

# Cellular Automata: An implementation of a fault tolerant scheme of the FSSP

Jean-Baptiste Yunès  
LIAFA  
Université Paris 7 – Denis Diderot  
France

## FSSP

On a finite line of cells connected to their nearest neighbors, we define the Firing Squad Synchronization Problem as follows:

- given a **starting configuration** of arbitrary length: a general on the left cell and quiescent soldiers everywhere.
- find a **local function** which gives to any starting configuration a global behavior with the following properties:
  - evolves to a **synchronizing configuration** (everybody fires),
  - under some **restriction** (it is totally forbidden to fire before the synchronization configuration).

Many solutions exist in minimal time  $(2n - 1)$  [**Goto, Waksman, Balzer, Mazoyer**] or not, and in different lattices.

## Defective CA

**Umeo** shows that FSSP can be solved (with some restrictions) even if some cells are defectives :

- diagnosis circuit switch from a **normal cell** state to a
- **faulty cell** state (full-duplex channel).

## FTFSSP

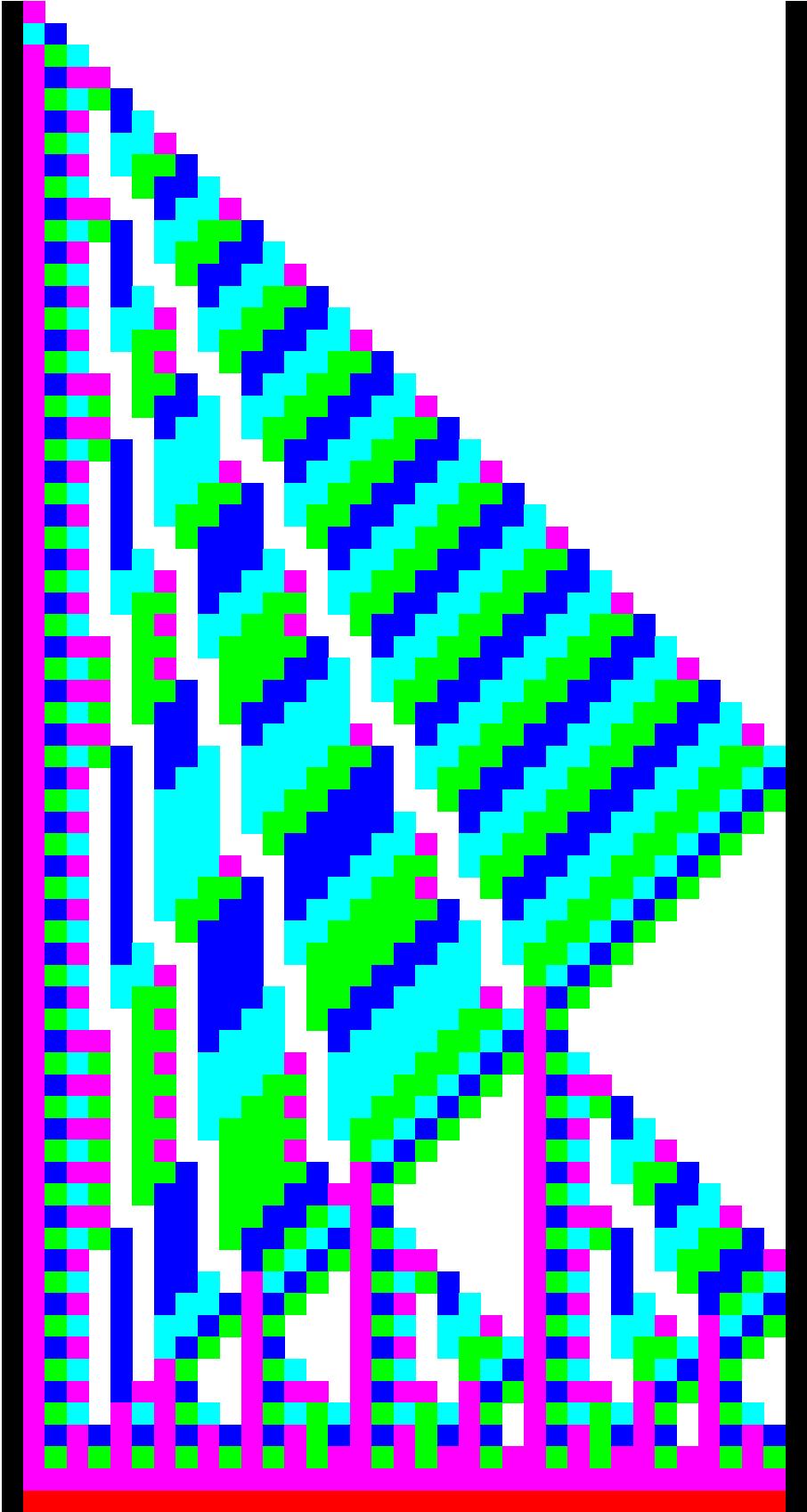
So one can extend the FSSP to FTFSSP in which some cells can be defectives. Umeo have found that :

