits the same digit to the first cell. Thus, in both cases, the first cell enters the Fire at time 3 which is impossible. □

From Proposition 2, we deduce that the minimal time possible is, at best, $2n - 2$ almost everywhere and we know that an exception is $n = 2$ or $n = 4$. We do not

search for defining with more accuracy what is the minimal time with the one-way constraint. We only set the following definition.

Definition 7. A one-way solution to the FSSP in time $t(n)$ is in (n_0)-minimal time if $t(n) = 2n - 2$ for all values of n greater than n_0.

In the remainder of this Section 4, we prove the following theorem which shows that (n_0)-minimal time one-way solutions exist.

Theorem 3. *There exists a* (6)*-minimal time one-way solution to the FSSP. Its values of* $t(n)$ *are given by*: $t(2) = 3$, $t(3) = 6$, $t(4) = 8$, $t(5) = 8$, $t(6) = 12$ *and for* $n \geqslant 7$, $t(n) = 2n - 2$.

4.3. Modifications to the previous strategy

Fig. 15 shows the crosses appearing in the evolution of the solution described in Section 3. All these crosses appear during the evolution setting up the breaks of the segment (see Fig. 2 for the delay which does not use digit 1). We list the different locations of the crosses:

(i) Crosses between digits 1 setting up ($\star$) potential eventual-end sites and the reflected initial wave (for example between cells 4 and 5 at time 20 if the segment has length 13 in Fig. 15).

(ii) Crosses between digits 1 setting up ($\star$) eventual left-end sites and the reflected initial wave (for example between cells 4 and 5 at time 8 if the segment has length 7 in Fig. 15). This point concerns only the first break signal.

(iii) Crosses between digits 1 indicating the parity and the ones corresponding to the auxiliary signals setting up the first break signal (for example between cells 4 and 5 at time 5 in Fig. 15).

(iv) Crosses related at the initiation of the whole process: between cells 1 and 2 at time 1 (for a length of 2), between cells 2 and 3 at time 4 (for a length of 4).

Fig. 16 depicts the new information flow obtained when we achieve the following modifications.

Point (i) is easy to solve. We observe that – in the exchange of digits described in Fig. 15 – the first ($\star$) potential eventual-end site (corresponding to the jth break) can be set up one unit of time later avoiding the litigious cross. This does not introduce any new problem: any cell knows that the synchronization was initiated at its left and that the 1st break signal has been set up; thus, in this case, it will always understand a digit 1 coming from its left as a signal putting it in the ($\star$) potential eventual-end sites. For example, in Fig. 16, a digit 1 is set by cell 4 to its right neighbour not at time 20 but at time 21. This is done introducing new states: to send the digit and to receive it.

Point (ii) is more difficult to solve. First, the previous modification is also introduced for the 1st break signal. By this way, the crosses of point (ii) disappear but new crosses appear one unit of time later between digits 1 setting up ($\star$) eventual left-end sites and,

