Identification of binary CAs

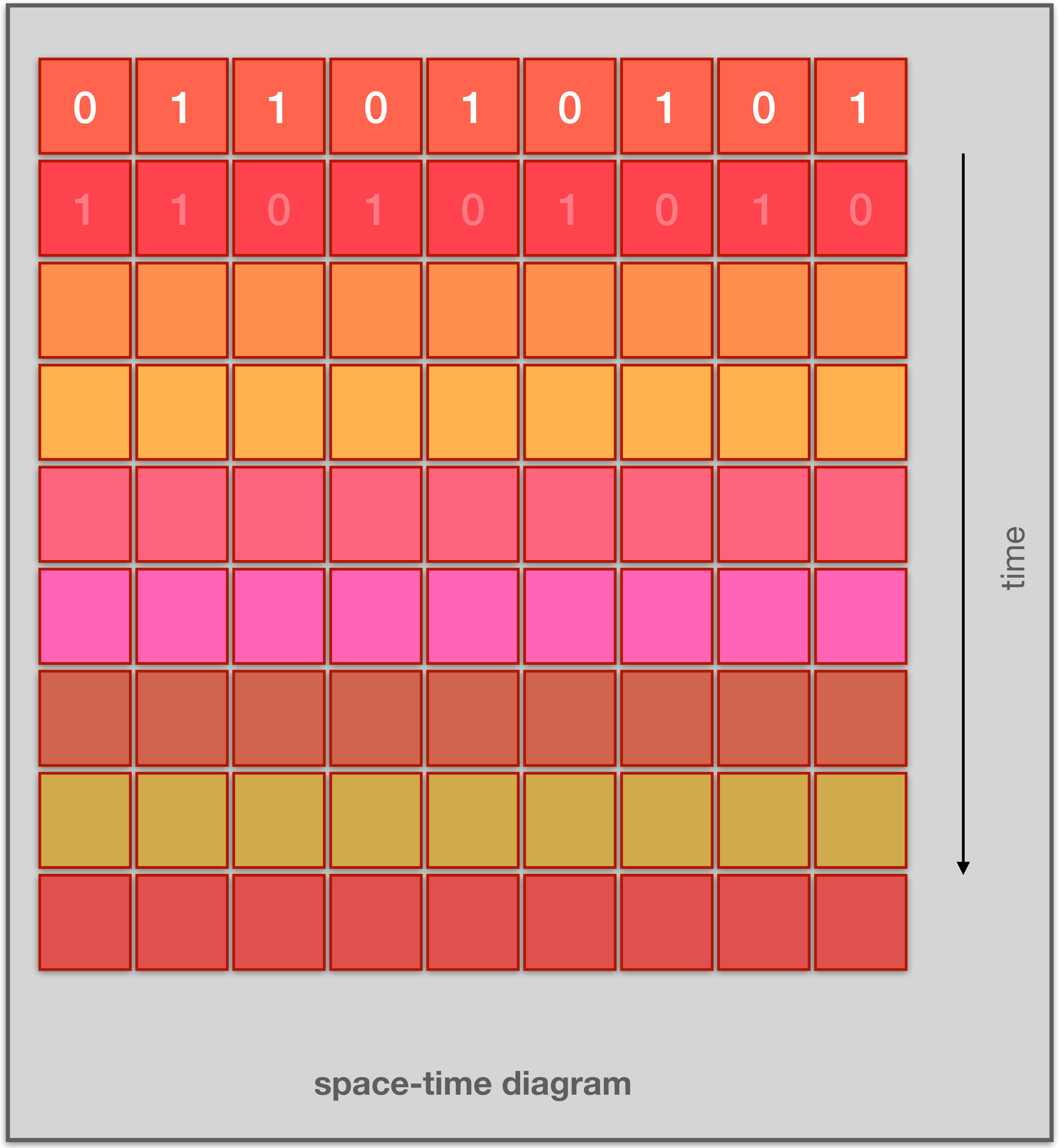
based on incomplete and *noisy* observations

Witold Bołt, 28th May 2022

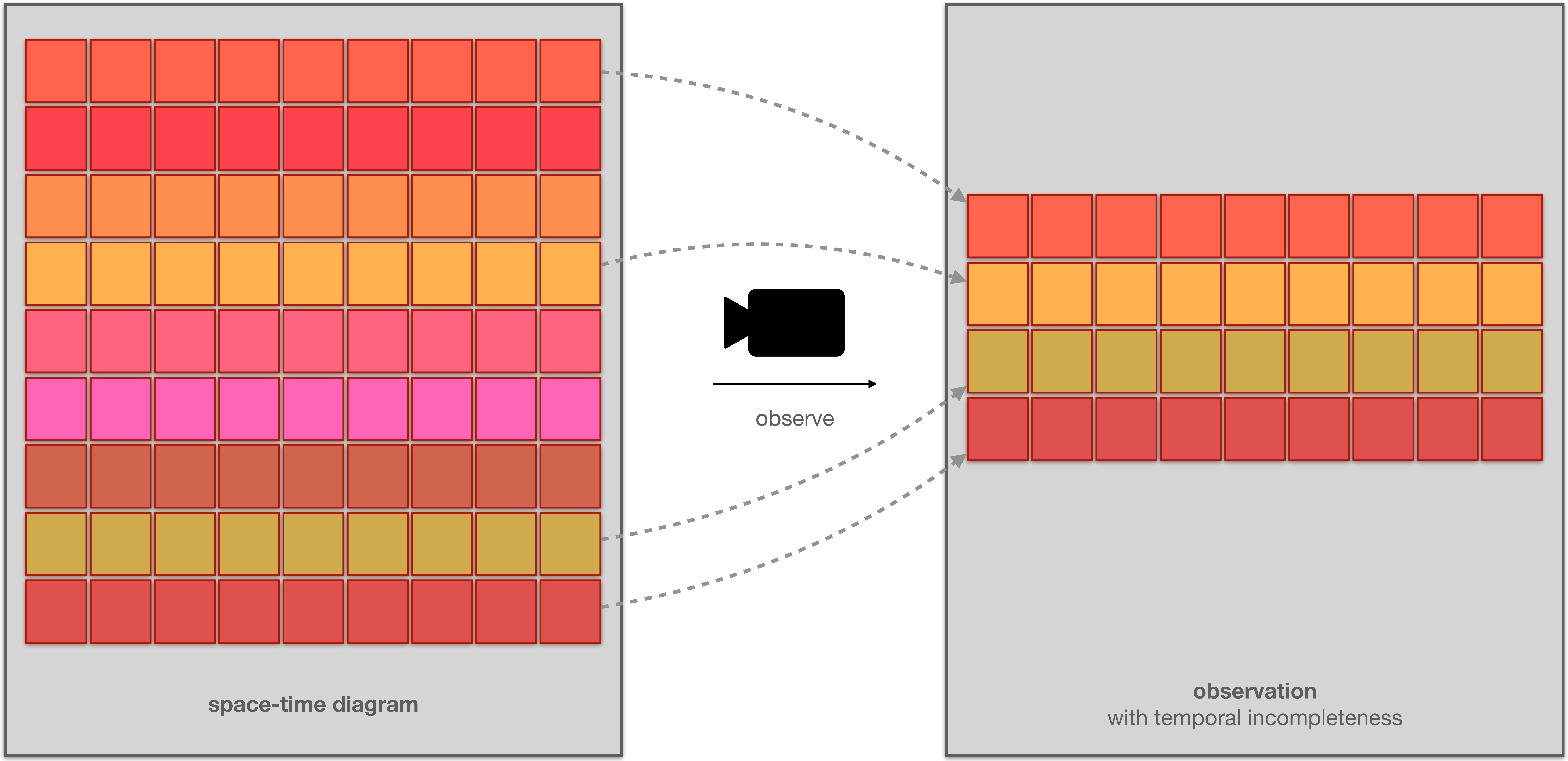
JitTeam™

Definitions

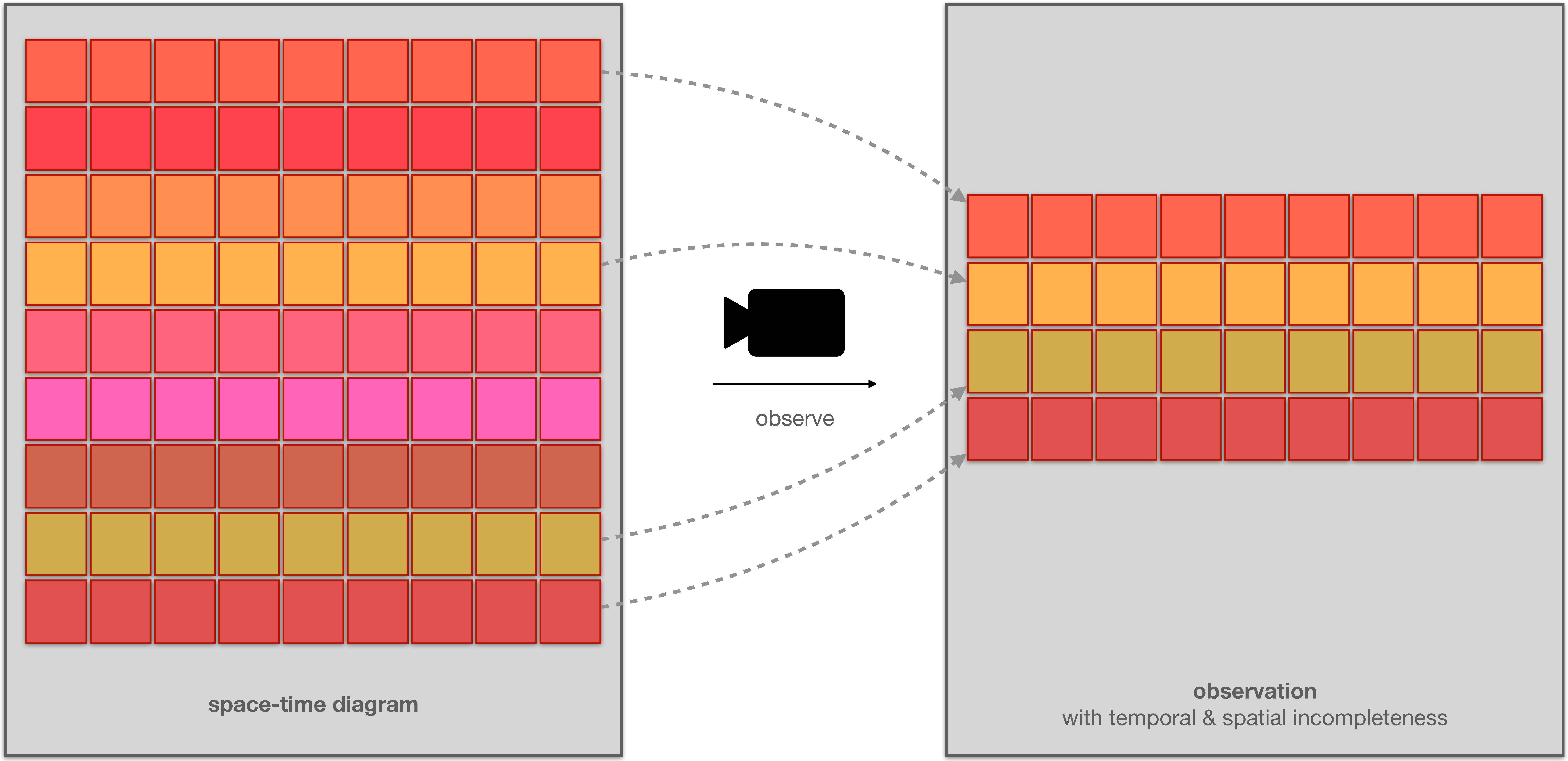
- 1D, binary CA with finite number of cells
- Finite time
- Periodic boundary conditions
- Deterministic
- Local rule with radius r > 1
- Lookup Table (LUT) = binary vector
- Space-time diagram