1. A CA synchronizing a  $p$ -faulty  $n$ -line in  $2n - 2$  steps (minimal time) exists if  $p$  is known and if  $\forall i \in [1, p], n_i \geq m_i$ ,
2. A CA synchronizing a  $p$ -faulty  $n$ -line in  $2n - 2 + p$  steps (nearly minimal time) exists if  $p$  is unknown and if  $\forall i \in [1, p], n_i \geq m_i \wedge n_i + m_i \geq p - i$ .

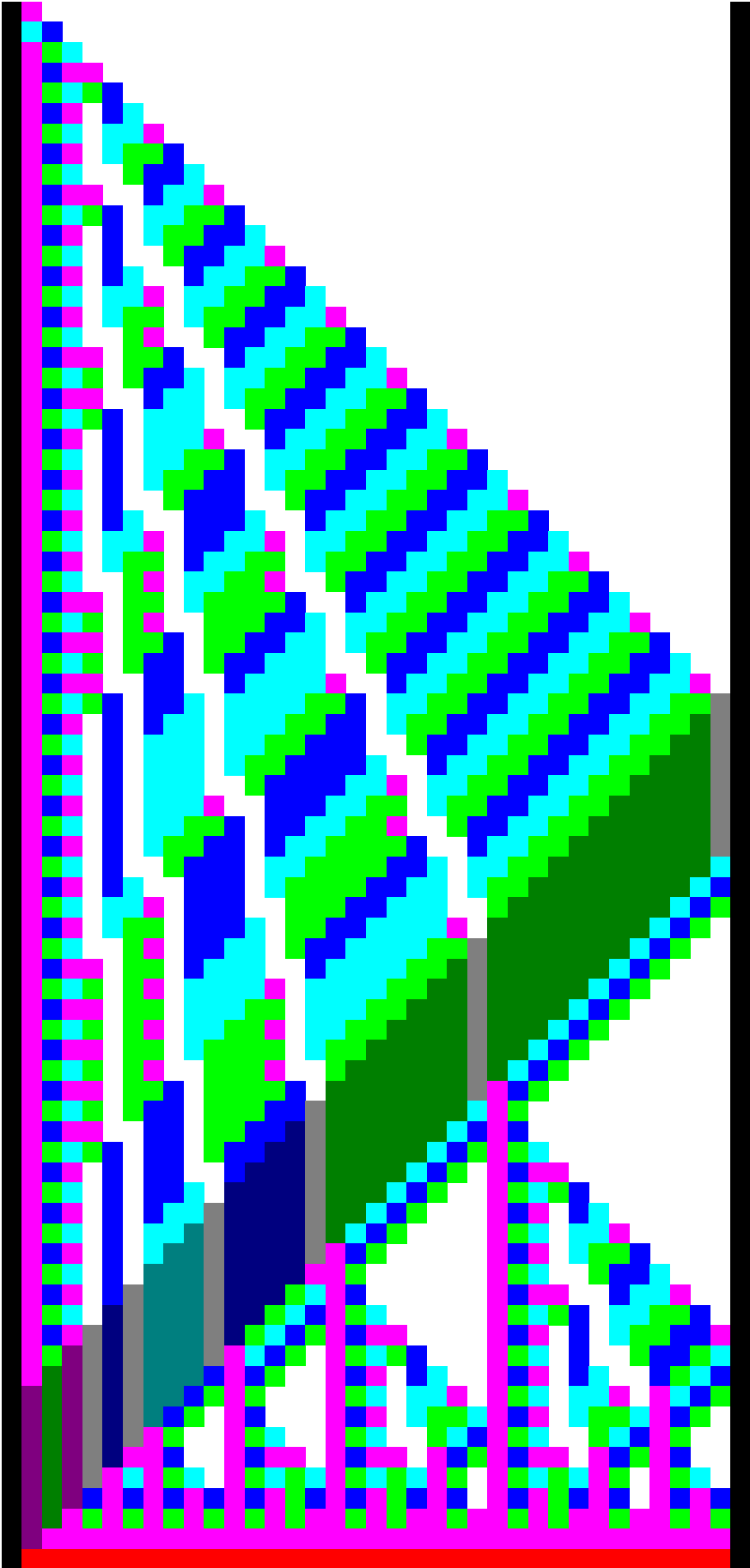
## Freezing/Thawing

- **Freezing process** : a mechanism which permits the computation to be stopped for a good while.
- **Thawing process** : a mechanism which permits a freezed computation to be warmed up.

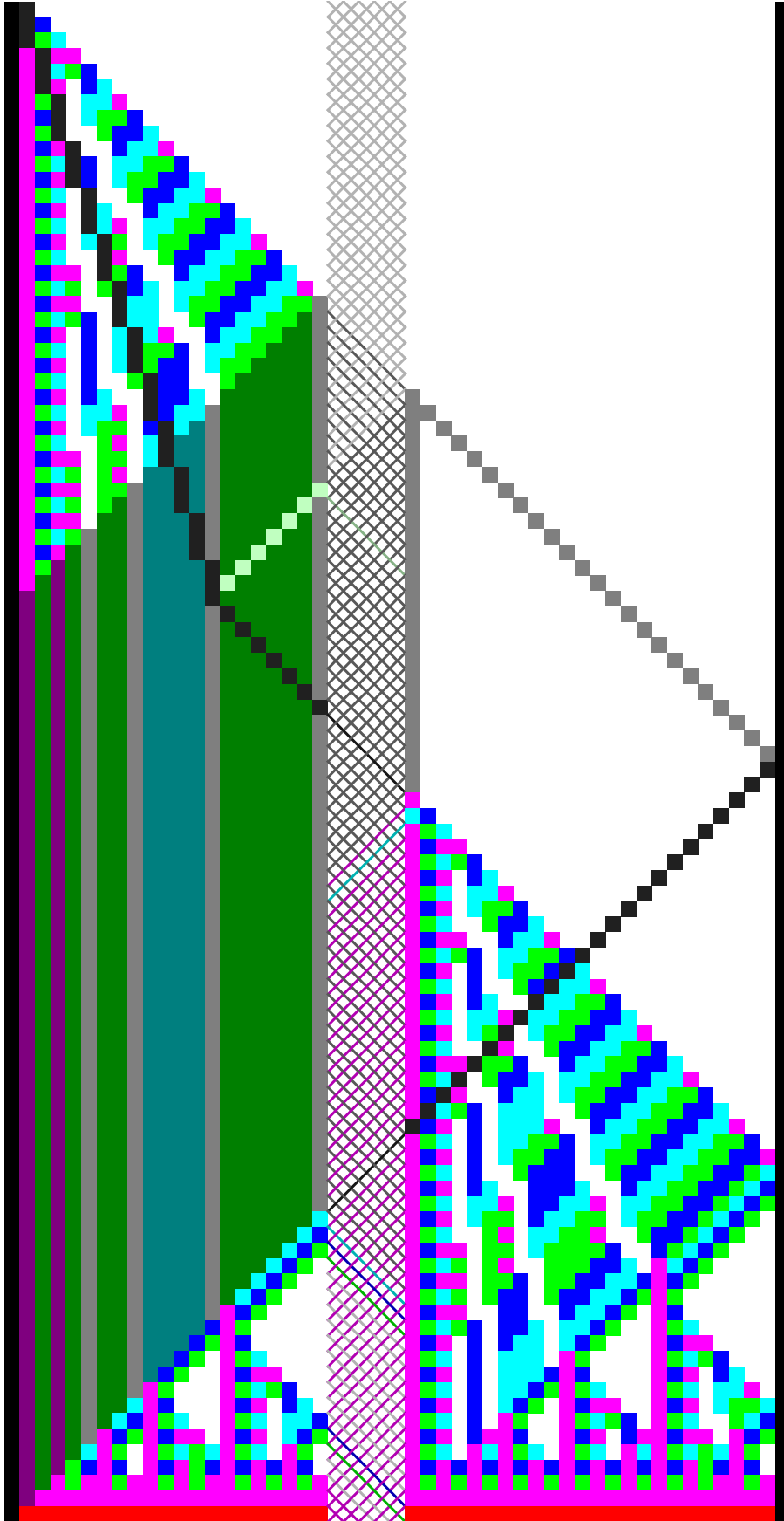
Mazoyer's solution synchronizing a line with 35 cells :