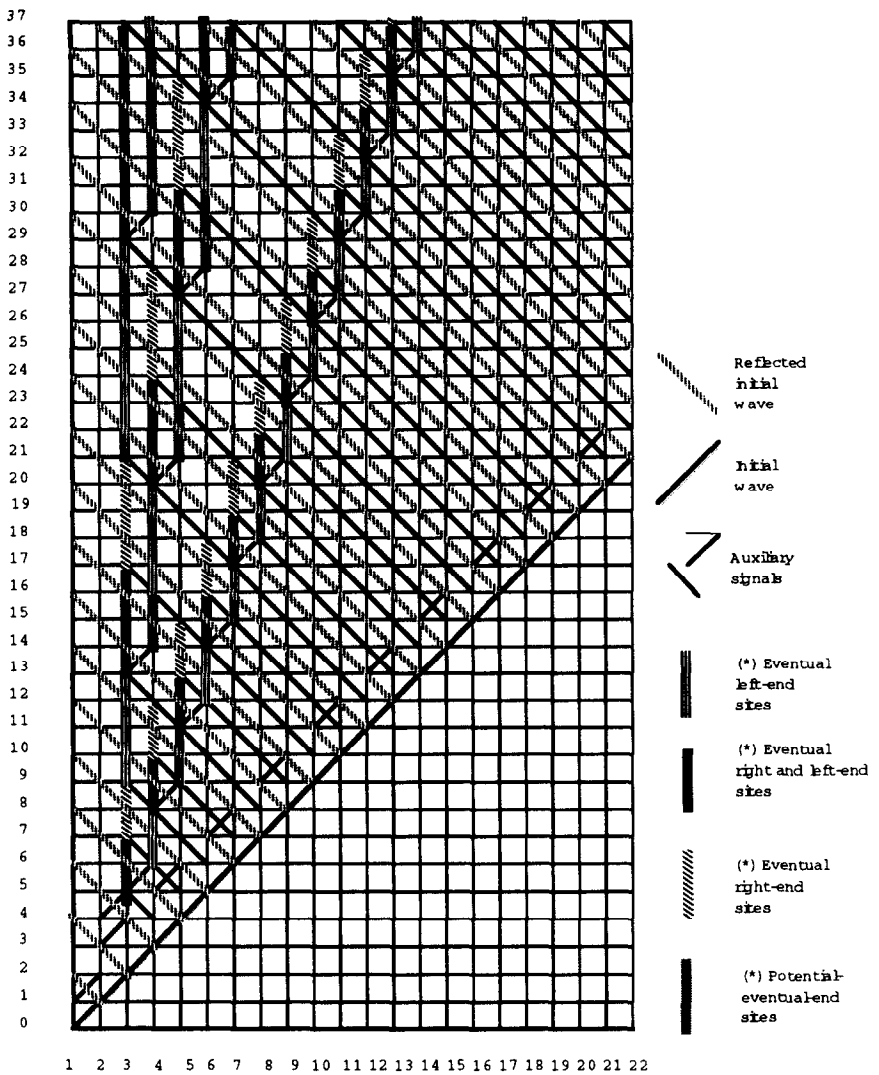


Fig. 15. Crosses appearing in the information flow setting up breaks of the segment in the two-ways solutions of Section 3.

no more the reflected initial wave, but the auxiliary signals setting up the 1st break signal (for example between cells 4 and 5 at time 9). Now the solution is obvious: we suppress one out of two of the auxiliary signals setting up the 1st break signal. This is possible observing that the 1st break signal remains only one time in any break state at level 1. This 1st break signal has now the following behaviour (illustrated in Fig. 16):

- Receiving digit 1 from its left neighbour, one cell enters the state $\vec{E1}$ or $\vec{O1}$ (eventual-left state). If simultaneously it receives digit 1 from its right neighbour (the reflected initial wave) it enters state $\vec{L}$.