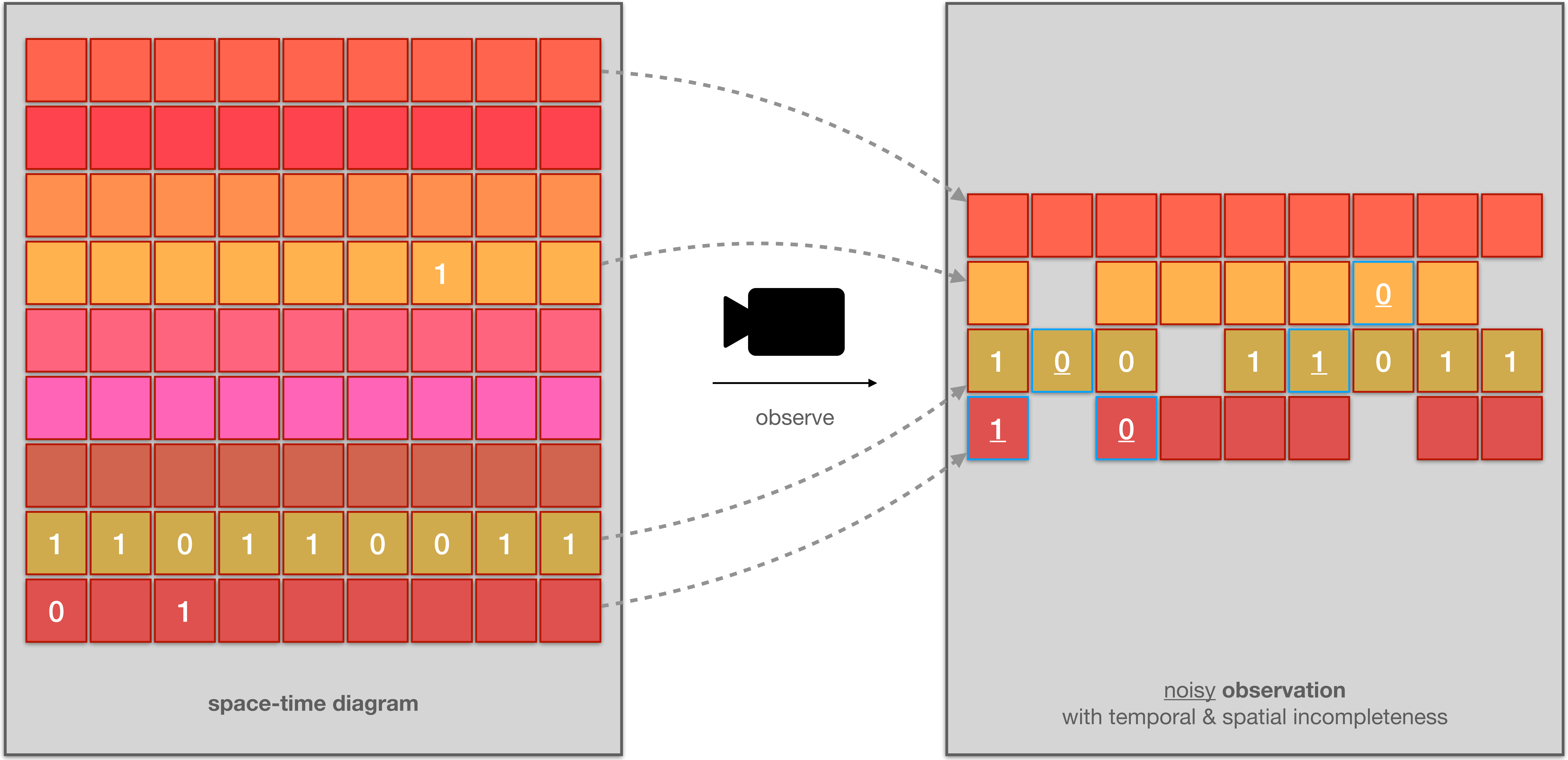
Observation of a space-time diagram



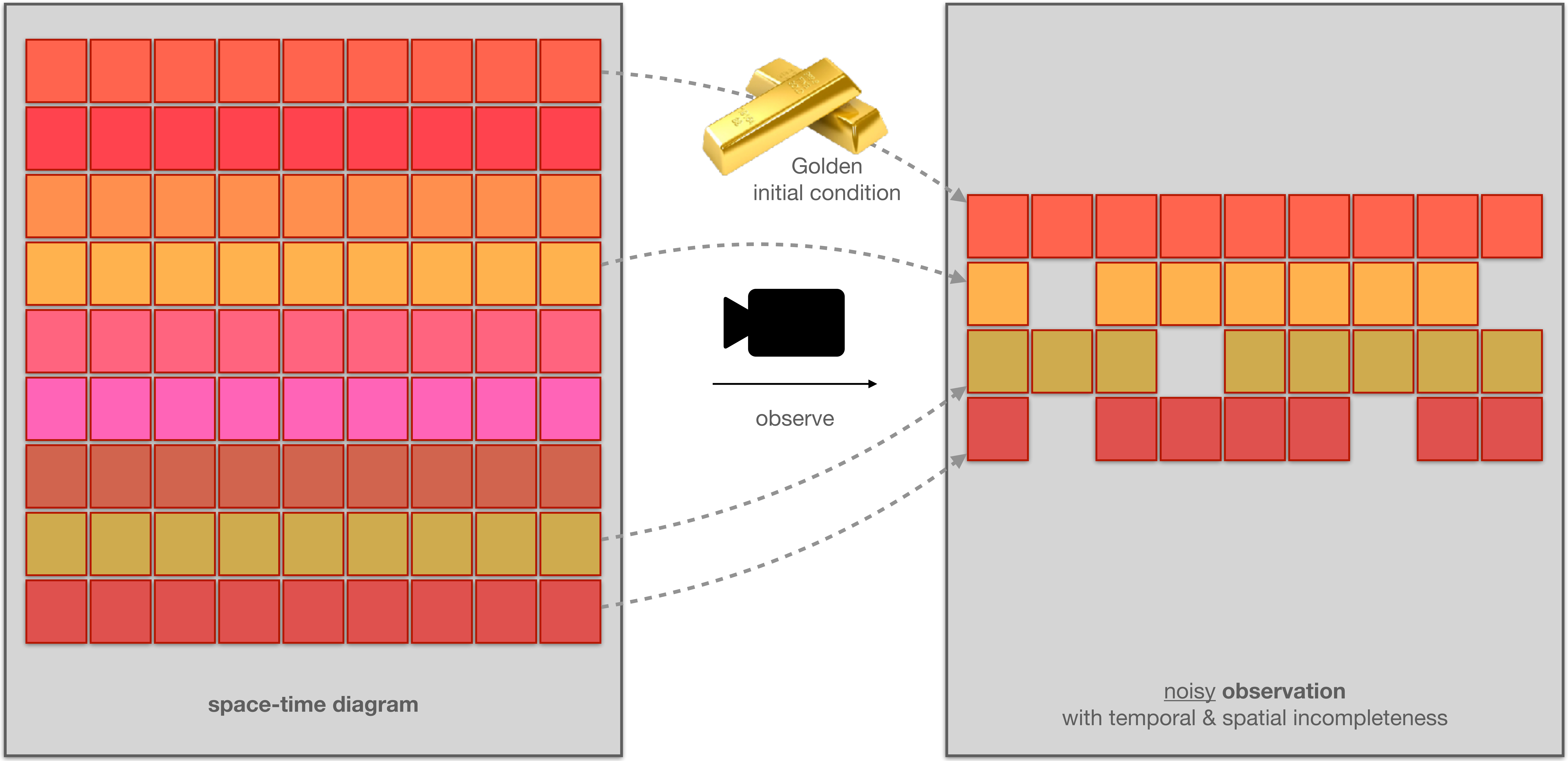
Observation of a space-time diagram

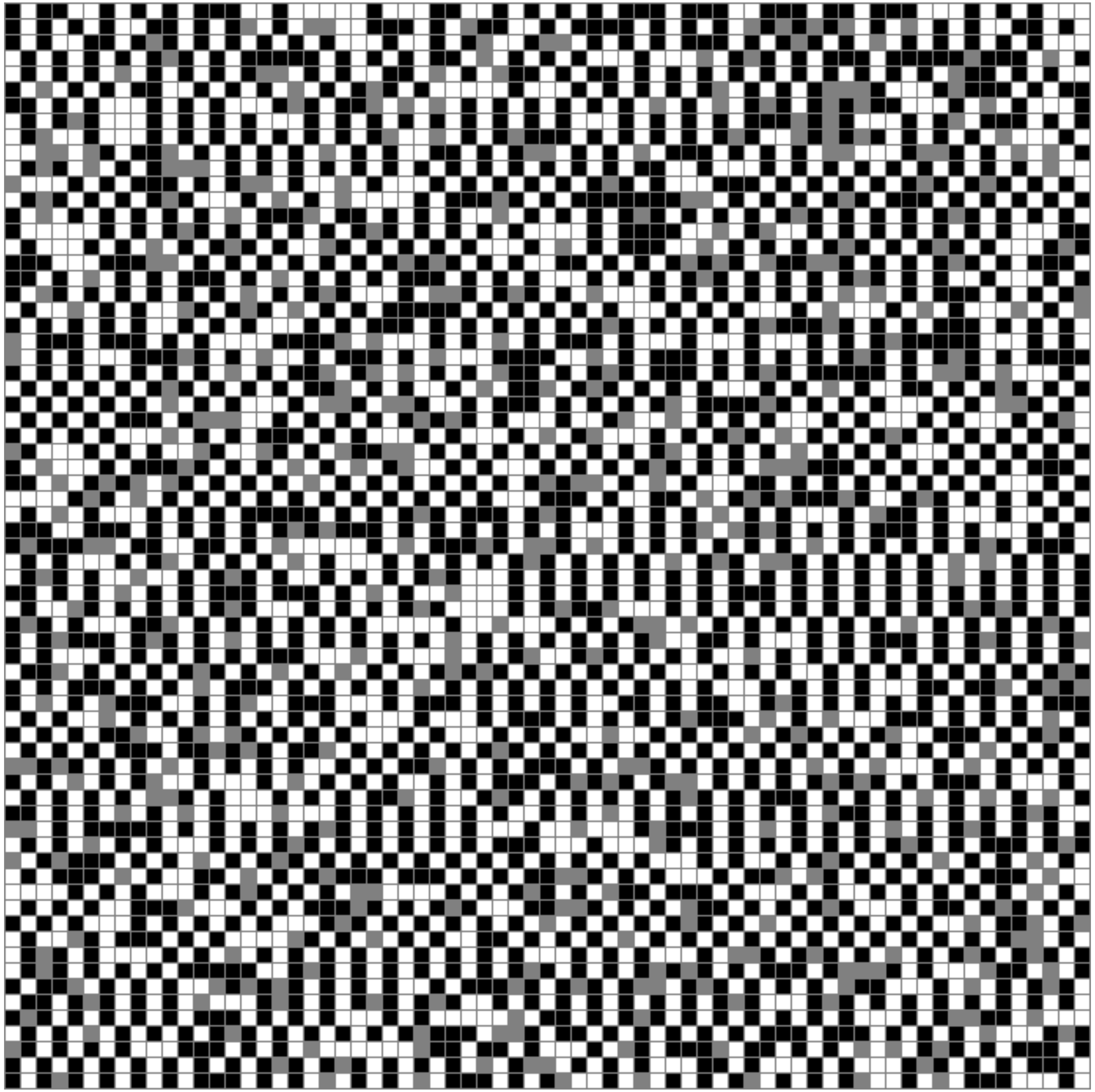


Observation of a space-time diagram

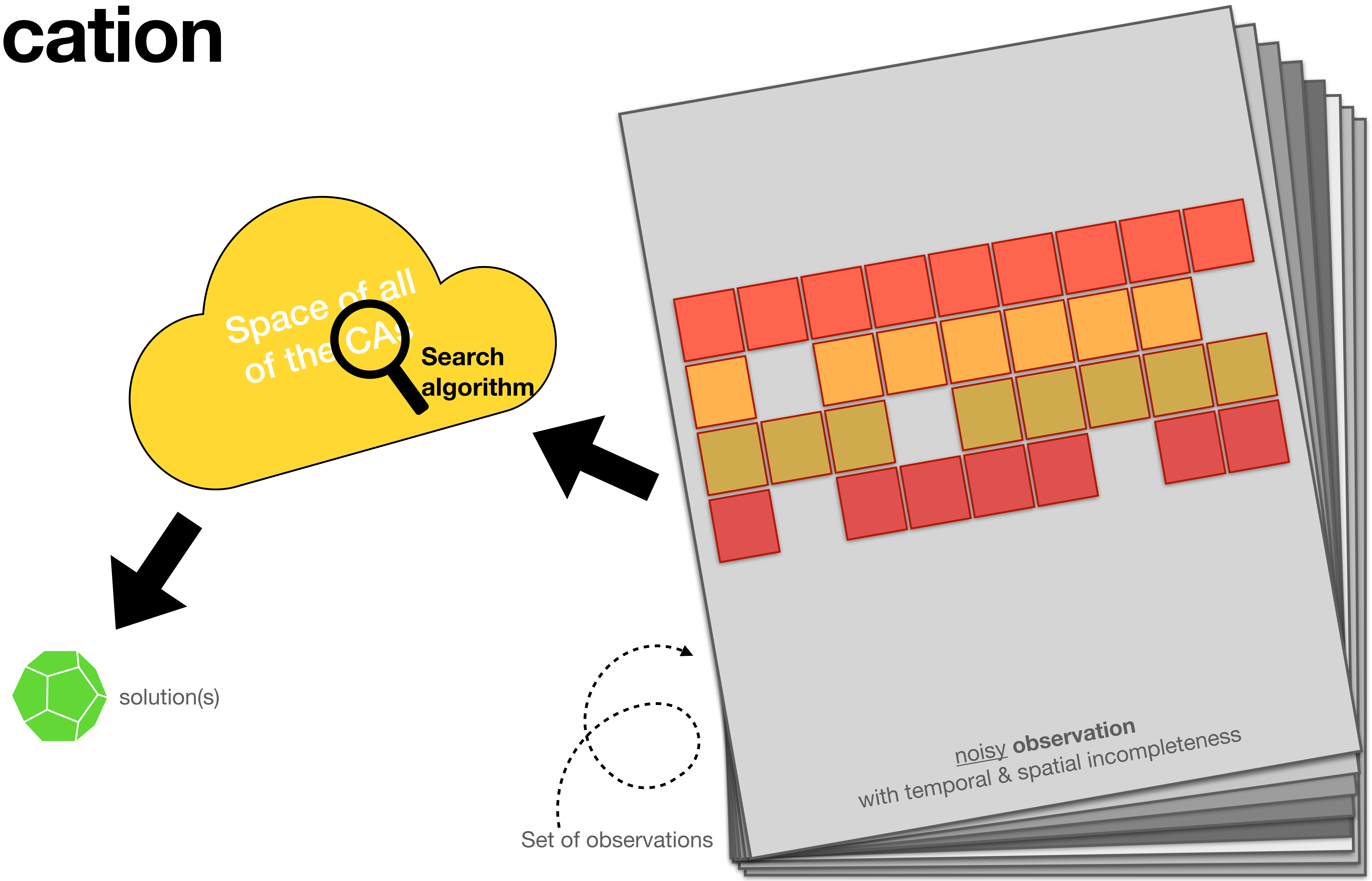


Observation of a space-time diagram



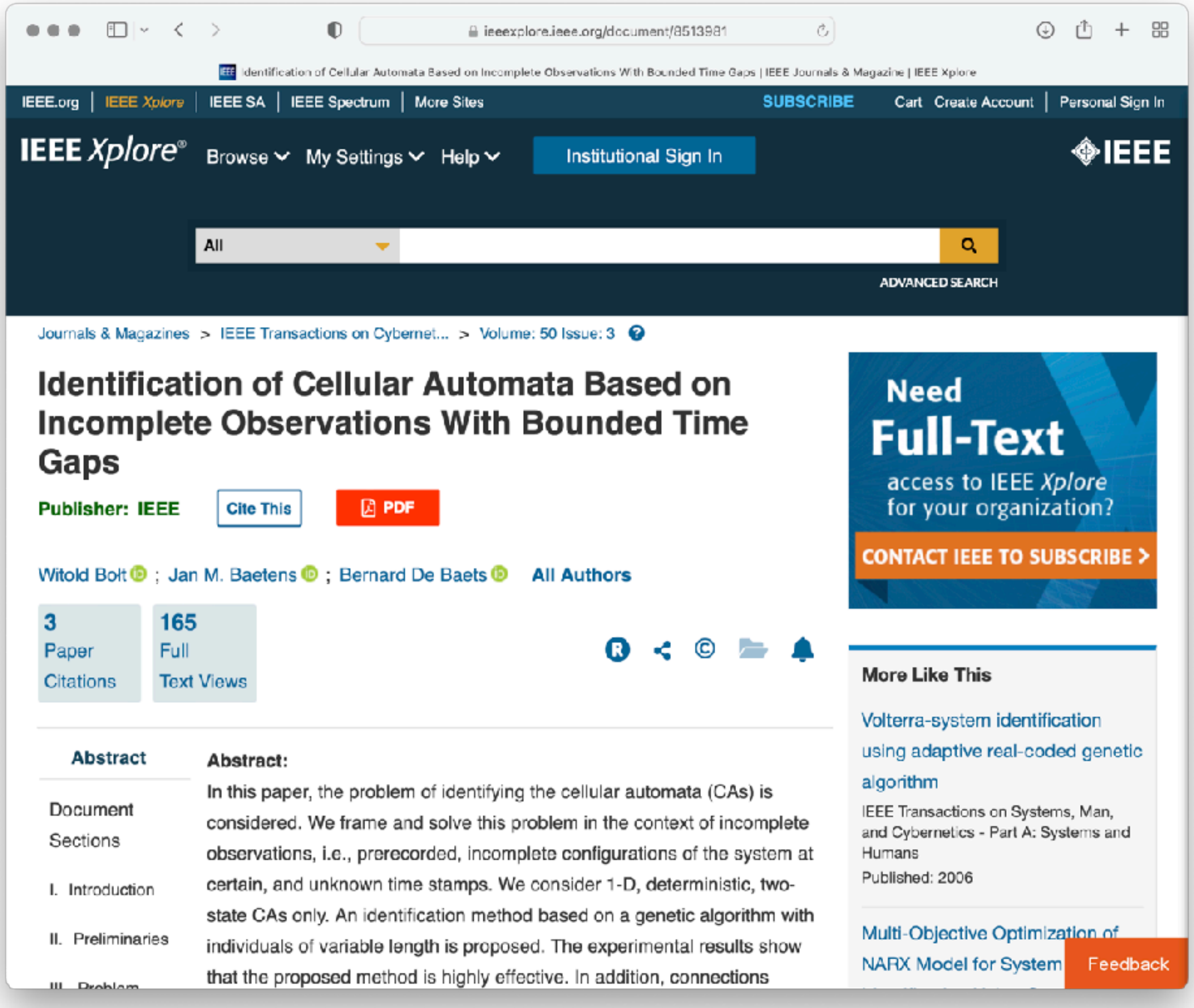


Identification