Umeo's freezing/thawing process on a Mazoyer's solution syn-  
chronizing a line with 35 cells :



Umeo's FTFSSP in a 2-faulty 49-line with  $n_1 = 20$ ,  $m_1 = 5$  and  $n_2 = 24$ :



## About the number of states...

Mazoyer's solution is can be build with only 6 states (general, quiescent, 3 auxiliary states, fire) and 115 transitions.

We need only 11 states for Umeo's Freezed-Thawed FSSP (general, quiescent, 3 auxiliary states, fire, freezed general, freezed quiescent, 3 freezed auxiliary states), provided that the stimulus are external. No *grouping* is necessary.

A simple implementation of a 1-faulty Umeo's solution needs about 60 states :

- 5 (Mazoyer's states)
- 5 (Umeo's states)
- $3 \times 10$  (speed  $\frac{1}{3}$  on top of M or U)
- $2 \times 5$  (2 speed 1 on top of U)
- 1 (speed 1 main signal)
- $1 \times 5$  (speed 1 on top of M)
- and some auxiliary states...

About 1300 transitions are defined.

## FTFSSP

And what if  $m_i \geq n_i$ ?

1. A CA synchronizing a  $p$ -faulty  $n$ -line in

$2n - 2 + \sum_{i=1}^{i=p} (m_i - n_i)$  steps exists if:

- $p$  is unknown
- $\forall i \in [1, p], m_i \geq n_i$
- $\forall i \in [1, p], n_{i+1} \geq m_i - n_i$
- $\forall i \in [1, p], 2m_i \geq p - i + 1$
- $n_{r+1} \geq 2m_r - n_r$

