Cellular automata rules solving the wireless sensor network coverage problem

Rolf Hoffmann 1 · Dominique Désérable 2 · Franciszek Seredyński 3

Accepted: 26 April 2022 / Published online: 16 June 2022 © The Author(s) 2022

Abstract
The problem of an optimal coverage of a wireless sensor network area is considered. To solve this problem, a Cellular Automata (CA) approach is proposed. More specifically, the objective is to find CA rules which are able to cover the 2D space by a minimum number of so-called “Sensor Tiles”. A sensor tile consists of a von Neumann neighborhood of range 2 centered at sensor “point” and surrounded by 12 sensing “pixels”. Two probabilistic CA rules were designed that can perform this task. Results of an experimental study show that the first rule evolves very fast stable sub-optimal coverings, starting from a random configuration. The second rule finds optimal coverings, however it needs much more time for their evolution. The results are supported by a theoretical study on von Neumann neighborhoods and borrowing either from heuristics or from the spectral theory of circulant graphs.

Keywords Coverage problem · Wireless sensor network · Probabilistic cellular automata · Asynchronous updating · Matching templates · Von Neumann neighborhoods · Circulant graphs

1 Introduction
Wireless sensor networks (WSNs) are one of the main components of currently developing Internet of Things technologies. They are used in many sectors of human activities, such as industry, agriculture, transport, environment protection or logistics to monitor critical, very often remote and difficult to access areas, to discover events or collect data which are used to take decisions. A typical WSN consists of a number of sensors located at the monitored area, and each sensor is equipped with a single use battery. It is expected that some Quality of Service criterion, typically a predefined level of coverage of the monitored area should be fulfilled. The problem of coverage assumes answering to the question: when a battery of a given sensor should be turned on to make it active and able to monitor (cover) its part of the area, having in mind that it will result in spending battery energy, in the situation when perhaps other neighboring sensors are turned on and cover partially the same part of monitored area. This problem is closely related to the other issue, WSN lifetime maximization: keeping the coverage on the requested level should be achieved by a minimal number of active sensors in order to minimize the total energy consumption and prolong this way a lifetime of WSN. With low battery consumption, the lifetime of WSN can be maximized by switching between optimal configurations of active sensors. These issues are known to be NP-hard, and a number of centralized optimization algorithms assuming existence of full knowledge about the problem and an offline execution of an algorithm, and distributed and localized algorithms to find a solution has been proposed (Thai et al. 2008; Ab et al. 2009). In this paper we suggest a novel approach to solve the coverage problem by applying cellular automata (CA). This approach belongs to the class of distributed algorithms and is based on using concurrently
only local information, what gives a possibility to use it in real–time in online mode.

Our goal is to find a covering of the 2D space by so–called sensor tiles using CA. Our problem is one of the diverse covering problems (Snyder 2011) and it is related to the NP–complete vertex cover problem introduced by Hakimi (1965). A vertex cover is a set of nodes in a graph such that every edge of the graph has at least one end point in the set. A minimum cover is a vertex cover which has the smallest number of nodes for a given graph. Hakimi proposed a solution method based on Boolean functions, later integer linear programming (Gomes et al. 2006), branch–and–bound, genetic algorithm, and local search (Richter et al. 2007) were used, among others. Other related problems are the Location Set Covering Problem (Church and ReVelle 1976) and the Central Facilities Location Problem (Mehrez 1987). These problems aim to find the locations for P facilities that can be reached within a weighted distance from demand points, minimizing the number of P, or minimizing the average distance, or maximizing the coverage. For covering problems there are a lot of applications, in economy, urban planning, engineering, etc.

We assume that sensors are regularly located in an area to be covered, available at any discrete location of a superimposed grid. The question is how to turn them skillfully ON (active) or OFF (passive) to yield a sensor network with a minimum number of sensors, which we will call min point pattern. As shown in Fig. 1a, each active sensor (here also called point) senses a certain area in a circular range when battery is ON. Several sensors shall cover the whole space. A sensor with its range will be approximated by a discrete area (tile) (Fig. 1b).

Here, the idea is to treat the coverage problem as a pattern formation problem, where Parallel Substitution Algorithms (Achasova et al. 1994) served also as a source of inspiration. For the problem of forming a Domino Pattern we yielded already good results by using a probabilistic CA rule and overlapping tiles. In Hoffmann et al. (2021) the number of dominoes was maximized, whereas in Hoffmann et al. (2021) it was minimized. In Hoffmann (2022); Hoffmann and Seredyński (2020) the current problem of finding an optimal coverage in a WSN was already treated. We want here to follow this general approach of evolving patterns by applying a probabilistic cellular automata rule. Compared to Hoffmann and Seredyński (2020) additional material and a more effective First Rule is presented.

The following section provides the state–of–the–art on coverage issues in wireless sensor networks. In Sect. 3 the sensor tiling problem is described and optimal solutions are presented. In Sect. 4 the theoretical context in which this study takes place is described. In Sect. 5 two probabilistic CA rules are designed and their performance is evaluated in Sect. 6 before the conclusion where future explorations are proposed.

2 Related work

The coverage problem in WSNs and the lifetime maximization problem related to it are subjects of intensive studies, the results of which can be found in the current literature (for an extensive overview, see, e.g. Yetgin et al. (2017)). A number of centralized and distributed algorithms have been proposed recently. The covering problem is computationally expensive, and exact solutions can be found only for relatively small instances of the problem with use of classical optimization algorithms, such as e.g. linear programming (Cardei et al. 2005) or integer programming (Cheng et al. 2005). For realistic instance sizes of the problem one can rely on applying some heuristic (see, e.g. (Saadi et al. 2020)) or metaheuristic which can deliver approximate, near–optimal solutions.

Currently, one of the most popular approaches to solve the coverage problem is one based on applying Nature–inspired metaheuristics, which belong to the class of centralized algorithms and require full information about a state of the system. Different versions of evolutionary algorithms were applied, in particular memetic algorithms (Liao and Ting 2018) and genetic algorithms (Charr et al. 2019; Manju et al. 2018). Another popular technique is
particle swarm optimization. It was applied to solve the coverage problem considered either as a single objective problem (Jia et al. 2012; Jiao et al. 2019; Jawad et al. 2020) or multi objective problem (He et al. 2019). Two another optimization techniques like harmony search (Alia and Al-Ajouri 2017) and ant colonies were also recently applied (Rathee et al. 2021). A recent paper (Zhong et al. 2020) applies a novel approach called a metaheuristic, where with the use of the evolutionary technique of genetic programming, a high level heuristic to solve the problem is created on the base of a set of low–level heuristics.

During recent years, we can observe an increasing interest in designing distributed algorithms to solve the coverage / lifetime maximization problems in WSNs with use of learning automata (LA) or CA. These algorithms have a number of advantages in comparison with centralized algorithms. They consider a given problem as a distributed system consisting of a number of autonomous agents and focus on a description of local dependencies between agents and their actions to solve collectively a problem. Using this approach results in a more accurate description of a problem and may result in obtaining better results. What is also important, these algorithms do not require full information about a state of the system, can react on current changes of a system states, and therefore can be applied in real–time mode.