Algorithm

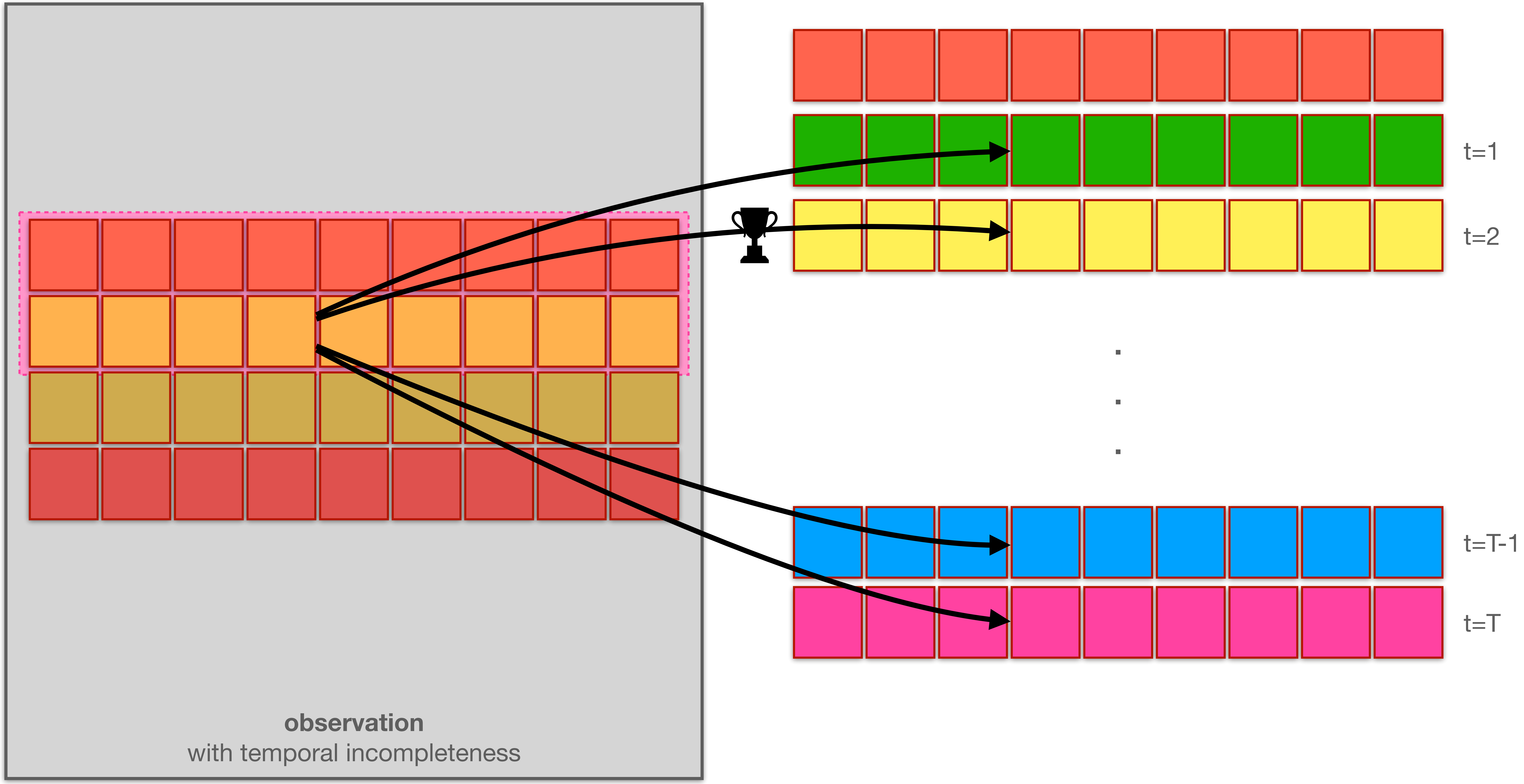
- Genetic Algorithm
- Population of candidate CAs encoded as LUTs
- Variable radii (between r_{min} and r_{max})
- Multiple small tricks
  - Subset of observations used
  - Elite survival
  - Adaptive mutation rate
  - Re-starting
- github.com/houp/identify (C, Python)



https://ieeexplore.ieee.org/document/8513981

Fitness function