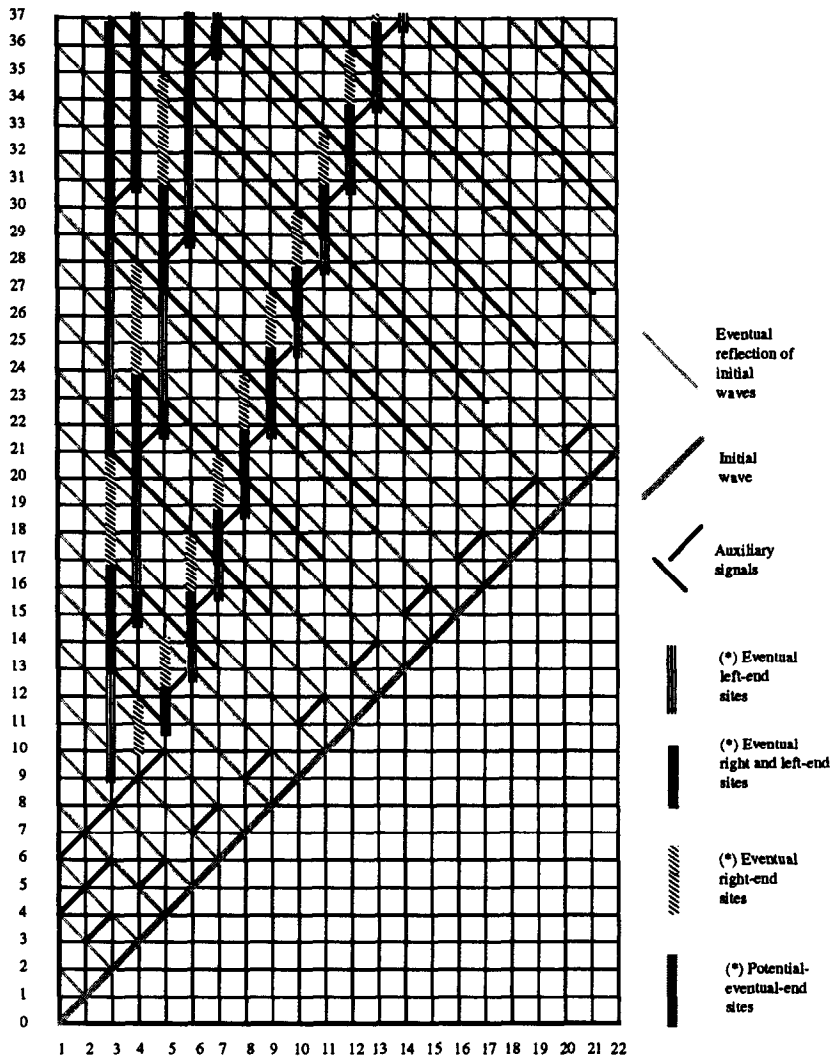
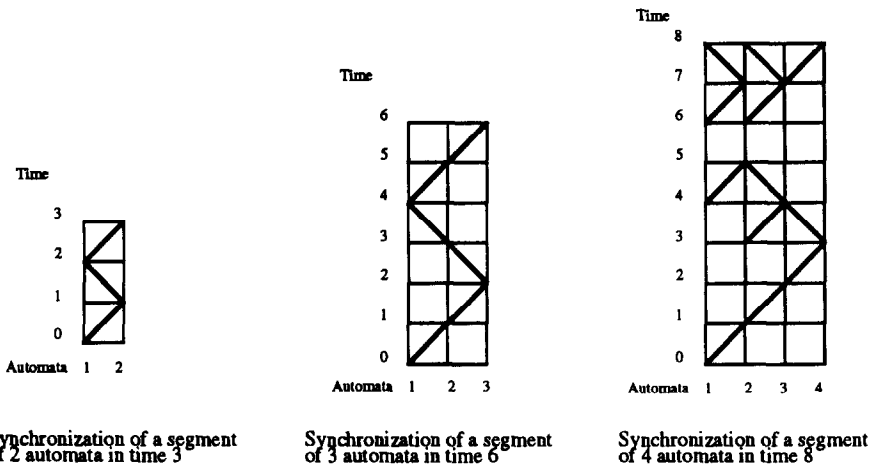


Fig. 16. The information flow setting up breaks of the segment in the one-way solution.

- One unit of time later, it receives digit 1 from its right neighbour and enters state $E\vec{m}^{\star}$ or $O\vec{m}^{\star}$. The next time it enters the state $E\vec{m}$ or $O\vec{m}$.
- The next time, it enters (without receiving digit 1 from its right neighbour) the state $E\vec{r}^{\star}$ or $O\vec{r}^{\star}$.
- Then, its evoluation is as in Section 3 with the previous modification.

Point (iii) is easy to solve: it is sufficient to send auxiliary signals some units of time later. Observing Fig. 15, we see that only one unit is sufficient. In fact, the number of units added depends on how we solve the point (iv). As indicated in Fig. 16, we have chosen to add 6 units of time. This is achieved by introducing 6 new states (for odd and even automata) before to enter the states $O\vec{o}^{\star}$ or $E\vec{o}^{\star}$.

Point (iv) is solved following Fig. 17. This figure depicts the synchronization of short segments from length 2 to 6. Many other solutions exist but these ones are simpler. In the general case (segments of length greater than 6), we must initiate the whole process by the exchange of digits depicted on the Fig. 16. The six first automata now know their number (see Fig. 16) and the initiation is completed at time 11. Then, after time 11, only the three first ones remember their number according to the process of Section 3.