LA are a class of reinforcement learning machine algorithms and can be directly used to solve learning and optimization problems. They were applied to solve different variants of the coverage / lifetime maximization problems in WSNs, in particular by Mostafaei and Meybodi (2013); Razi et al. (2017) or Gaśior et al. (2018). Classical CA are not learning machines, and applying them to solve optimization problems requires some effort. The study presented in Tretyakova et al. (2016) is promising and shows that there exists a strong relationship between coverage levels and WSN lifetime, and specific CA rules describing the behavior of CA–based agents controlling the activation of node batteries. One possible way to provide CA with learning and optimization capabilities is to convert them into the Second Order CA, as it was proposed in Seredyński et al. (2021) and consider them as reinforcement learning machines operating in the Spatial Prisoner’s Dilemma environment or combining CA model with the spatial–temporal evolutionary process and learning automata theory like it was proposed in Lin et al. (2018). Another way is to find appropriate CA rules to solve the considered problem. Our recent work (Hoffmann et al. 2021) shows a potential of such an approach and in this paper we follow it. In this context, it is also worth to notice the work (Plènet et al. 2021) where CA–based approach to the observability problem was defined in the context of an autonomous network of mobile sensors considered as a control theory system, and oriented on an ability to reconstruct the initial system state.

3 Optimal covering with sensor tiles

3.1 The problem and its CA modeling

Given an array of \( N = (n \times n) \) cells, also called field. We assume that each cell contains a sensor which is either active or passive. The objective is to find a CA rule that can form a Sensor Coverage Pattern with a minimum number of active sensors that cover the whole area. An active sensor can cover (sense) a certain number of cells in its neighborhood. We can relate an active sensor with its sensed cells to a sensor tile as shown in Fig. 2a. A sensor tile is a discrete approximation of a real area sensed by an active sensor, as depicted in Fig. 1b. Note that sensor tiles are not automata cells, they are only used as a mean to find a cell rule and to define the covering of the space. We call the elements of a tile “pixels” in order to not confuse them with the cells of the space. A sensor tile consists of one center pixel (the kernel with the pixel value 1, in blue) and 12 surrounding pixels (the hull with value 0, in yellow). In short, a sensor tile is a certain von Neumann neighborhood of range 2.

Hull pixels of different tiles are allowed to overlap, but not with sensor points. The sensor points are said to have a mutual repulsive action.

We call the number of overlapping pixels at a certain site \((x, y)\) “overlap” or “cover level” \(v(x, y)\). Patterns with overlapping tiles are shown in Fig. 2b, c. The cover level is depicted here by numbers and colors. In the figures presented later, only numbers or colors will be used. Note that the cover level of an active sensor (in blue) is constant \(v = 1\).

The cell state is modeled as \(q = s\) for the First Rule (see Sect. 5.1) and as \(q = (s, h)\) for the Second Rule (Sect. 5.2). The state \(s \in \{0, 1\}\) models an inactive/active sensor, and all sensor states build the pattern (a sensor configuration). The hit number \(h \in \{0, 1, 2, 3, 4, [-1]\}\) stores the number of template hits (explained later in Sect. 5.2); the last symbol in brackets denotes the repulsive action of kernels. We assume cyclic border conditions in order to simplify the problem. Constant zero–boundaries of width 2 could also be used in order to keep the sensor points within certain borders. In the case of a fixed border, an appropriate number of hits has to be assumed for the border cells (e.g., \(h = 1\)), because the later described Second Rule reads hits from them.
3.2 Optimal solutions

We call a coverage valid, if the sensor tiles cover the whole space without gaps (uncovered cells). There are valid coverages/patterns with a different number of active sensors, between a minimal and a maximal number (as you can see later in Fig. 3). We call a valid coverage with a minimal number of active sensors min sensor pattern (for short min pattern), and a coverage with a maximal number max sensor pattern (for short max pattern). In this paper, we are interested in min patterns, but max patterns will also be considered. Note that there exist many equivalent sensor patterns taking into account the symmetries: translation, rotation, and reflection. When we speak of a pattern, we mean any representative in the class of equivalent patterns.

Using the CA rules described later, valid sensor patterns covering the whole space were found. They are listed in Table 1 for different field sizes. This table presents the number \( L \) of sensor tiles (equal to the number of points \( p \)), the maximum overlap \( v_{\text{max}} \) in the set of solutions, and the density \( R(N) = p/N \) of sensors (point density). E.g. for \( N = (7 \times 7) \), there are 5-tile patterns with (a) \( |v_{\text{max}} = 2| \geq 1 \) (several sites have overlap 2), (b) \( |v_{\text{max}} = 3| = 1 \) (only one site has overlap 3), (c) \( |v_{\text{max}} = 4| = 4 \) (four sites have overlap 4). The minimal point density for this example is \( R_{\text{min}}(49) = 5/49 = 0.102 \), and the maximal density is \( R_{\text{max}}(49) = 8/49 = 0.163 \). Recall that we search for min point patterns with a minimal point density.

Some min and max sensor patterns are shown in Fig. 3. The following min and max pattern were found (Table 1):

- \((3 \times 3)\) : There is only one solution.
- \((4 \times 4)\) : There are two solutions, each with two points. The maximal overlap level is 3 (appears twice) for the upper one \(|v_{\text{max}} = 3| = 2 \Leftrightarrow v_{\text{max}} = 3(2)\), and 4 for the lower one \(v_{\text{max}} = 4(2)\). There is no special min pattern.
- \((5 \times 5)\) : A min pattern with 3 and a max pattern with 5 points exist, but no pattern with 4 points. Note that there exists one cell with cover level of 3 in the min pattern, and there is no min pattern with \( v_{\text{max}} = 2 \) as we can find for \( n = 6-10 \).
- \(n = 6, 7, 8\) and \(9, 10, 11\) : There exist min–max patterns with 4–6, 5–8, 7–10 and 8–13, 8–20, 12–22 points.
- \((10 \times 10)\) : It was difficult to discover the min 8 point pattern shown in Fig. 3 (top right). You can observe 4 cells there with cover level 2. They define a square of size \((5 \times 5)\), and they are placed in the middle of a

![Fig. 2](image-url)  
(a) The “Sensor Tile” as von Neumann neighborhood of range 2.  
(b), (c) Some overlapping tiles with cover levels.

![Fig. 3](image-url)  
Minimal sensor patterns (upper half), maximal sensor patterns (lower half). A black square represents a sensor point, and the numbers give the overlap level. Field sizes range from \((3 \times 3)\) to \((10 \times 10)\)
horizontal or vertical pair of sensors. After extensive search, no valid pattern with 19 points could be found. It remains an open question if there exists such a pattern. The point number range is large (8–20), therefore this field size was used as a test scenario for the rules described in the following.
• \( (11 \times 11) \): There exists a regular min pattern (Fig. 4a) with 11 points that is difficult to evolve by the later described CA rules.