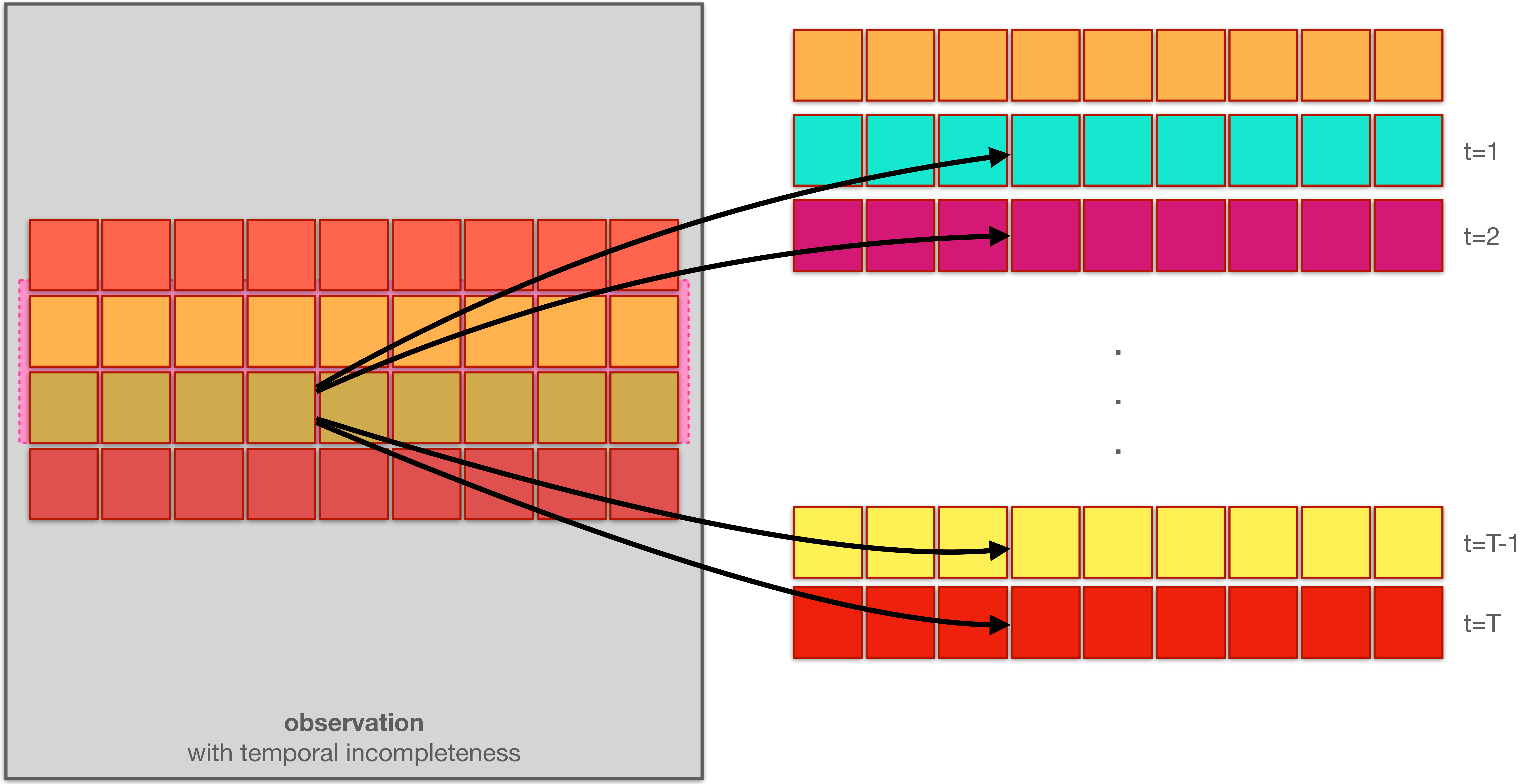
simple case: observations with temporal incompleteness



- **error** := 0
- cell-wise compare observation vs. outcome for each **t=1…T** and store to **error[t]**
- Find “best” time step $\mathbf{t}_{best}$ which minimizes **error[t]**
- Add **error[$\mathbf{t}_{best}$]** to **error**
- Go to next pair of rows in observation …