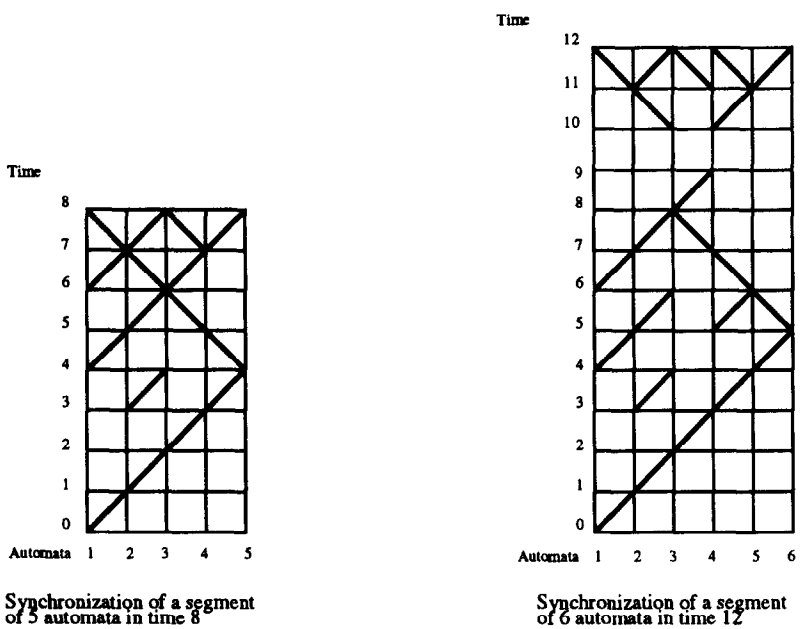


Fig. 17. Synchronization of short segments in the one-way solution.

Such an automaton may be constructed with 230 states without any minimization, giving us a one-way solution to the FSSP with 920 states.

5. Solution with few states

5.1. Results

The definition of the FSSP (see Definition 4) involves three states, $0, \vec{1}$ and F (the "outside" ! is not considered as a state in this count). In fact, one can think about a definition involving only two states 0 and F. State $\vec{1}$ is used to initiate the synchronization, but the synchronization may be initiated when the general sends some special information μ to its right neighbour. Thus, the minimal number of states needed to get the synchronization may be 2. In this section, we show that there does not exist any solution to the FSSP with 2 states but that there exists one with 3 states.

Theorem 4. *There does not exist any solution to the FSSP with* 2 *states.*

Proof. For contradiction, we suppose that such a solution (in time $t(n)$) exists. The initial configuration $C[n]$ is now:

$$^{\infty}(!,!,!)(\mu,O,\mu),\underbrace{(0,0,0),\ldots,(0,0,0)}_{n-1 \text{ times}}(!,!,!)^{\infty}.$$

All active cells (from times 0 to $t(n)$) are in state 0. A simple induction shows that, due to the quiescent character of state 0 and information 0, the evolution is such that:

At even times, cells with an even number receive and emit in both directions information 0; and at odd times, cells with an odd number receive and emit in both directions information 0.

Without loss of generality, suppose that $t(n)$ is even, at time $t(n)-1$ an odd cell receives in state 0 information 0 from its two neighbours and thus enters, at time $t(n)$, state 0 (point (iii) of the definition 4) and not the Firing state. Contradiction. □

In the next section, we prove Theorem 5.

Theorem 5. *There exists a minimal time* $(2n-2$ *units*$)$ *solution to the FSSP with* 3 *states.*

5.2. An automaton with 3 states and large information flow

The synchronization is initiated by information and not by state. Thus, the initial configuration is

$$^{\infty}(!,!,!)(\mu,0,\mu),\underbrace{(0,0,0),\ldots,(0,0,0)}_{n-1 \text{ times}}(!,!,!)^{\infty}.$$

We shall only give some indications on such a solution. We consider the Balzer's solution, depicted in Fig. 1 and, briefly, described in Section 3.1.

Looking to the proof of Theorem 4, we observe that, without using new states, the initiation of the synchronization can modify the information flow emitted by all the points of the space–time diagram of the form (x,t) with $x+t$ odd. We call "first grid", denoted F_1, the set of sites $\{(x,t) \mid x+t \text{ odd and } x+t \geqslant 3\}$. The first grid corresponds to the area influenced by the digit μ. The "second grid" is F_2, defined by $\{(x,t) \mid x+t \text{ even and } x+t \geqslant 4\}$. We observe that all sites in the area of the synchronization ($\{(x,t) \mid t \geqslant x-1\}$) are:

$$F_1 \cup F_2 \cup (1,0) \cup (1,1).$$

We shall use a new state *ev* to mark sites of F_1 and 0 to mark F_2; the two other sites will have a special treatment.