| Field Size \( N = n \times n \) | Tiles L Points P | Max. Overlap (number for different solutions) \( v_{\text{max}} \) | Point Density \( R = p/N \) | Comment |
|---|---|---|---|---|
| 3 x 3 | 1 | 2 | 1/9 = 0.111 | only one solution |
| 4 x 4 | 2 | 3(2), 4(2) | 1/8 = 0.125 | only two solutions |
| 5 x 5 | 3 | 3(1) 3(everywhere) | 3/25 = 0.12 1/5 = 0.2 | equidistant |
| 6 x 6 | 4 | 2(+), 3(2) 3(+) | 1/9 = 0.111 5/36 1/6 = 0.167 | |
| 7 x 7 | 5 | 2(+), 3(1), 4(4) 3(1 .. 5), 4(1) 2(everywhere) 3(+), 4(1 .. 2) 4(2 .. 5) | 5/49 = 0.102 6/49 1/7 1/7 8/49 = 0.163 | equidistant |
| 8 x 8 | 7 | 2(+), 3(2, 3, 10) 3(+), 4(1) 3(+), 4(1 .. 4) 3(+), 4(1 .. 4) | 7/64 = 0.109 1/8 9/64 5/32 = 0.156 | |
| 9 x 9 | 8 | 2(+), 3(1 .. 4), 4(1) 2(+), 3(1 .. 7) 3(+), 4(1 .. 2) 3(+), 4(1 .. 2) 3(+), 4(1 .. 4) 3(+), 4(1 .. 4) | 8/81 = 0.099 1/9 10/81 11/81 12/81 13/81 = 0.160 | |
| 10 x 10 | 8 | 2(4) 2(+) 3(1 .. 4) 3(1 .. 7) 3(+), 4(1) 3(+), 4(1 .. 3) 3(1), 4(1 .. 3) 3(+), 4(1 .. 4) 3(+), 4(1 .. 6) 3(+), 4(1, 2, 4, 8) 3(+) 3(everywhere) | 8/100 = 0.08 9/100 1/10 11/100 3/25 13/100 7/50 3/20 4/25 17/100 9/50 1/5 = 0.2 | equidistant |
| 11 x 11 | 11 | 2(22) 3(2 .. 6) 3(+) | 11/121 = 0.091 12/121 22/121 = 0.182 | |
(13 × 13) : There exists a regular min pattern (Fig. 4b) with 13 points with cover level 1 everywhere. So it can be named “absolute” min pattern because there exists no overlap at all. Obviously, multiples of size 13 yield such a pattern, too. The point density reaches the absolute minimum of $R_{\text{min}}^{13} = 1/13$. For comparison, the absolute maximal point density is $R_{\text{max}}^{13}(5, 10, \ldots) = 1/5$ for 5 × 5 max patterns with 5 points, and multiples thereof, e.g. for the 10 × 10 max pattern.

4 Theoretical background

4.1 Tiling the plane with von Neumann neighborhoods of range r

In 2d cellular automata, the von Neumann neighborhood of range r at point $(x_0, y_0)$ is the set of cell centers defined by

$$N_r'(x_0, y_0) = \{(x, y) : |x - x_0| + |y - y_0| \leq r \} \quad (r \in \mathbb{N})$$

at a Manhattan distance $d((x, y), (x_0, y_0)) \leq r$. The cardinality of this neighborhood, hereafter referred as $r$-neighborhood, is the centered square number

$$n_r = 2r^2 + 2r + 1 \quad (1)$$

illustrated in the inset of Fig. 5 for the first usual values of $r > 0$ (the 0-neighborhood is the trivial point $\{(x_0, y_0)\}$). Von Neumann’s 2d neighborhoods define a family of polyominoes, that are “prototiles” which perfectly tessellate\(^1\) the plane.

The labeling scheme is defined from a generating set $(s_1, s_2)(r)$ and must satisfy that $s_1, s_2$ and $n_r$ be pairwise coprime (Yebrá et al. 1985). A usual choice is to set $(s_1, s_2(r)) = (1, 2r + 1)$ : the first direction (horizontal) is generated by the infinite sequence $(0, 1, 2, \ldots, n_r - 1)$ of elements of $\mathbb{Z}/n_r\mathbb{Z}$ with 0 at the center of the prototile, whence the generation of the second sequence (vertical) is deduced. Thus, a cell $c$ is adjacent to cells $c \pm (2r + 1)(\text{mod} \ n_r)$ according to the first direction of generators and is adjacent to cells $c \pm (2r+1)(\text{mod} \ n_r)$ according to the second direction. This adjacency relation is formally depicted by the circulant graphs $C_{n_r}^{s_2(r)}$ – namely $C_5^{1,3}$ and $C_5^{1,5}$ – of Fig. 6. They are Cayley graphs and therefore have the property of being vertex–transitive (Boesch and Tindell 1984; Monakhova et al. 2020).

The adjacency matrices $M_{n_r}$ associated with the (von Neumann) $r$–circulant $C_{n_r}^{s_2(r)}$ have handsome spectral properties (Davis 1970). They are bisymmetric and the simple knowledge of coefficients $c_1, c_2, \ldots, c_{n_r-1}$ suffices to determine the set of eigenvalues. Thereby $c_1 = c_{2r+1} = 1$ and

$$\lambda_{n_r, k} = 2\cos\left(\frac{2k\pi}{n_r}\right) + 2\cos\left(\frac{(2r + 1)2k\pi}{n_r}\right) \quad (2)$$

where $\lambda_{n_r, k}$ denotes the $k$–th eigenvalue of matrix $M_{n_r}$. For a given $r > 0$ there exists a unique maximal eigenvalue $\lambda_{n_r, 0} = 4$ that denotes a 4–regular graph and there exist $(n_r - 1)/4$ equivalence classes with 4 equal eigenvalues and such that

$$\text{tr} (M_{n_r}) = \lambda_{n_r, 0} + \sum_{k=1}^{n_r-1} \lambda_{n_r, k} = 0$$

where $\text{tr}(M_{n_r})$ denotes the trace of $M_{n_r}$. Thereby from (2) and after simple trigonometric transformations we get

$$\lambda_{5,4} = \lambda_{5,3} = \lambda_{5,2} = \lambda_{5,1} \text{ for } M_5$$

\(^1\) In the algorithmic and practical context outside this section, the synonymy prototile ↔ “tile” and tile ↔ “pixel” will be used, without any confusion being to be feared.
λ_{13,12} = λ_{13,8} = λ_{13,5} = λ_{13,1} ; λ_{13,11} = λ_{13,10} = λ_{13,3} = λ_{13,2} ;
λ_{13,9} = λ_{13,7} = λ_{13,6} = λ_{13,4}

for \( M_{13} \). In particular, the spectrum highlights the rotational symmetry of the prototiles in Fig. 5 and by comparing with (Yebra et al. 1985), one could verify that the set of eigenvalues does not depend on the chosen labeling scheme.

The vertex-transitivity implies that any window of size \( n_r C_2 \) and embedded into this cellular space will contain exactly \( n_r \) distinct cells whatever the position of the window. Moving the window by a vector \( (\alpha, \beta) \) turns into the automorphism \( \tau_{ab} \) mapping \( \mathbb{Z}/n_r \mathbb{Z} \) into itself as
\[
c \rightarrow \tau_{ab}(c) = c + (\alpha s_1 + \beta s_2(r)) = c + (2r + 1)\beta(\mod n_r)
\]
for any \( c \in \mathbb{Z}/n_r \mathbb{Z} \). This therefore defines a \( n_r \times n_r \) wrap-around toroidal topology that provides the exact number of required sensor points. For any multiple \( n = p \cdot n_r \), this property remains true, that is, with \( p^2 n_r \) sensor points.

### 4.2 Min-coverage problem

For other sizes \( n \times n \) further analysis should be carried out. The perfect tiling will then give way to patterns with overlapped prototiles. Let \( V_n = (v_{ij})_{1 \leq i, j \leq n} \) therefore be the coverage matrix where element \( v_{ij} = v(x, y) \) denotes the cover level at site \((x, y)\). Let \( \sigma_n \) be the number of “0” in the \( n \times n \) field—the number of “sensor points”—for a slide...
given configuration. Then \( \sigma_n \) also denotes the number of prototiles and we consider the rational \( \frac{n^2}{n_2} \) surrounded by the two consecutive integers \( \sigma_n - 1 \) and \( \sigma_n \) such that

\[
\sigma_n - 1 < \frac{n^2}{n_2} \leq \sigma_n \quad \text{with} \quad n_2 \sigma_n = \sum_{ij} v_{ij} \tag{3}
\]

and where the sum in matrix \( V_n \) denotes the global cover index with \( \sigma_n \) points. It follows that \( \sigma_n \) defines a lower bound for the required number of points in the \( n \times n \) field. However, in most cases this rough lower bound is not a tight bound and the values of \( \sigma_n \) displayed in Table 2 are often lower than those provided by the simulation. This theoretical inaccuracy results from the fact that the toroidal constraint is not taking into account. A heuristic framework is presented, applied for any \( n \) (\( 3 \leq n < 13 \)) and illustrated in Appendix A1. The four following primitives are stated thereafter. The result is the formation of valid \textit{min} patterns for almost all values of \( n \).