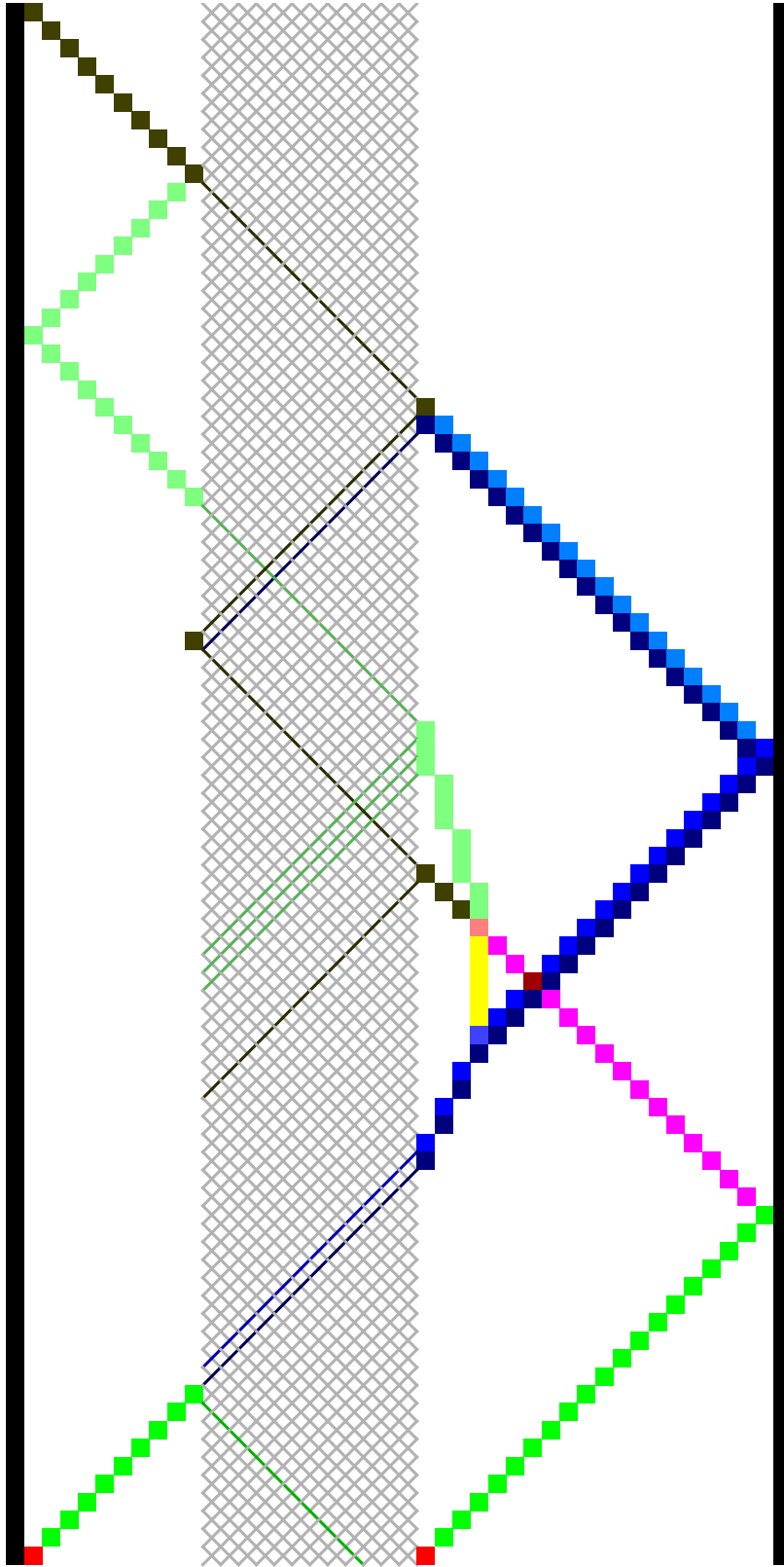
### About the number of states...

More than 80 states are needed to build the skeleton of such a solution :