The information is viewed as a product of elementary informations:

$$J = J_{\text{Lay}} \times J_{\text{Dir}} \times J_{\text{Sta}} \times J_{\text{StaRef}} \times J_{\text{Trans}} \times J_{\text{Ref}},$$

where:

(i) $J_{\text{Lay}} = \{0, \mu, \alpha, \varepsilon, o, f\}$,
(ii) $J_{\text{Dir}} = \{\ell, r\}$,
(iii) $J_{\text{Sta}} = \{0, g, a, b, l\}$,
(iv) $J_{\text{StaRef}} = \{0, g, a, b, l\}$,
(v) $J_{\text{Trans}} = \{0, g, a, b, l\}$,
(vi) $J_{\text{Ref}} = \{0, \rho, \mu, \mu^*\}$.

The quiescent information is $(0, \ell, 0, 0, 0, 0)$ and at time 0, the left-end automaton sends $(\mu, \ell, 0, 0, 0, 0)$.

Now, we describe the behaviour of the automaton.

1. The first component, J_{Lay}, is used to set up the two grids F_1 and F_2. Its behaviour is shown in Fig. 18.

At time 0, the first cell sends information μ.

At time 1, the first cell, receiving in state 0 the quiescent information, enters state 0 sending the quiescent information. But, the second, receiving in state 0 information μ from its left neighbour, can enter state *ev* sending α to its left neighbour and ε to its right one.

At time 2, the first cell, in state 0, receives α from its right neighbour, and enters *ev* sending ε. The second, the state *ev* receives 0 from its two neighbours and enters 0 sending o in both directions. The third, in state 0, receives ε from its left and enters *ev* sending ε in both directions.

For times greater than 2, every cell in state 0 (*ev*) receives ε (o) and enters *ev* (0) sending ε (o) in both directions.

By this way, since time 2, state *ev* (0) marks F_1 (F_2).

We observe that a cell receiving, in state $0, \varepsilon$ (μ) from its left and ! from its right knows that it is the right-end cell and that its number is greater than 2 (is equals to 2).

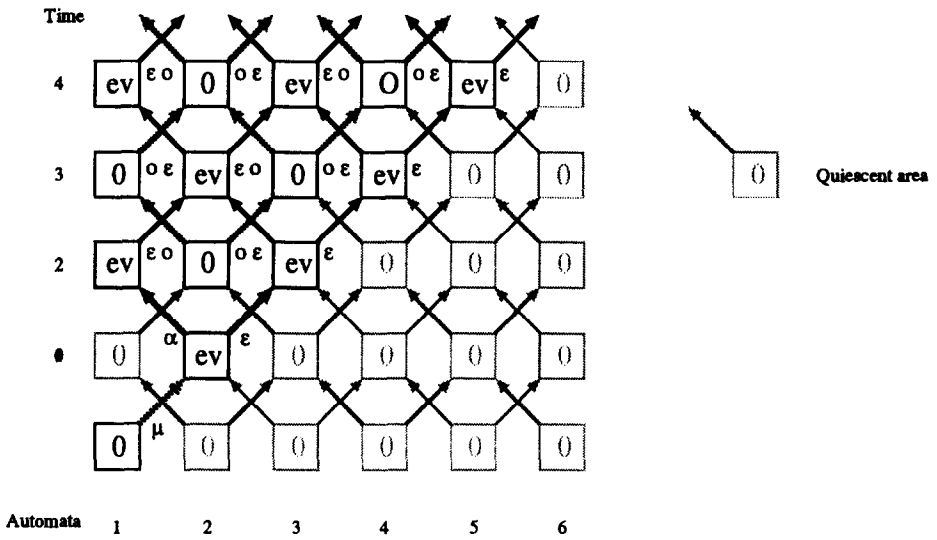


Fig. 18. Sorting the two grids with 2 states.

Segment of length 2 has a special synchronization process. When the second cell, in state 0, receives μ and !, it sends to its left neighbour f and enters the Fire. The first cell, receiving f, enters the Fire.