The heuristic evolves more or less easily depending on whether or not there exists some “affinity” between \( n \) and \( n_2 \). The 4-fold rotational symmetry initiated at the beginning is conserved throughout evolution for \( n = 3, 7, 10 \) and is eventually achieved for \( n = 6 \). It can be checked that the relation \( n_2 \cdot \sigma_n^* \equiv_4 n^2 \) holds for those cases.

For \( n = 4 \) the case is somewhat trivial: at least 2 points are expected from (3), now a \( n \times n \) torus has diameter \( n \) and there exist \( \sigma_n^* = 2 \) antipodals at distance 4.

The symmetry is broken for \( n = 5, 8, 9 \) but an optimal pattern is easy to achieve.

For \( n = 12 \) the symmetry holds until the penultimate step, that is, until a single point is added. Indeed, we can observe that \( n_2 \cdot (\sigma_n^* - 1) \equiv_4 n^2 \) holds for this case.

\[
\text{Initialize} \quad (n, \sigma_n(0))
\]

\[
\begin{align*}
&\text{// Define a } n \times n \text{ window with } 4\text{-fold rotational symmetry:} \\
&\quad \text{// For } n \text{ odd the center is the } 0\text{-kernel of a centered prototile, surrounded by a maximum of prototiles whose kernels are strictly inner to the } n \times n \text{ window} \\
&\quad \text{// For } n \text{ even (} n > 4 \text{) the center is the } [2, 3, 10, 11] \text{ } 2 \times 2 \text{ square, surrounded by a maximum of prototiles whose kernels are strictly inner to the } n \times n \text{ window; for } n = 4 \text{ the pattern is empty} \\
&\quad \sigma_n^* = \sigma_n(0) \quad \text{// initial number of points}
\end{align*}
\]

\[
\text{Set_PBC} \quad (n, \{(i,j)\}, \{\text{pbc1}\})
\]

\[
\begin{align*}
&\text{// From the inner } (i, j) \text{ kernel, fix the periodic boundary conditions (PBC) generating the four } N-S \text{ and } E-W \text{ outer images } (i, j \pm n) \text{ and } (i \pm n, j) \text{ as well as the four } NW-SE-NE-SW \text{ outer images} \\
&\quad \text{// } \{(i, j)\} : \text{List of kernels} \\
&\quad \text{// pbc1: Number of points lost by PBC sequence} \\
&\quad \text{empty parameter is the default (pbc1 = 0)} \\
&\quad \sigma_n^* = \sigma_n^* - \text{pbc1}
\end{align*}
\]

\[
\text{Add_P} \quad (n, (i, j))
\]

\[
\begin{align*}
&\text{// Add a new prototile centered at } (i, j) \text{ and propagate PBC} \\
&\quad \sigma_n^* = \sigma_n^* + 1
\end{align*}
\]

\[
\text{Move_P} \quad (n, \{(i, j), \text{dir}\})
\]

\[
\begin{align*}
&\text{// Move the } (i, j) \text{ prototile to one of the } N-S-E-W \text{ direction } \text{dir} \text{ and propagate PBC} \\
&\quad \{(i, j), \text{dir}\} : \text{List of movements}
\end{align*}
\]

Surprisingly, this heuristic evolves badly for \( n = 11 \). Although it could lead to a pattern with \( \sigma_n^* = 13 \) points without too much difficulty (it can be checked that relation \( n_2 \cdot \sigma_n^* \equiv_4 n^2 \) holds), this pattern is far from optimal. The question arises whether this complexity could explain the excessive time required by the simulations for this particular value, as highlighted in the sequel, in Table 5.

Observing for \( n = 11 \) that \( n^2 = n(n_2 - 2) \) could suggest a possible tessellation of the \( n \times n \) field by some \( n \)-prototile. Again, the adjacency matrices \( M_{11} \) are bisymmetric and the simple knowledge of the two first coefficients is needed. An exhaustive examination shows that there exist exactly \( \sum_{k=1}^{4} k = 10 \) circulants

\[
C_{n,k}^s : \quad 1 \leq \mu < \frac{n - 1}{2} ; \quad \mu < \nu \leq \frac{n - 1}{2} \quad (n = 11)
\]

that split into two isospectral classes

\[
\begin{align*}
\mathcal{C}_1 &= \{ C_{11}^1, C_{11}^5, C_{11}^{24}, C_{11}^{34}, C_{11}^{35} \}; \\
\mathcal{C}_2 &= \{ C_{11}^3, C_{11}^{14}, C_{11}^{23}, C_{11}^{25}, C_{11}^{43} \}
\end{align*}
\]

and their circulant matrices are such that

\[
\lambda_{n,k} = 2 \cos\left(\frac{2 \mu k \pi}{n}\right) + 2 \cos\left(\frac{2 \nu k \pi}{n}\right) \quad (0 \leq k \leq n - 1) \quad (n = 11)
\]

where \( \lambda_{n,k} \) denotes the \( k \)-th eigenvalue of matrix \( M_{n}^{s,v} \). Dividing the circulants into the two classes \( \mathcal{C}_1, \mathcal{C}_2 \) results from simple trigonometric transformations. Each class \( \mathcal{C}_1 \) (resp. \( \mathcal{C}_2 \)) has its own minimum (negative) eigenvalue \( \lambda_1 \) (resp. \( \lambda_2 \)) and the criterion \( \lambda_2 < \lambda_1 \) yields the \textit{min} valid patterns in \( \mathcal{C}_2 \). The whole set of circulant \( \mathcal{C}_{11} \) with their respective pattern is displayed in Appendix A2 (Figs 24, 25).
Resulting either from heuristic or from spectral theory, a set of valid patterns \( \sigma^*_n \) resulting either from heuristic or from spectral theory, is displayed in Fig. 7. Each cell is “decorated” from its von Neumann label of Fig. 5. The number of labels at site \((x, y)\) is the cover level \(v_{ij} = v(x, y)\) with one color per eigenvalue in matrix \(M_{n,2}\).

The patterns are consistent with the results of the simulations in Figs. 3–4. They perfectly match for \(n = 3\), \(n = 6\), \(n = 10\) and now \(n = 11\). The number \(\sigma^*_n\) of points appears in the last row of Table 2.