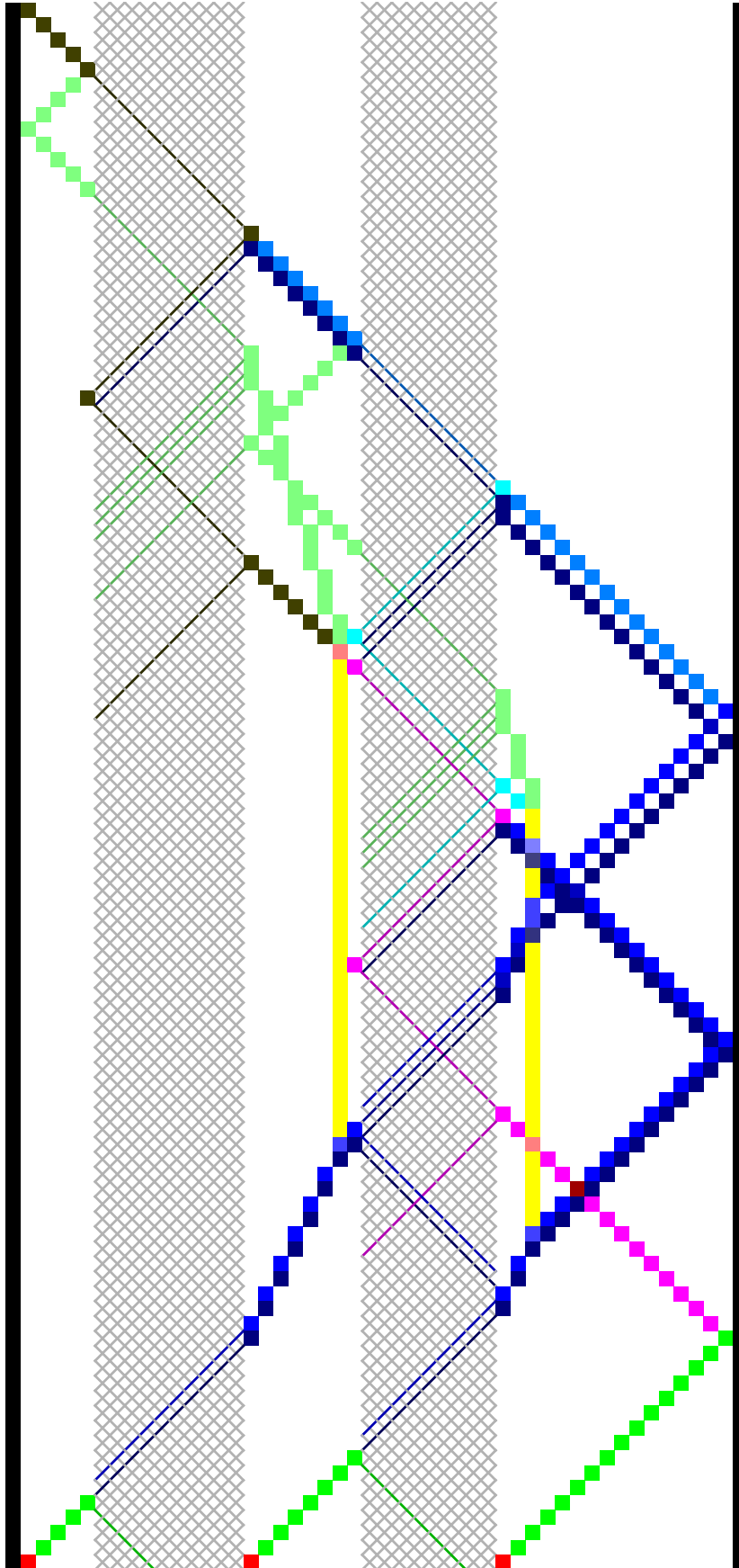
- many cases have not yet been tested,
- no freezing-thawing process implemented,
- no synchronization mixed-in.

About a thousand states are probably needed, and certainly less than 2 thousand.