Fitness function

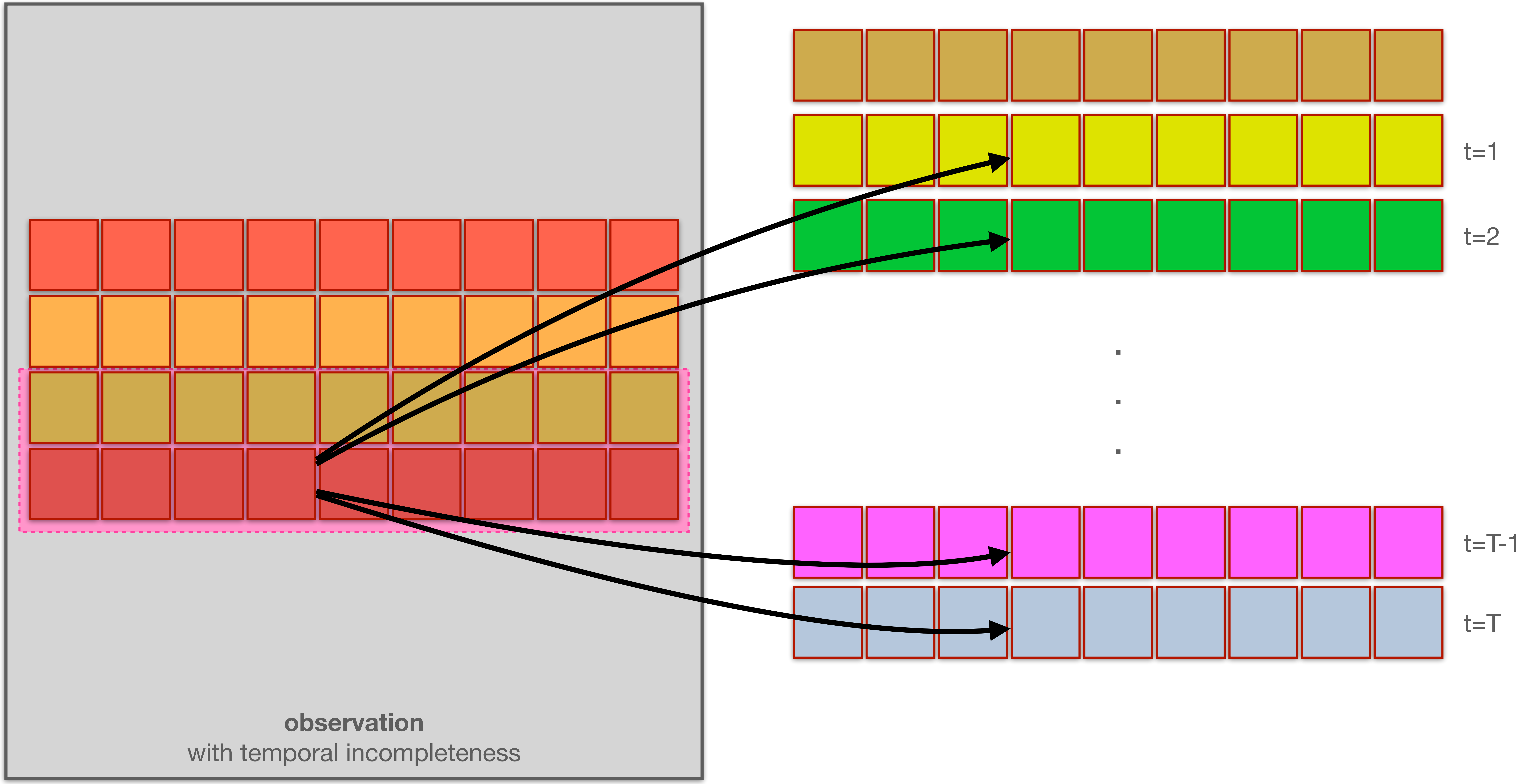
simple case: observations with temporal incompleteness



- cell-wise compare observation vs. outcome for each **t=1…T** and store to **error[t]**
- Find "best" time step $\mathbf{t_{best}}$ which minimizes **error[t]**
- Add **error[$\mathbf{t_{best}}$]** to **error**
- Go to next pair of rows in observation …

Fitness function

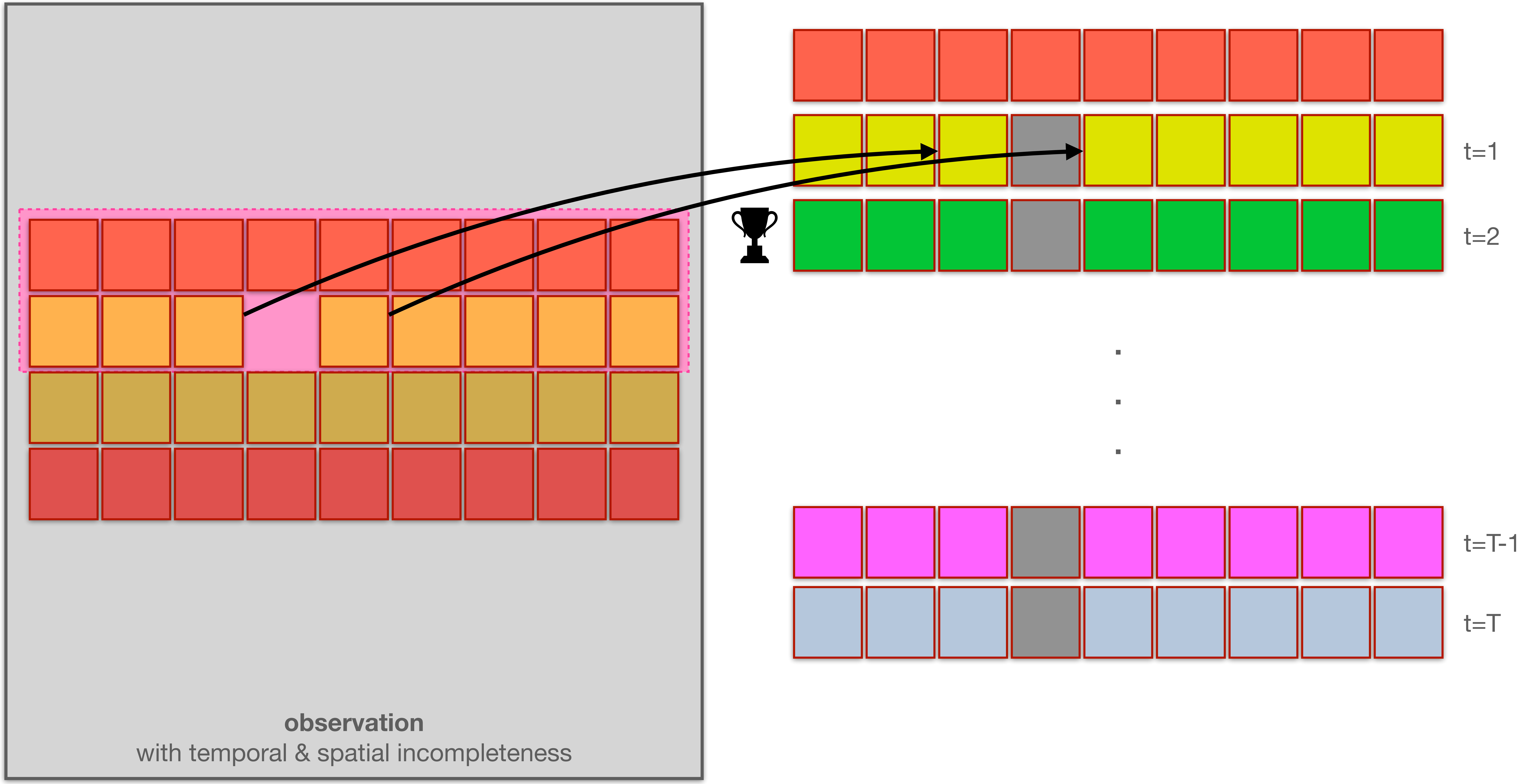
simple case: observations with temporal incompleteness



- cell-wise compare observation vs. outcome for each **t=1…T** and store to **error[t]**
- Find “best” time step $\mathbf{t}_{\text{best}}$ which minimizes **error[t]**
- Add **error[**$\mathbf{t}_{\text{best}}$**]** to **error**
- Return the fitness value as **1 - error / examined_cells**

Fitness function

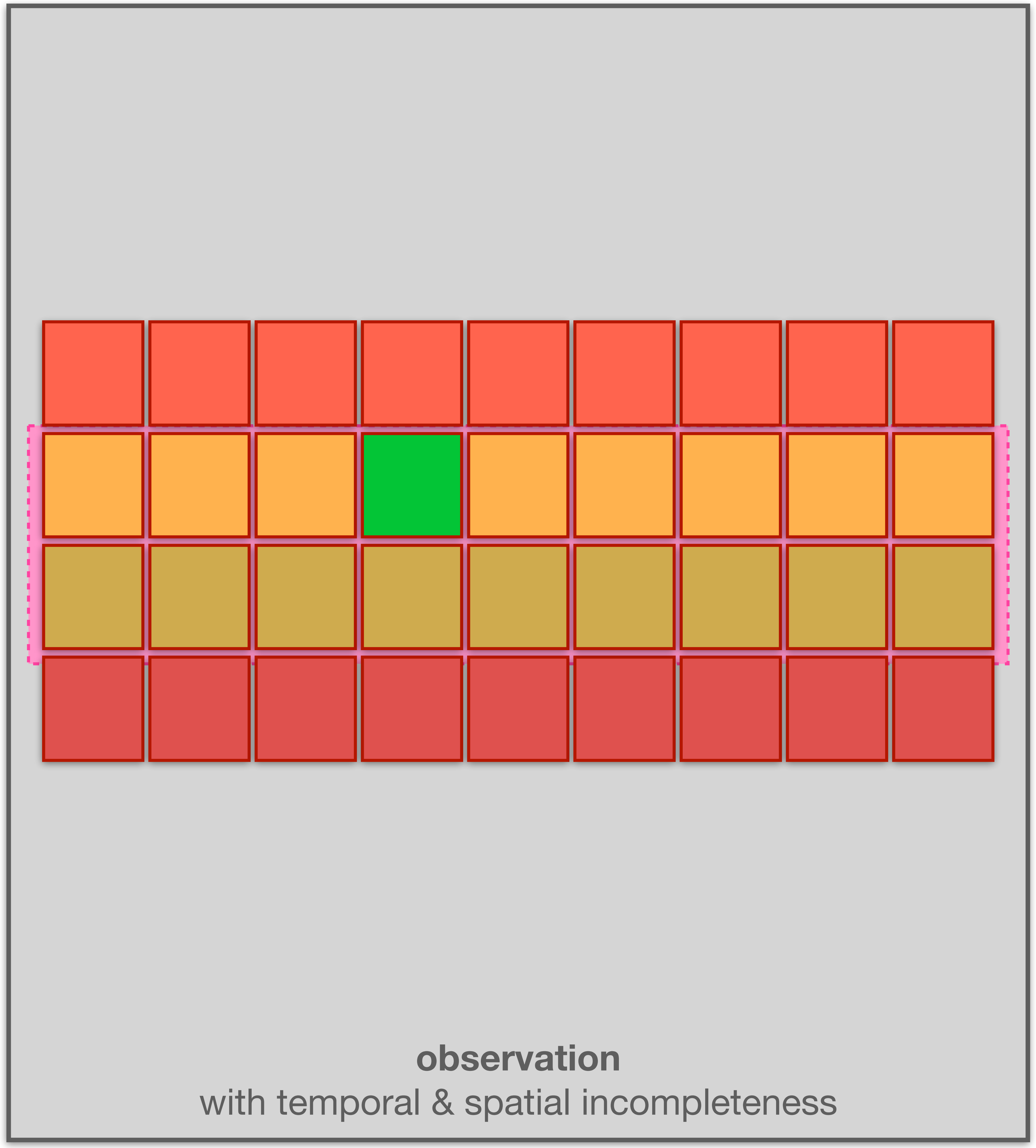
what happens in spatially incomplete case?