### 4.3 Equivalence between Min and Max coverage problems

We close this theoretical framework by pointing out a relation between our problems of minimization and maximization. Basically, our min problem could be associated with a physical distancing configuration while our max problem could be associated with a tightly-coupled configuration and the objective function thus becomes the

| \(n\) | \(3\) | \(4\) | \(5\) | \(6\) | \(7\) | \(8\) | \(9\) | \(10\) | \(11\) | \(12\) | \(13\) |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| \(n^2/n_2\) | 0.69 | 1.23 | 1.92 | 2.77 | 3.77 | 4.92 | 6.23 | 7.69 | 9.31 | 11.08 | 13.00 |
| \(\sigma_n\) | 1 | 2 | 2 | 3 | 4 | 5 | 7 | 8 | 10 | 12 | 13 |
| \(\sigma_n(0)\) | 1 | 0 | 1 | 4 | 5 | 4 | 5 | 8 | 9 | 12 | 13 |
| \(\sum v_{ij}\) | 13 | 26 | 39 | 52 | 65 | 91 | 104 | 104 | 143 | 169 | 169 |
| \(\sigma^*_n\) | 1 | 2 | 3 | 4 | 5 | 7 | 8 | 8 | 11 | 13 | 13 |
search for the optimal number \( \sigma_n \) of points in both cases. For simplicity we consider the infinite plane and assert thereby that the pattern must be biperiodic. Obviously for the max problem, trying a 4-fold coverage of the whole hull will prove to be impossible due to the repulsive action of the kernel. Therefore the maximal suitable solution is a 3-fold coverage of the whole hull as suggested in Fig. 3 and illustrated on the decorated pattern of Fig. 8. It follows that solving the min–problem in the 1-neighborhood amounts to solve the max–problem in the 2-neighborhood.

By extension, we conjecture that there would exist a close relationship between solving the min–problem in a \( r \)-neighborhood and solving the max–problem in the \((r + 1)-\)neighborhood, shortly: \( \min_r \approx \max_{r+1} \) with of course an additional condition for a finite field.

We refer back to inequalities in (3) and assume that equality holds for \( n_1 = n_2 \). On the other hand, let

\[
\sigma_n = \frac{n^2}{n_1} = \frac{\sigma_n + 3(n^2 - \sigma_n)}{n_2} \quad (n_1 = 5, n_2 = 13)
\]

(4)

that means (i) for the min–problem, all \( n \times n \) sites take on value “1” regardless of their state either of kernel or of hull (ii) for the max–problem, all \( \sigma_n \) kernel points take on value “1” while the remaining \((n^2 - \sigma_n)\) hull sites take on value “3”. It follows that

\[
n_2 \cdot n^2 = n_1 (3n^2 - 2\sigma_n) \Rightarrow n^2 (3n_1 - n_2) = 2\sigma_n \cdot n_1 \Rightarrow \sigma_n \cdot n_1 = n_2^2
\]

whence the equivalence \( \min_1 \approx \max_2 \).

5 The designed CA rules

First, as a matter of principle, we have to decide (i) whether to use a synchronous or an asynchronous updating scheme, and (ii) whether to use a deterministic or a probabilistic rule. This makes four options: (1) synchronous updating & deterministic rule, (2) synchronous updating & probabilistic rule, (3) asynchronous updating & deterministic rule, and (4) asynchronous updating & probabilistic rule.

We have to keep in mind that we search for a CA rule that converges always or with a high probability to optimal or near-optimal patterns. From our previous work we have learned that it is very difficult or even impossible to design such a rule with Option 1 because we may have to avoid or dissolve conflicts, deadlocks, live-locks, and emerging oscillating, moving or clustering structures, as we know, e.g. from the Game of Life, in order to drive the pattern continuously to an optimum (not to get stuck in suboptimal solution areas).

The remaining options (2–4) are related because the computation of a new configuration is stochastic. It seems that they can be transformed into each other to a certain extent. Here we want to use Option 4 because we have gained good results in solving another problem (Hoffmann et al. 2019) in this way. Moreover, we don’t need a clock for synchronization and buffering for the configuration, which is closer to the modeling of natural processes. In contrast to that former solved problem, we address here another difficult problem where the number of tiles is minimized and not maximized.
5.1 The first rule

The idea is to modify the current configuration systematically such that valid patterns appear and at last a min pattern. To do this, the CA configuration is searched for tile parts (specific local patterns) and if an almost correct tile part is found, it is corrected, otherwise some random noise is injected.

The tile parts are called templates $A_i$. They are systematically derived from the sensor tile (Fig. 9a). For each of the 13 tile pixels (so-called derivation pixels, marked in red) a template is defined by shifting the tile in a way that the derivation pixel appears in the center. Note that many of these templates are similar under various symmetries: $A_3, A_4, A_5$ are rotations of $A_2$; $A_7, A_8, A_9$ are rotations of $A_6$ and $A_{11}, A_{12}, A_{13}$ are rotations of $A_{10}$.

We represent a template $A_i$ as an array of size $(k/k)$ of pixels, where $k = 2m - 1$ and $(m \times m)$ is the size of the tile, enlarged to a square box embedding it. Our tile is of size $(5 \times 5)$ including empty pixels, and the templates are larger because of shifting, maximal of size $(9 \times 9)$. The pixels within a template are identified by relative coordinates $(\Delta x, \Delta y)$. The center pixel at $(\Delta x, \Delta y) = (0, 0)$ is called “reference pixel”. Each template pixel carries a value $\text{val}(A_i, \Delta x, \Delta y)$ with $\text{val}(A_i, 0, 0) \in \{0, 1, #\}$. The value of the reference pixel is called “reference value”, $\text{refval}(A_i) = \text{val}(A_i, 0, 0) \in \{0, 1\}$, which is equal to the value of the derivation pixel. The symbol # represents “Don’t Care”, meaning that a pixel with such a value is not used for matching (or does not exist (empty pixel) in another interpretation). Pixels with a value 0 or 1 are valid pixels, their values are equal to the values derived from the original tile. Some templates can be embedded into arrays smaller than $(k \times k)$ when they have Don’t Care symbols at their borders.

We need also to define the term “neighborhood template” that is later used in the matching procedure. The neighborhood template $A_i^*$ is the template $A_i$ in which the reference value is set to #, in order to exclude the reference pixel from the matching process. The cell processing scheme is:

- At time-step $t$ a new configuration is formed by updating $N$ cells in a random order. For each time-step a new random permutation is used. The new configuration is complete after $N$ cell updates (each cell is updated once during this period) and it defines the next configuration at time-step $t + 1$.
- The rule is applied asynchronously. The new cell state $s' = f(s, B')$ is computed and immediately updated without buffering. $B'$ denotes the states of the neighbors within a local window, where the center cell $s(x, y)$ is excluded for matching.

The First Rule is the following:

Fig. 9 a The 13 templates $A_i$ of the sensor tile. The value $\text{refval}(A_i)$ of the reference pixel (marked in red) is used for cell updating in the case that all remaining template pixels (the neighborhood template) match with the corresponding cells of the current configuration. b The neighborhood templates $A_i^*$. The dotted box marks the $(5 \times 5)$ window used for matching. The remaining neighborhood templates result from rotation of the shown ones. c $A_{10}^*$ represented as an array reduced from $(9 \times 9)$ to $(5 \times 5)$.
The neighborhood templates $A_i^s$ are tested against the corresponding CA cell neighbors $B^s(x, y)$ in the current $(5 \times 5)$–window at position $(x, y)$. Thereby the marked reference position $(\Delta x, \Delta y) = (0, 0)$ of a neighborhood template is aligned with the center of the window. Note that we use for testing a window of size $(5 \times 5)$ which is smaller than the full size $(9 \times 9)$ of the neighborhood templates. Therefore, some valid pixels outside the $(5 \times 5)$–window are not tested (e.g. the bottom 4 yellow pixels of $A_{10}$ in Fig. 9b). The implementation with these incomplete neighborhood templates worked very well, but further investigations are necessary for proving to which extent they can be incomplete.