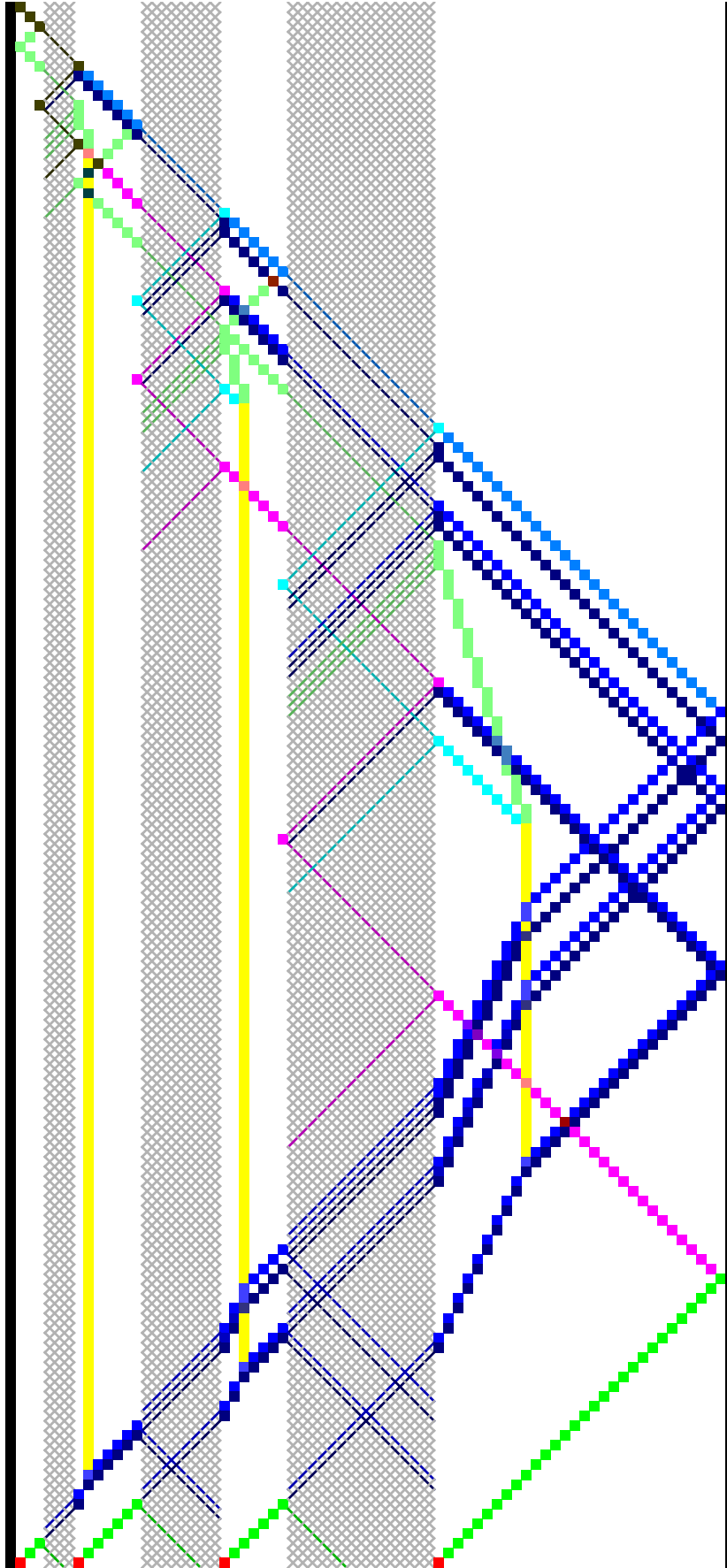
Synchronizing a 1-faulty 42-line ( $n_1 = 10$ ,  $m_1 = 12$ ,  $n_2 = 20$ ):



Synchronizing a 2-faulty 48-line ( $n_1 = 5, m_1 = 10, n_2 = 8, m_2 = 9, n_3 = 16$ ):



Synchronizing a 3-faulty 73-line ( $n_1 = 3, m_1 = 3, n_2 = 7, m_2 = 8, n_3 = 7, m_3 = 15, n_4 = 30$ ):



## Standardization

In general:

$$- \forall i \in [1, p], n_i \geq m_i \vee n_i < m_i$$

So we have :

1. A CA synchronizing a  $p$ -faulty  $n$ -line in

$2n - 2 + \sum_{i=1}^{i=p} (m_i - n_i)$  steps exists if :

-  $p$  is unknown

$$- \forall i \in [1, p], n_{i+1} \geq |m_i - n_i|$$

$$- \forall i \in [1, p], 2m_i \geq p - i + 1$$

$$- n_{r+1} \geq 2\max(m_r, n_r) - \min(m_r, n_r)$$

## Some questions...

- What about mixing Umeo's (when  $m_i \leq n_i$ ) and Yunes' (when  $m_i > n_i$ )?
- What about optimization?