2. The second component, J_{Dir}, is used to indicate the direction of the synchronization. If the general is the left-end cell, it is set to ℓ, else to r. We shall see later how to invert it.

3. The third, fourth and fifth components, $J_{\mathrm{Sta}}, J_{\mathrm{StaRef}}$ and J_{Trans}, are used to simulate the Balzer's solution below the path drawn by the reflected initial wave.

Looking at Fig. 1, we observe that, below the reflected initial wave, if some site (x,t) of F_1 is in some state, then the site $(x,t+1)$ of F_2 is in the same state. This fact is proved in [1] or [5]. Thus to know the state of (x,t) of F_1 in the Balzer's solution, cell x only needs to know the state of $x-1$ at time $t-1$ (which can be transmitted through F_1), its own state at time $t-2$, and the state of $x+1$ at time $t-1$ (which can be transmitted through F_1). At time 1, the second automaton knows its number and, at time 3, the first and the third ones also know their number (see point 1). At these times they set their third component to g (first automaton), a (second) and b (third), corresponding to the states of the Balzer's solution. Then, when a cell in state ev receives g or a or b in its third component, its reflects this value to the sender in its fourth component. It also sends, in its fifth component, the current state of the Balzer's solution. This trick allows us to carry on with the simulation on the grid F_1: at time t, cell x receives state of $x-1$ at time $t-1$ from the left in the fifth component, its own state at time $t-2$ from the both directions in the fourth component, state of $x+1$ at time $t-1$ from its right in the fifth component.

The simulation on the grid F_2 is similar. It is sufficient to observe that:

- First, the second and third cells know their number at time 3, 2 and 3; and, thus, they can set their component in g, a and b.
- A cell, in state 0 receiving o from its left and ! from its right knows that it is the right-end cell and that the initial wave has reached it one unit of time before.

4. The sixth component, J_{Ref}, is used to set up the reflected initial wave and the delay. As previously observed, a cell knows that it is the right-end cell and that the reflected initial wave reaches it or has reached it for one unit of time. At this time, it sends in its sixth component a value different from 0, value $\mu^\star$ or value μ, according to the value of the received second component (indicating the direction of the synchronizing process), and it changes the value sent in its second component.

The delay is set up by a flip-flop process. If the length is even (indicated in Balzer's solution by state a), then the right-end cell sets up its sixth component to μ and the next time to ρ. Else, the order is $\mu^\star$ and, then, ρ. When an automaton receives μ (or $\mu^\star$) in its sixth component, it considers than its current state is ev and that its current layer is F_1. In this way, if the length is even, F_2 becomes F_1 and one unit of time is spent. Fig. 19 illustrates the flip-flop process setting the delay.

This process is simple when the length is odd. Automata, in state ev receiving as their previous state a and μ in their sixth component act as if their state were 0 and they receive the quiescent state. The only difficulty is to set up the break. The break signal is indicated by the fact that an automaton receives l as its previous case. In this case, in state ev receiving μ, it simulates the state of the general (g) and emits μ or $\mu^\star$ to its two neighbours according to the values of its fifth component. But, in state 0 receiving $\mu^\star$, it also simulates g and emits μ or $\mu^\star$.

When the length is even, in state ev receiving in its fourth component b (its previous state), it transmits $\mu^\star$. But, in state ev receiving l in its fourth component, it simulates g and emits μ or $\mu^\star$ to its left (and not right) neighbour, according to the value received in its fifth component from the left. In state 0 receiving b in its fourth component and μ in its sixth, it acts as if it was in ev, emitting ε in its first component and simulating Balzer's solution. But, if it receives μ (sixth component) and l (fourth one), it simulates g (setting up the right-end automaton of the left subline). If it receives μ (sixth component) and l (fifth one from the left), it simulates g (setting up the second time of the right-end automaton of the right subline).

The obtained automaton has three states ($0, ev$ and F) – in fact, state F is useless and may be replaced by an information – and about 4500 elementary pieces of information corresponding to 12 channels. We have not searched to minimize the number of pieces of information.