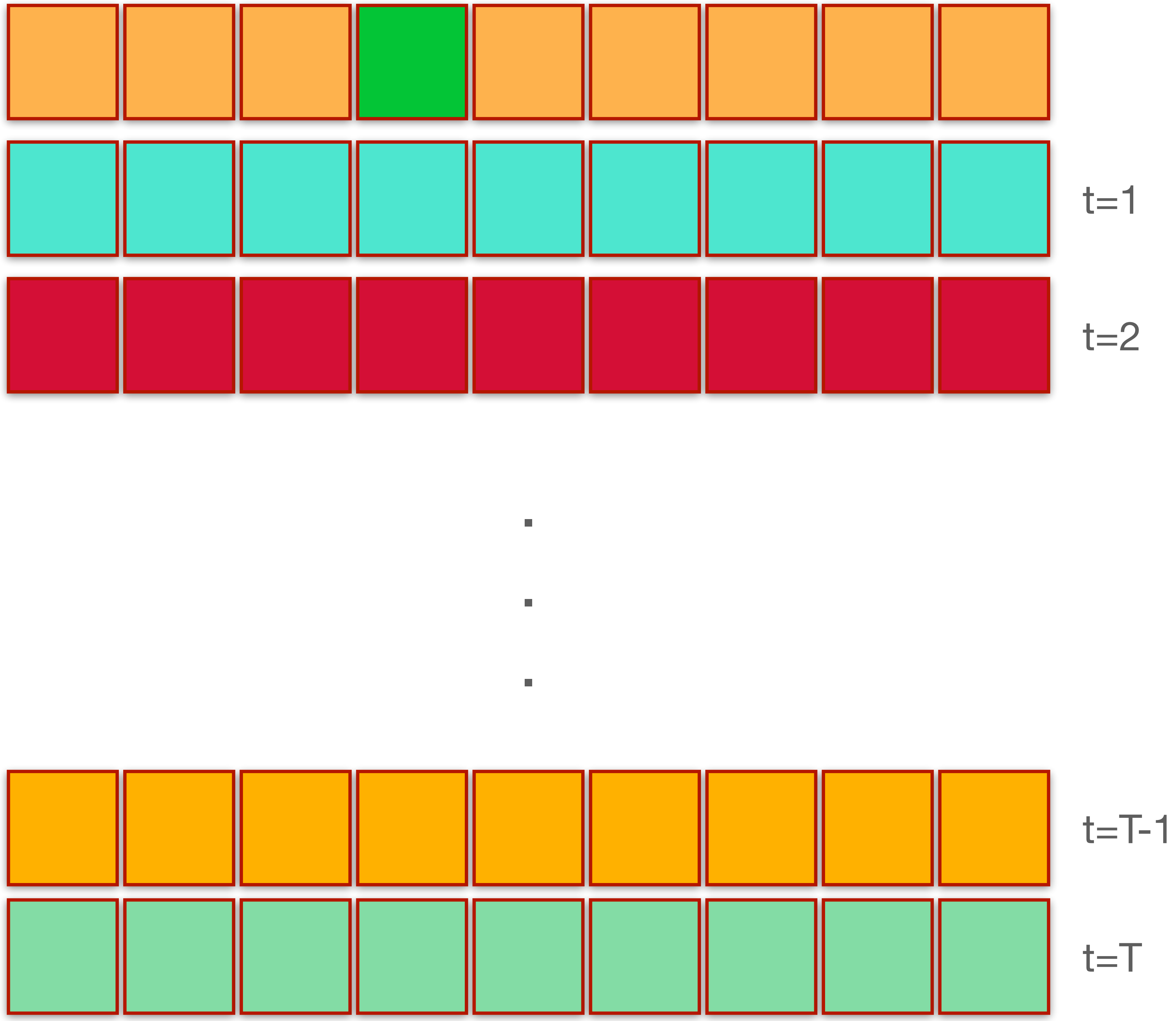


We **ignore** the missing cell in the comparison.

Fitness function

what happens in spatially incomplete case?