If all values of a neighborhood template $A_i^s$ match with $B^s(x, y)$ then we register a hit that is stored only temporarily. There can be several hits equal to the number of matching templates. The number of hits approximates the cover level and is equal to it when the pattern becomes stable. If we have at least one hit (Rule part (b)), then the sensor state of the current cell $s(x, y)$ is set to the reference value $refval(A_i)$ and then we create or validate a correct tile part in the current $(5 \times 5)$–window. Otherwise we could not find or adjust a correct tile and the local pattern is noisy. Then we inject additional noise (Rule parts (c),d) at the current cell position $(x, y)$ in expectation to form a valid tile.

If the current state is 0 we change it to 1 with probability $\pi_0$, and if it is 1 we change it to 0 with probability $1 - \pi_0$. The idea behind is to inject “asymmetric” noise in order to force the evolution into more white or more black cells. Using a low probability $\pi_0$ means that white cells mainly stay white, whereas black cells (sensor points) are mainly forced to get white in order to disappear. Note that the cell’s state may remain unchanged as default (Rule part (a)) if none of the conditions in (b), (c), (d) triggers an update. The lowest point density have min pattern of field size $(13 \times 13)$ or multiples thereof, with a point density of $1/13 = 0.077$. So we may choose this density for $\pi_0$ in order to drive (hopefully) the evolution to min patterns.

There can be no conflicts, because the reference value is the same (uniquely derived from the tile) if there are several hits. (Examples: If $A_1^s$ matches, there is one hit only, and the reference value is 1. If $A_{10}^s, A_{11}^s, A_{12}^s, A_{13}^s$ match, we get 4 hits with reference values 0). As no conflicts can arise, the sequence of testing the templates does not matter, and one could skip further tests after a first hit.

It is important to note that this rule obeys the criterion of stability, which means that a valid pattern without gaps (uncovered cells) is stable because we have matching hits at every site. Otherwise, some asymmetric random noise is injected in order to drive the evolution to the aimed pattern.

### 5.1.1 Testing the first rule

10,000 runs were performed on $(10 \times 10)$–fields with random initial configurations and a time–limit of 100 iterations, with different probabilities $\pi_0 = 0.5, 0.2, 0.09, 0.077, 0.02, 0.01$. The CA system converges quickly to a stable sub-optimal sensor pattern after $t_{\text{average}} = 16.72 - 1.99$ time–steps on average (Table 3). We observe in Fig. 10 how fast a stable pattern with 15 points can evolve. The evolution is faster for lower $\pi_0$.

All patterns cover the space as required. Most often the patterns contained 14 points. The average number of points $(14.96 \rightarrow 13.92)$ decreases with the probability $\pi_0$. The reason is that the probability $1 - \pi_0$ for injecting zeros is then higher (favoring low cover levels) than the probability $\pi_0$ for injecting ones (favoring points). No min pattern with 8 points and even no pattern with 9 points was found during 10,000 runs. A few near–min patterns with 10 points evolved for $\pi_0 = 0.09 - 0.01$. Max patterns were found only for high probabilities $\pi_0 = 0.5, 0.2$. We can conclude that min sensor patterns are very rare in the whole set of all valid patterns covering the space. We have chosen the probability $\pi_0 = 0.01$ for the following work because the average number of points is lower than for $\pi_0 = 1/13$, the value that we first expected to give the best results.

We can conclude that we have found a CA rule that can evolve valid sensor point patterns, but unfortunately the number of active sensors is not necessarily minimal. What is the reason? The rule evolves patterns that fulfill one of the two conditions for each cell at $(x, y)$:

1. $s(x, y) = 1 \wedge \forall (\Delta x, \Delta y) : s(x + \Delta x, y + \Delta y) = 0$, where $(\Delta x, \Delta y) \in$ set of relative coordinates of the hull pixels, and where the point tile’s center $(\Delta x, \Delta y) = (0, 0)$ is excluded. This condition means that a sensor point finds only zero–state cells in its beams.

2. $s(x, y) = 0 \wedge \exists (\Delta x, \Delta y) : s(x + \Delta x, y + \Delta y) = 1$, where $(\Delta x, \Delta y) \in$ set of relative pixel coordinates of any template (centered at $(\Delta x, \Delta y) = (0, 0)$), and where the templates’ centers $(\Delta x = 0, \Delta y = 0)$ are excluded. Don’t Care pixels shall not contribute to this set. This condition means that each zero–state cell is at least covered by one sensor point’s beam, or in other words a zero–state cell is connected to a sensor that senses it.
These two conditions are only necessary conditions for valid coverages, but not sufficient to define min patterns. It seems to be quite difficult to find local logical conditions that ensure a global minimum of points, except for special cases like \( n = 13 \) where there is only one optimal solution with cover level \( v = 1 \) everywhere (Fig. 4b). For that special case the second condition can be defined more strictly ("... exists exactly one...") as

\[ s(x, y) = 0 \land \exists!(Dx, Dy) : s(x + Dx, y + Dy) = 1 \]

Now we need to improve our CA rule in order to evolve min patterns with a high probability.

### 5.2 The second, improved rule

The purpose of this enhancement is to improve the rule in such a way that the number of points reaches a minimum. Whereas the first rule works with the state \( q = s \) only, now the state is extended by the number of hits \( h \), thus the full state \( q = (s, h) \) is used. Now all neighborhood templates are tested and all hits are stored for every site \((x, y)\). The number of hits \( h(x, y) \) is:

- \( 0 \) : no neighborhood template matches or there is a gap.
- \( 1 \) : one neighborhood template matches where the reference value is zero (yellow colored).
- \( 2–4 \) : \( h \) neighborhood templates match with reference values zero, that means that 2–4 tiles (yellow hull pixels) are overlapping.
- \([-1]\) : the neighborhood template \( A_f^1 \) matches where the reference value is 1 (blue). Recall that blue pixels are not allowed to overlap. The symbol in brackets denotes the repulsive action of kernels.

The hit number \( h(x, y) \) holds the actual value after matching with all the neighborhood templates. Because of the random sequential updating scheme, the \( h \)-values in the \((x, y)\)-neighborhood may not be up-to-date and can carry old values from the former configuration at time-step \( t-1 \). Nevertheless, the \( h \)-values correspond mainly to the cover levels \( v \) especially when the pattern becomes more stable. This inaccuracy introduces some additional small
noise which can even speed-up the evolution. And when
the pattern becomes stable, the hit number equals the cover
level: \( \nu(x, y) : h(x, y) = \nu(x, y) \).

The idea is to minimize the overlap between tiles by
destroying cell states with high overlap level \( (h > 1) \)
through noise, allowing reordering with a lower number of
points. In order to find a rule, we need to study the min
point patterns with respect to their overlap values and local
situations. From Table 1 and Fig. 3 we can see that min
patterns contain some cells with a \( v_{\text{max}} \) overlap \( h'_{\text{max}} = 2, 3 \).
(There is a special case with \( n = 13 \) or multiples of 13
where there exists a pattern with \( v_{\text{max}} = 1 \) that we will not
be taken into consideration here.)