6. Conclusion

We observe that if the notion of optimality in time is well defined for a solution to the FSSP (in the one-dimensional case here studied but also in case of graphs [3, 4]),

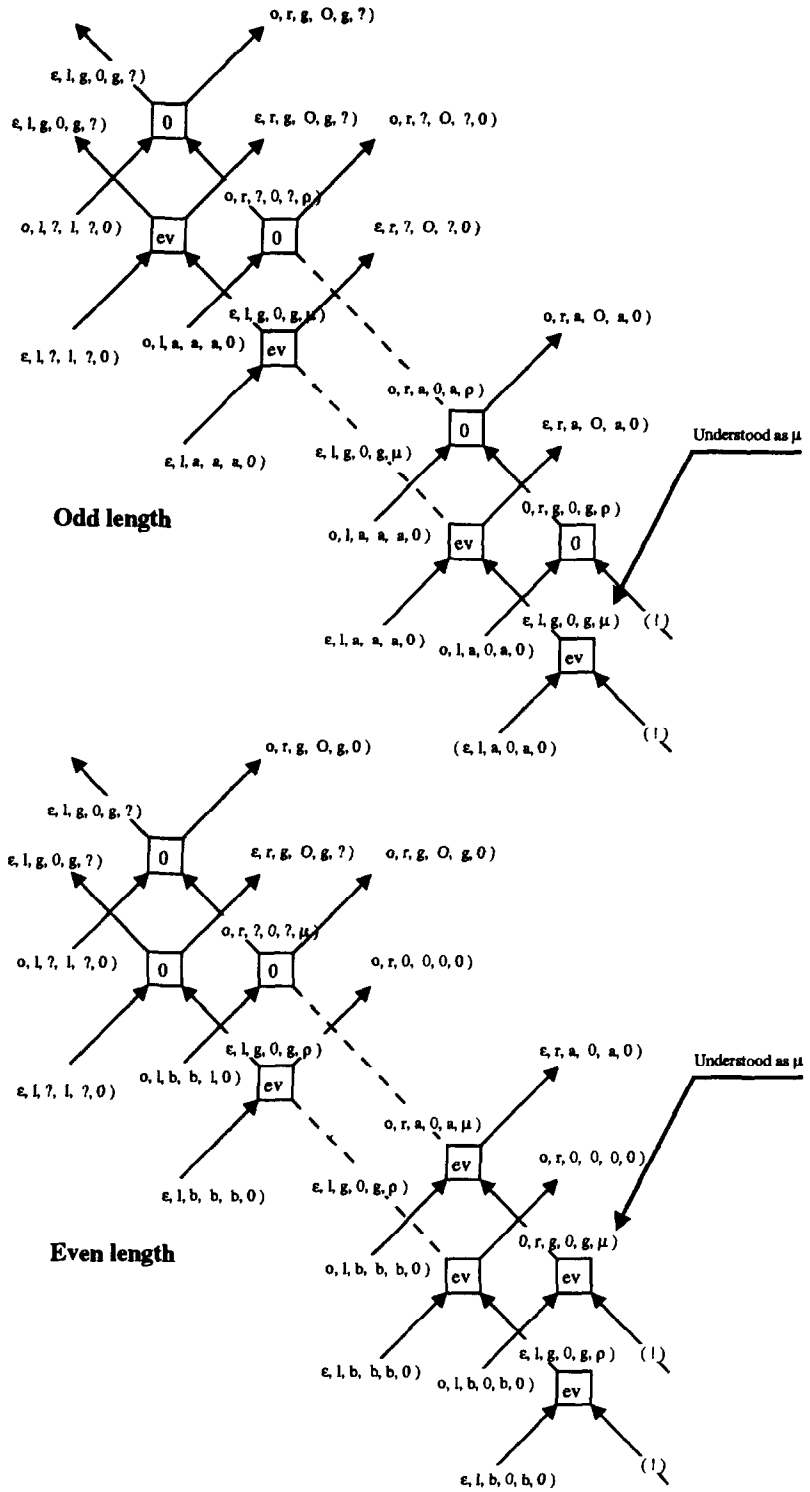


Fig. 19. Setting up the delay with two states.