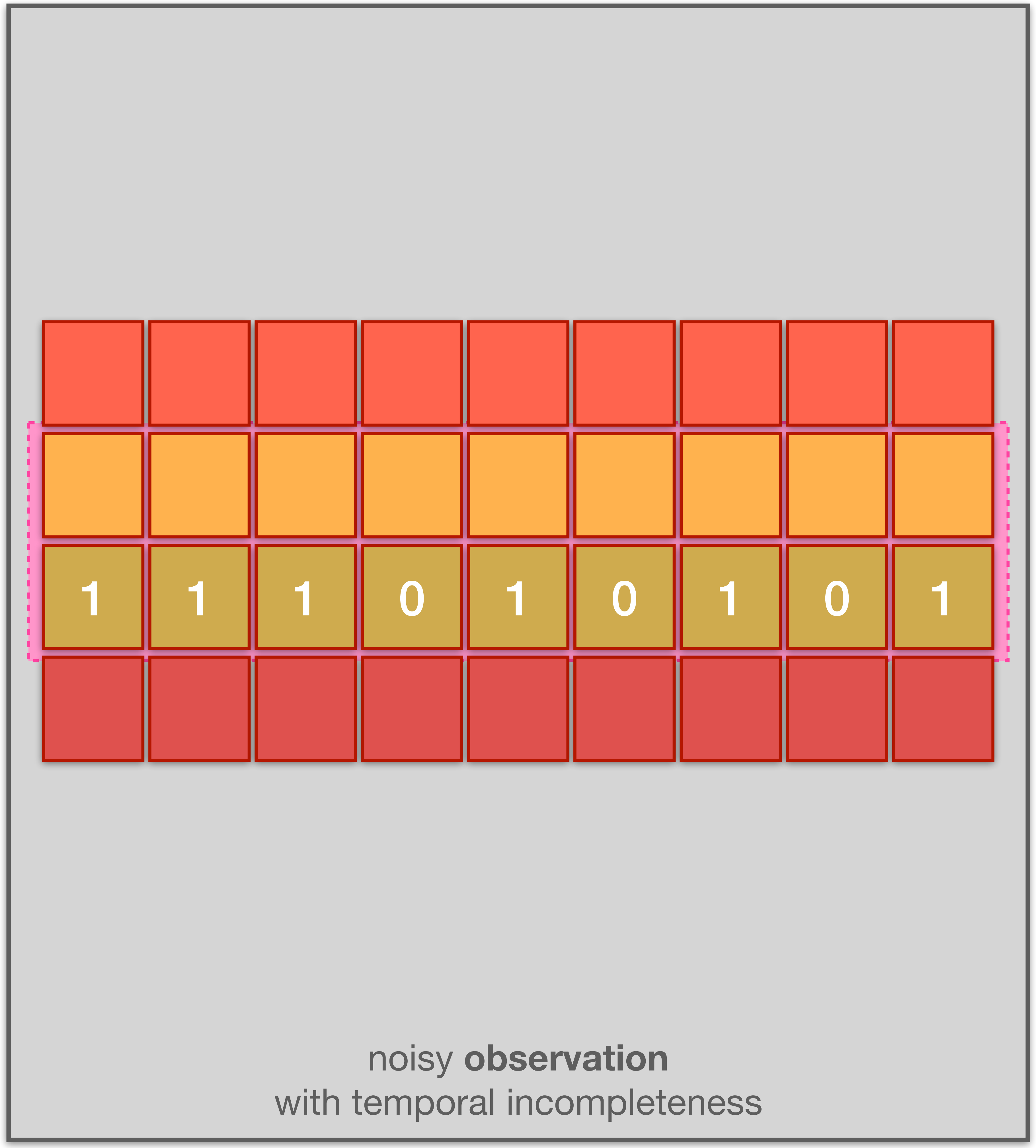


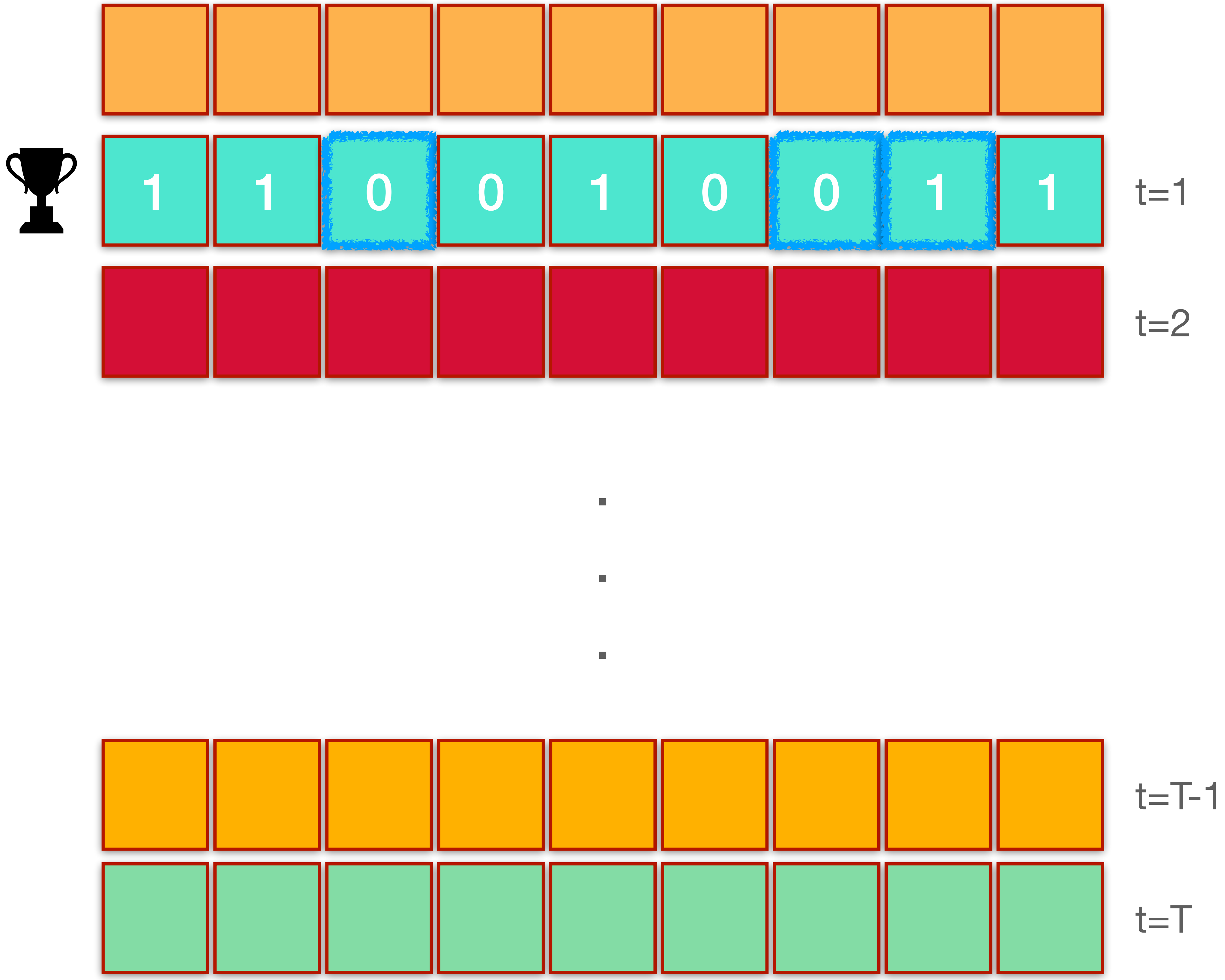


We **fill the gap** (for the purpose of comparison) with the value from the "best" row in the previous step.

Fitness function

what happens in the case of noisy observations?





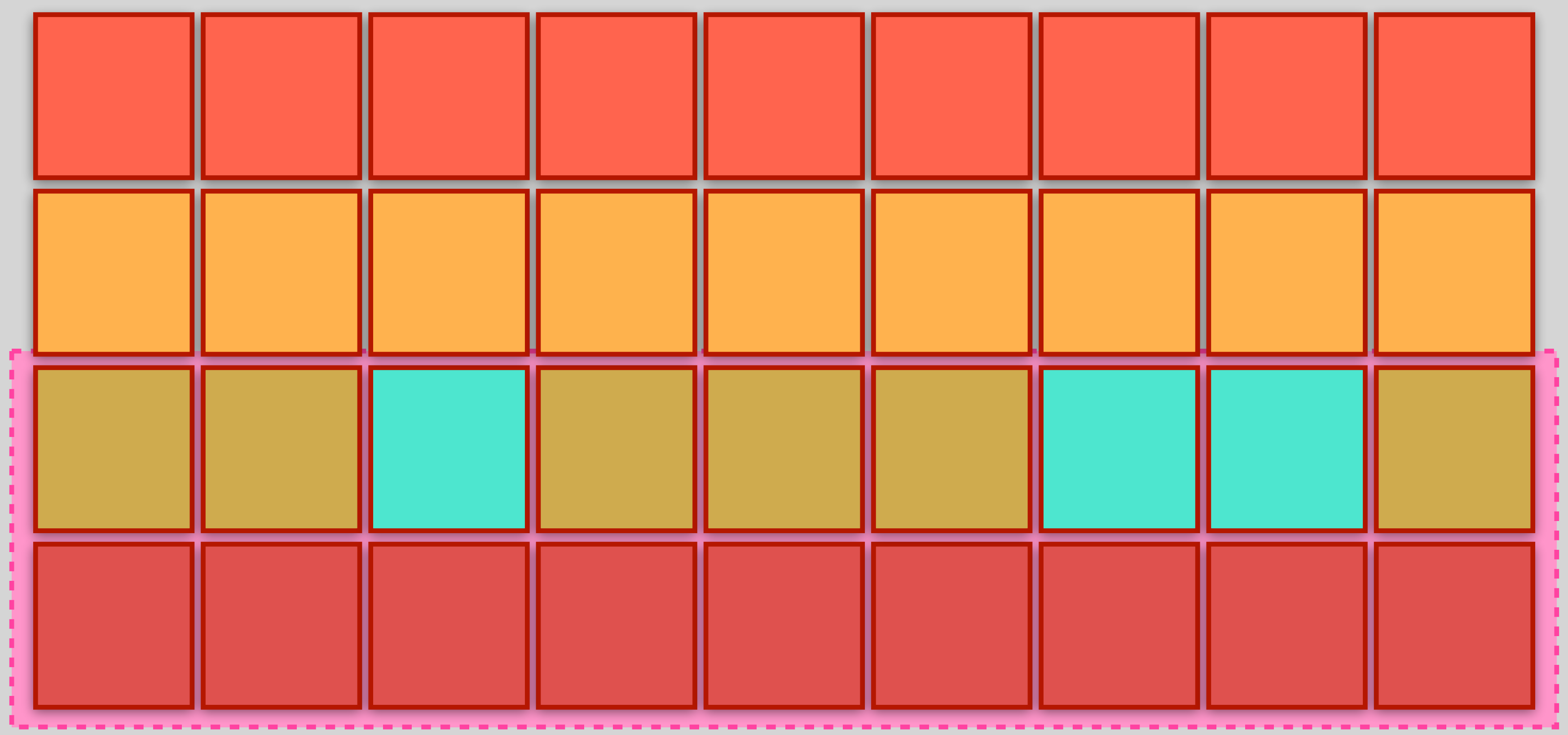
Number **C** > 0 is a correction budget set for each observation at the first step.

We pick t_{best} just as before.

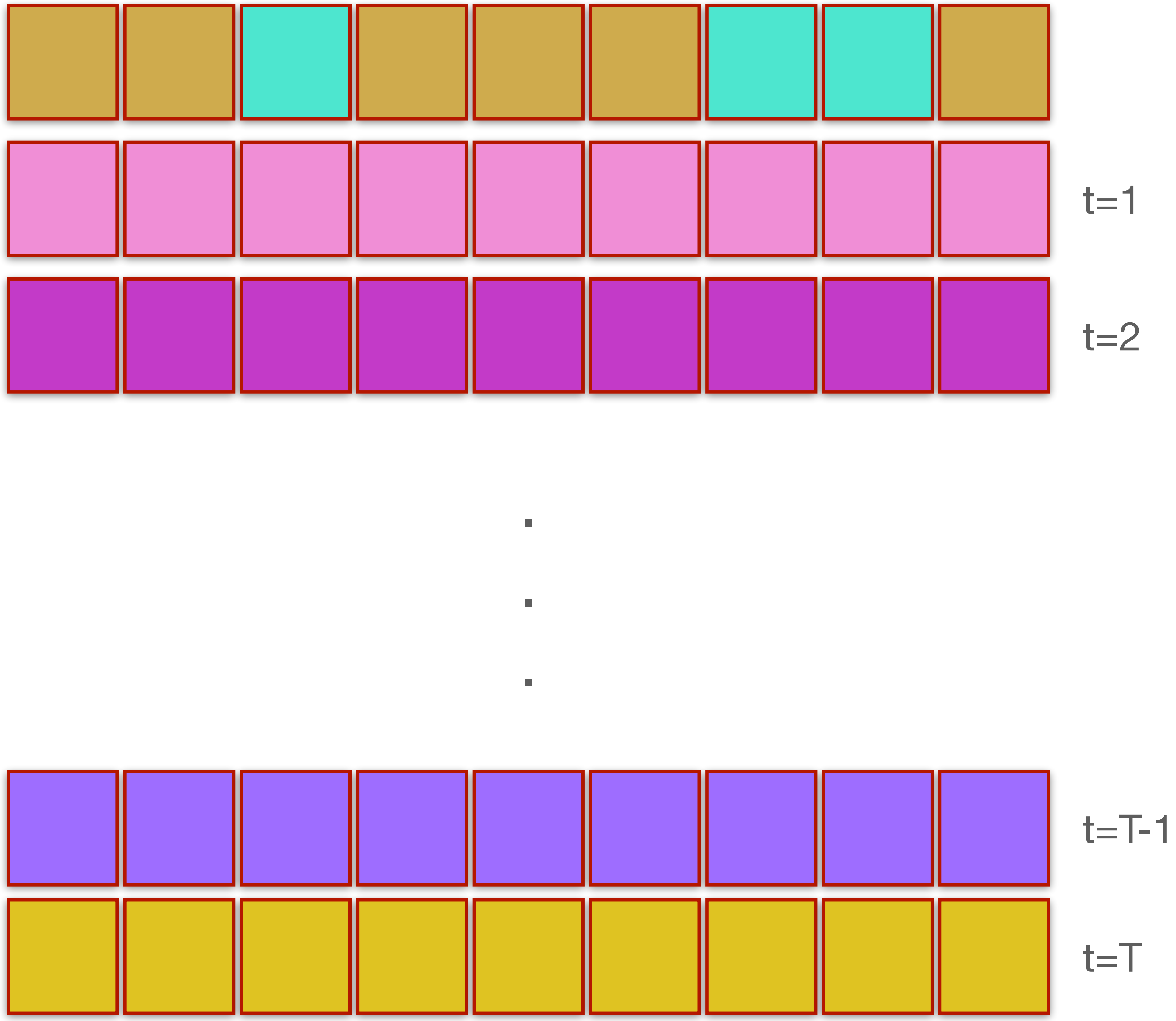
For cells with differing states we **introduce changes in the observation** following the outcomes from the rule.

Fitness function

what happens in the case of noisy observations?



noisy **observation**
with temporal incompleteness



Number **C** > 0 is a correction budget set for each observation at the first step.