First the new state \( s'(x, y) \) is computed according to
the First Rule, and additionally, the number of all hits \( h(x, y) \)
is computed and stored. Then the new state is modified to
\( s''(x, y) \):

\[
s''(x, y) = \begin{cases} 
  s'(x, y) & \text{default} \\
  \text{random} \in \{0, 1\} & \text{with probability } \pi_4 \quad \text{if } h(x, y) = 4 \\
  \text{random} \in \{0, 1\} & \text{with probability } \pi_3 \quad \text{if } C_1 \text{ or } C_2 \text{ or } C_3 
\end{cases}
\]

where \( C_1 = (\text{hits}3\times3(x, y) > 14), \)
\( C_2 = (\text{hits}3\times3(x, y) > 13) \) and \( (\text{Active} 3 \times 3, (x, y) > 0), \)
\( C_3 = (\text{hits}3\times3(x, y) = 12) \) and \( (\text{Active} 3 \times \)
\( 3 (x, y) = 0) \) and \( (h(x, y) = 3). \)

The conditions \( C_{1,3} \) add additional noise in order to
drive the evolution to the optimum when the local hit
density is above a certain level. It was quite difficult to find
these conditions through many trial and error simulations
taking into account the local patterns in \( (3 \times 3) \)-windows
of valid optimal and near-optimal solutions. It would be
interesting to find better conditions through further
research. The ultimate goal is to find a rule that always
drives to a stable optimal solution, not excluding any
solution from the set of all possible solutions.

The function \( \text{hits}3\times3(x, y) \) computes the sum of the hits
of inactive cells in a local \( (3 \times 3) \)-window with its center
at \( (x, y) \), where active sensor cells and the center are
discarded. The function \( \text{Active}3\times3(x, y) \) computes the sum of
active cells in a \( (3 \times 3) \)-window.

Now, for this improved rule, it is not clear whether the
stability criterion is still fulfilled because of the additional
noise. In fact, it turned out that reached min patterns are
often stable, although some non-min patterns can be stable,
too. Extensive simulations showed that noise injection
under these additional conditions drive non-min patterns to
min patterns. Unfortunately, at the moment, we cannot
show that the evolution always ends up with a stable min
pattern, because (a) we cannot prove that all reached valid
non-min patterns are transient (meaning that then further
noise will still be injected), and (b) that all reached min
patterns are stable (meaning that then noise injection is
always stopped).

A deeper analysis is a subject to further research. It
remains an open question, whether a local CA rule can be
found that always drives the evolution to a min point pat-
tern, and preferably to any of all possible min patterns, not
excluding solutions with a certain max cover level or cer-
tain local sensor arrangements.

During a simulation, the number of complete tiles \( L \)
points \( L \) is increasing, decreasing and fluctuating, and at the
end the evolution often is driving towards a valid stable pattern, which often is a min pattern. Many ex-
periments showed that optimal min patterns can successfully be
found with the Second Rule if (a) the maximal number of
time-steps \( T_{\text{Limit}} \) is chosen large enough and \( L \) or (b) sev-
eral runs with random initial states are performed.

6 Simulation and performance evaluation

6.1 Performance for field size \( (10 \times 10) \)

The improved rule was tested 10 000 times on \( (10 \times 10) \)-
fields with random initial states \( (s \in [0, 1], \) for \( T_{\text{Limit}} = 1 \)
000 time-steps, with \( \pi_4 = 0.1, \pi_3 = 0.9, \) and \( \pi_0 = 0.01 \)
yielding best results). For each run, several parameters
were recorded, such as the time-stamp for reaching the
greatest or smallest number of points in valid patterns. The
pattern frequency (the number of evolved patterns with a
certain number of points) is given in Table 4. The average
number of points \( p_{\text{average}} \) and the average number of time-
steps \( t_{\text{average}} \) needed are also presented. In order to evolve
patterns with a few points, the probability \( \pi_0 \) should be
kept low.

Now we were able to evolve min patterns with 8 points
for a high \( T_{\text{Limit}} \). Most of the evolved patterns are close to
the optimum, and lie between 9 and 12 for a small com-
putational budget of \( T_{\text{Limit}} = 800 \). For a high budget of
\( T_{\text{Limit}} = 102 400 \) the average number of points is only
\( p_{\text{average}} = 9.58 \) with \( t_{\text{average}} = 11 \) 237. Compared to the
First Rule, the Second Rule (together with the First Rule)
performs significantly better and the probability is high to
reach an optimal min pattern. Fig. 11 shows the evolution
of a stable min pattern with 8 sensors. During the evolution,
other valid transient patterns with a different number of
complete tiles \( L \) appear. The transient patterns between the
shown time-steps are not valid, they usually show some
tiles but are partially noisy.
6.2 Performance for other field sizes

The Improved Rule was also tested on other field sizes and for 1000 or 100 runs. During each run, the lowest number of points reached was stored and used for the statistics. With an increasing budget of $T_{\text{limit}}$, the probability to find $\min$ patterns increases.

In order to assess the time complexity, we define the \textit{computing effort per cell} to evolve $\delta \% \times R \ \min$ patterns during $R$ runs within time $t \leq T_{\%}(N)$ as $E(T_{\%}, N) = (T_{\%}(N))/N$, where the maximal needed time $T_{\%}(N)$ was extracted from the simulation data. If $E(N) = \text{const}$, then the needed time would be in $O(N)$ to reach $\delta \%$ min patterns on average of $R$ runs. We have chosen $\delta = 3$ because this was the lowest rate of found $\min$ patterns, for $n = 10$ and $n = 11$ (Table 5). In our experiments, this effort increases exponential with $N$ as shown in Fig. 12. Therefore, it is costly to compute optimal solutions for large $N$. But as the CA model is inherently parallel regarding $N$, we can reduce the computation time significantly on a parallel computer. For large $N$ the algorithm is still applicable, though we need to terminate it due to a restricted computing budget when having found a near-optimal solution. In order to reduce the computational effort in principle, one could try to find a more sophisticated rule or to follow a divide-and-conquer approach.

### Table 4

The improved rule was tested on (10 x 10)-fields for different time limits $T_{\text{limit}}$ and for 1000 or 100 runs. During each run, the lowest number of points reached was stored and used for the statistics. With an increasing budget of $T_{\text{limit}}$, the probability to find $\min$ patterns increases.

| Points | Pattern frequency [1/100] |
|--------|--------------------------|
| $T_{\text{Limit}}$ | 800 | 1600 | 3200 | 6400 | 12800 | 25600 | 51200 | 102400 |
| # of runs | 1000 | 1000 | 1000 | 1000 | 1000 | 1000 | 1000 | 1000 |
| 8 | 0.1 | 0.1 | 0.3 | 0 | 1 | 1 | 2 | 3 |
| 9 | 1.4 | 2.5 | 5.8 | 10.6 | 21 | 30 | 39 | 36 |
| 10 | 11.2 | 19.2 | 35.5 | 50.6 | 73 | 64 | 57 | 61 |
| 11 | 62.4 | 72.0 | 57.8 | 38.8 | 5 | 5 | 2 | 0 |
| 12 | 25.0 | 6.2 | 0.6 | 0 | 0 | 0 | 0 | 0 |
| Average # of runs | 10,11 | 10,81 | 10,52 | 10,28 | 9,82 | 9,73 | 9,59 | 9,58 |
| Average # of points | 255 | 531 | 942 | 1,972 | 4,521 | 6,756 | 7,898 | 11,237 |

**Fig. 11** Snapshots of an evolution yielding a (10 x 10) $\min$ pattern with 8 points. Only valid patterns with $L \geq 8$ points are shown. The final pattern is stable for $t \geq 179$.
7 Conclusion

In this paper, the problem of an optimal coverage of a wireless sensor network area was considered and solved by means of a probabilistic Cellular Automata (CA). Two CA rules were designed, that can find non–optimal and optimal min sensor patterns. The first rule evolves very fast stable valid patterns, with a peak number of points lying between minimum and maximum. The design principle behind is methodical and based on a set of templates derived from all pixels of the sensor tile. The second rule was designed especially to find min patterns, and it can do so, although the time to evolve an optimal min pattern can exceed the available processing capabilities. Moreover, regarding the r–von Neumann neighborhoods that serve as templates, it has been shown that there is a close relationship between min and max problems, depending only on their objective function. In addition, it has already been shown elsewhere (Hoffmann et al. 2021) that the core of the CA transition rule changes only slightly whatever it is a min problem or a max problem.