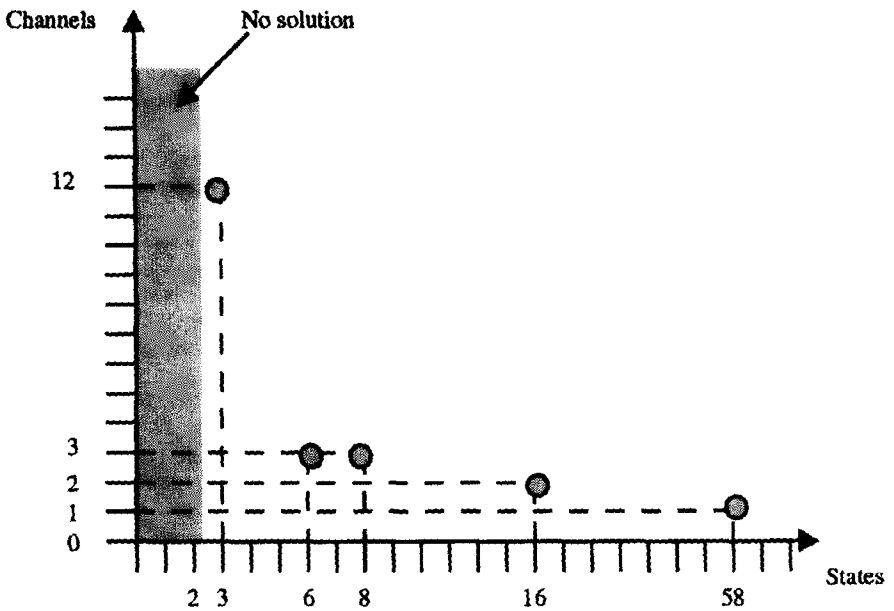


Fig. 20. Known results.

the notion of optimality in size is more complex. This is due to the following two remarks:

(a) It is very difficult to study the set of solutions since it is not recursively enumerable (Section 2).

(b) What is a good optimality? How to mix states and information flow?

The known results are summarized in Fig. 20. The solution with 16 states and 2 channels is easy to construct.

We also observe that all solutions here described can be extended to the case where the general is anywhere in the segment.

References

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

[2] G.T. Herman, Synchronization of growing automata, *Inform. and Control* **25** (1974) 103–122.

[3] K. Kobayashi, The firing squad synchronization problem for two dimensional arrays, *Inform. and Control* **34** (1977) 177–194.

[4] K. Kobayashi, The firing squad synchronization problem for a class of polyautomata networks, *J. Comput. System Sci.* **17** (1978) 300–318.

[5] J. Mazoyer, A 6-state minimal time solution to the firing squad synchronization problem, *Theoret. Comput. Sci.* **50** (1987) 183–238.

[6] J. Mazoyer, A minimal time solution to the Firing Squad Synchronization Problem with only one bit of information exchanged, *Research Report LIP-IMAG*, Report 89-03, 1989.

[7] J. Mazoyer, Solutions au problème de la synchronisation d'une ligne de fusiliers: étude de leur structure, Rapport d'habilitation, 200 pp, École Normale Supérieure de Lyon, 1989.

[8] M. Minsky, *Finite and Infinite Machines* (Prentice-Hall, Englewood Cliffs, NJ, 1967) 28–29 and 282–283.
[9] E.F. Moore, Sequential Machines, selected papers (Addison-Wesley Reading, MA, 1964) 213–214.
[10] G.G. Landon and F.R. Moore, A generalized firing squad problem, *Inform. and Control* **12** (1968) 212–220.
[11] P. Rosenstiehm, J.R. Fiksel and A. Holliger, Intellugent graphs: networks of finite automata capable of solving graph problems, in: R.C. Read ed., *Graph Theory and computing* (Academic Press, New York, 1972) 219–265.
[12] A.R. Smith, Real time language recognition by one-dimensional cellular automata, *J.A.C.M* **6** (1972) 233–235.
[13] H. Szwerinski, Time optimal solution of the Firing Squad Synchronization Problem for n-dimensional rectangles with the general at an arbitrary position, *Theoret. Comput. Sci.* **19** (1982) 305–320.
[14] V.I. Varshavsky, V.B. Marakhovsky and V.A. Peschansky, Synchronization of interacting automata, *Math. Systems Theory* **14** (1969) 212–230.
[15] A. Waksman, An optimum solution to the Firing Squad Synchronization Problem, *Inform. and Control* **9** (1966) 66–78.
[16] J.B. Yunes, Seven state solutions to the Firing Squad Synchronization Problem, **127** (1994) 313–332.