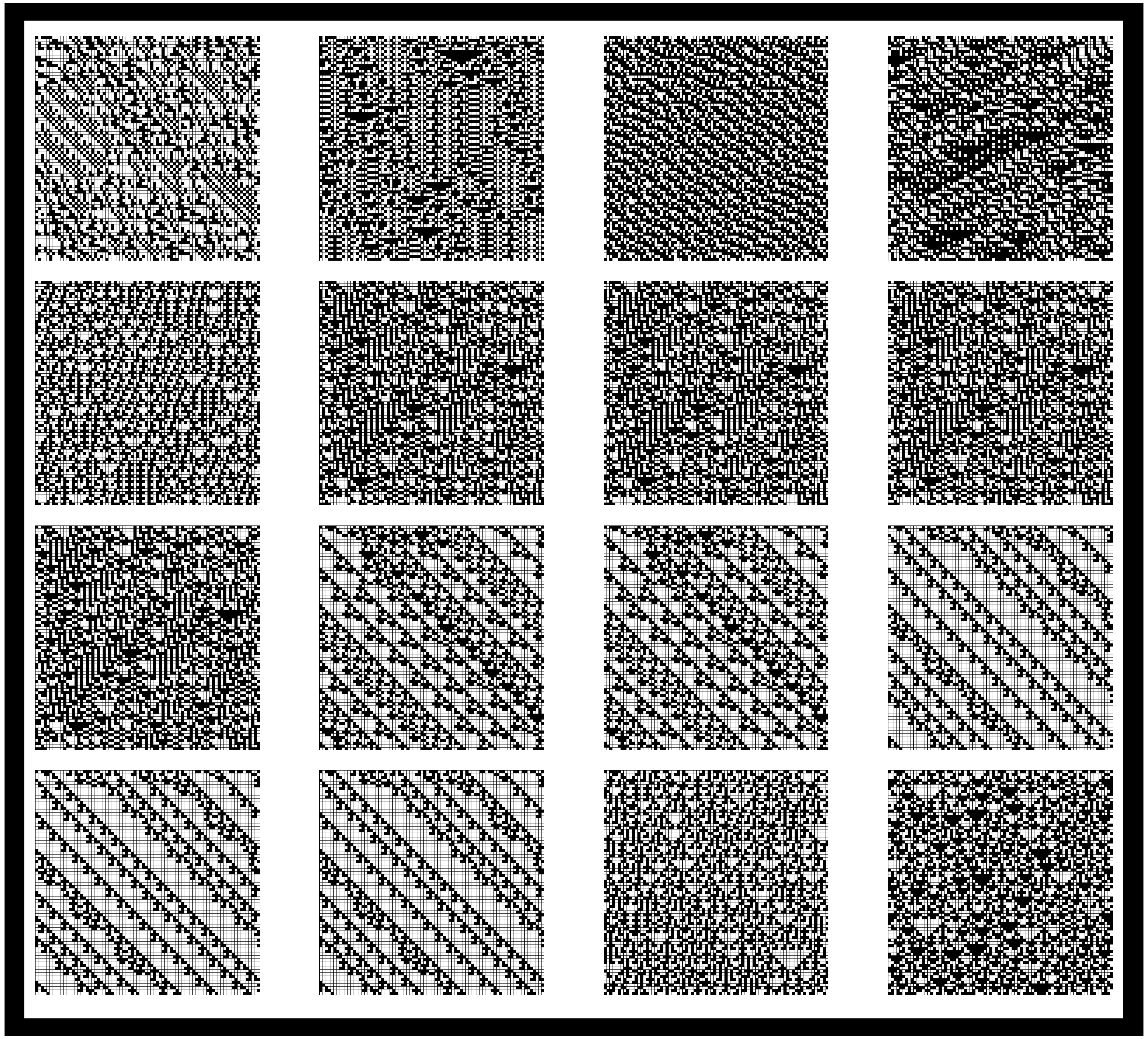
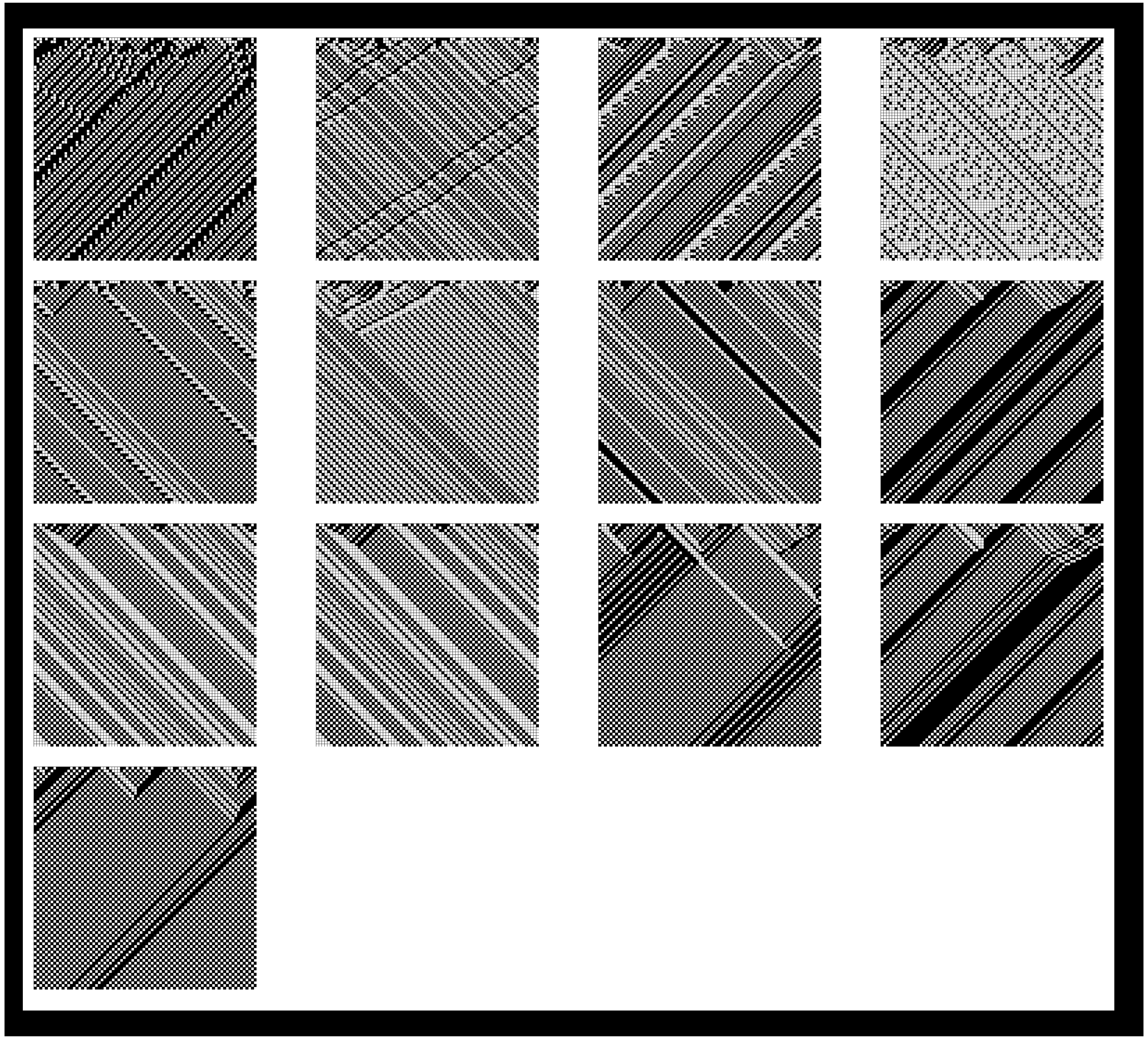
We pick t_{best} just as before.

For cells with differing states we **change the observation** following the outcomes from the rule … if the budget C allows. Each correction decreases the budget.

Obviously these changes are only done during the computation of fitness value and are **not** permanent.

Results?

- **Published**:
    - Incomplete observations <u>without</u> noise
    - Works very well - multiple experiments
    - Effort related to complexity properties
- **Unpublished** (yet):
    - Noisy case - works relatively well
    - Some specific CAs highly sensitive to noise (low noise - significant effort increase)

Open topics

Help needed

- Apply this to 2D and 3D CAs
- Multi-state (finite)
- ACCAs and other real-valued CAs — use Differential Evolution instead of GAs (Summer Solstice 2015)

- Accelerate fitness calculation with neural nets (or other estimation techniques)
- Replace GA with a purpose built neural net trained to identify CAs

Open questions

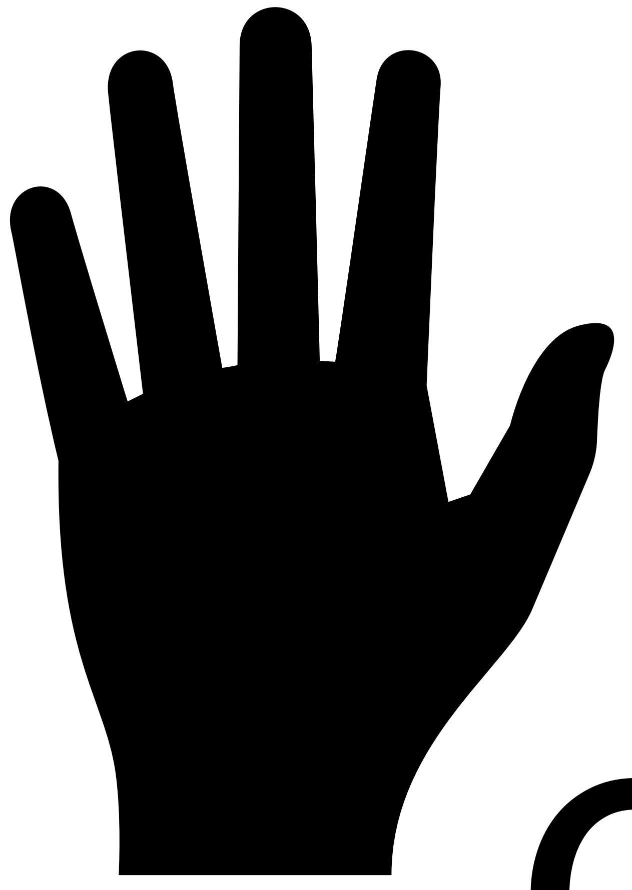
Open for cooperation

- Can we eliminate pre-setting correction budget **C** and evolve it during the GA run?
- Can we eliminate pre-setting time step limit **T**?
- Can we eliminate “golden initial condition” assumption?

- How to make this *useful* for real-world modeling or other applications?

Engineering learnings

- <u>**Use fitness value caching**</u>
- Use multi-threading (OpenMP etc)
    - Remember about thread safety (rand may be thread **unsafe**)
    - **Avoid** costly communication (no MPI needed)
- Pre-calculate memory - avoid dynamic allocations if you can
- Experiment with compiler / runtime choice & settings
- When running on your own hardware - keep it **cool** ❄❄❄



Questions?

Jit Team™

Thank you!

Witold.Bolt@jit.team