Regarding the required minimal number of sensor points, the results of the simulation (in Table 4) have been supported by a theoretical study (in Table 1) on von Neumann neighborhoods and borrowing either from heuristics (for almost all values of n) or from the spectral theory of circulant graphs (for n = 11). For this particular size, the question arises whether this complexity could explain the excessive time required by the simulations (in Table 4) or the atypical effort highlighted in Fig. 12.

The “artificial” intelligence of this model, based on the power of its template–based system, therefore has its

| Table 5 | Simulation for different time limits and number of runs. Percentage of found optimal min patterns, and time–steps needed (taverage, tmin, tmax) |
|---------|-----------------------------------------------------------------------------------------------------------------------------------------|
| N = n x n | Min points | Runs | T Limit | Found min patterns [%] | taverage of found min pat. | tmin | tmax | T₃/3% found | E = T₃% /N |
| 3 x 3   | 1          | 1000 | 8      | 100                  | 0.97                        | 0    | 1    | 0.1        | 0.011     |
| 4 x 4   | 2          | 1000 | 10     | 100                  | 1.22                        | 1    | 2    | 1          | 0.063     |
| 5 x 5   | 3          | 1000 | 20     | 100                  | 3.11                        | 1    | 13   | 1          | 0.040     |
| 6 x 6   | 4          | 1000 | 120    | 100                  | 12.4                        | 1    | 89   | 1          | 0.028     |
| 7 x 7   | 5          | 1000 | 1000   | 100                  | 111                         | 1    | 607  | 1.53       | 0.031     |
| 8 x 8   | 7          | 1000 | 3000   | 100                  | 407                         | 1    | 2851 | 4.53       | 0.071     |
| 9 x 9   | 8          | 100  | 50000  | 76                   | 2226                        | 12   | 15048| 12.5       | 0.154     |
| 10 x 10 | 8          | 100  | 100000 | 3                    | 3741                        | 944  | 9210 | 37.41      | 37.41     |
| 11 x 11 | 11         | 100  | 1000000| 3                    | 419961                      | 281031|681050|16803       | 3471      |
| 12 x 12 | 13         | 100  | 300000 | 5                    | 163098                      | 96764|291384|93584       | 650       |

Most significant values are in bold

Fig. 12 The effort (number of timesteps per cell) vs. the number of cells (size of the field). It is very low for N ≤ 81. The trend line shows an exponential growth. The peak for N = 121 means that the related regular absolute min pattern is one which is difficult to evolve.
counterpart, namely its limitation in processing power to handle large-scale computing fields, a very time-consuming process. As already mentioned in other words, this weakness therefore makes it a good candidate for implementation on parallel processing environments.

In further work the possible sensor locations could be restricted, the charge of batteries could be taken into account, or this approach could be related to the vertex cover problem in order to compare time complexity.

Finally, as introduced through the first figure of this paper, in order to approximate the circular sensing area better, a hexagonal lattice could be more favorable. Two families would then be under study: either a still circulant topology based on Eisenstein–Jacobi networks (Huber 1994; Martínez et al. 2008) or a fractal topology based on the figure of Sierpiński arrowhead (Sierpiński 1916; Désérable 1999).

**Appendix A1 – Heuristic construction of valid min $n \times n$ patterns**

A sample of patterns for $n \times n$ fields ( $3 \leq n < 13$ ) is constructed from heuristics. The stepwise construction is illustrated with the primitives in Sect. 4.2. All patterns are valid min patterns except for $n = 11$. For clarity’s sake, the field is surrounded by a $(n + 2) \times (n + 2)$ bounding box. The Initialize primitive, starting from a centered von Neumann neighborhood as initial configuration, has a very important influence to quickly reach an optimal pattern. When defining boundary conditions (Set_PBC), the interior reference point (circled in red) implies four N–S–E–W and four NW–SE–NE–SW image points (circled in black). It may happen that the images are outside the bounding box; however, their impact is still visible on the boundaries. The same happens when adding (Add_P) or moving (Move_P) a point since the boundary conditions follow (Figs. 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23).

---

**Fig. 13** Formation of a valid balanced pattern for $n = 3$. The 2-step sequence gives: Initialize (3, 1) $\rightarrow$ Set_PBC (3, (2, 2) ). The sequence yields $\sigma_I^2 = 1$.
Fig. 14 Formation of a valid balanced pattern for $n = 4$. The 3–step sequence gives:

- **Initialize** $(4, 0) \rightarrow$
- **Add P** $(4, (1, 1)) \rightarrow$
- **Add P** $(4, (3, 3))$. The sequence yields $\sigma_4 = 2$
Fig. 15 Formation of a valid pattern for $n = 5$. The 4-step sequence gives: Initialize $(5, 1) \rightarrow \text{Set}_{\text{PBC}} (5, (3, 3)) \rightarrow \text{Add}_P (5, (1, 1)) \rightarrow \text{Add}_P (5, (4, 5))$. The sequence yields $\sigma^2_n = 3$. 
Fig. 16 Formation of a valid balanced pattern for \( n = 6 \). The 4-step sequence gives:

- **Initialize** \((6, 4)\) →
- **Set_PBC** \((6, (1, 4), (3, 1), 2)\) → **Add_P** \((6, (6, 1))\) → **Add_P** \((6, (4, 4))\).

The sequence yields \( \sigma_6^r = 4 \). Two points are lost during PBC sequence but two new points are injected afterwards.
Fig. 17 Formation of a valid balanced pattern for \( n = 7 \). The 3–step sequence gives: Initialize \((7, 5)\) → Set_PBC \((7, (1, 2), (2, 7), (7, 6), (6, 1))\) → Move_P \((7, ((1, 2), N), ((2, 7), E), ((7, 6), S), ((6, 1), W))\). The sequence yields \( \sigma_7 = 5 \).
Fig. 18 Moving effect on the 7–pattern from Step 2 to Step 3: the 4–cluster of points emerging at the junction of 4 adjacent 7–patterns is released.
Fig. 19 Formation of a valid pattern for $n = 8$. The 6-step sequence gives: Initialize $(8, 4) \rightarrow \text{Set}_PBC(8, (4, 2), (2, 5), (5, 7), (7, 4)) \rightarrow \text{Add}_P(8, (1, 1)) \rightarrow \text{Add}_P(8, (8, 7)) \rightarrow \text{Move}_P(8, ((5, 7), N)) \rightarrow \text{Add}_P(8, (4, 6))$. The sequence yields $\sigma^n = 7$. The moving fills the two isolated gaps, even if it means producing a cluster of gaps.
Fig. 20 Formation of a valid pattern for \( n = 9 \). The 4-step sequence gives: Initialize \((9, 5)\) \(\rightarrow\) Set_PBC \((9, (2, 3), (3, 8), (8, 7), (7, 2))\) \(\rightarrow\) Add_P \((9, (1, 9))\), Add_P \((9, (5, 5))\), Add_P \((9, (5, 1))\) \(\rightarrow\) Move_P \((9, ((3, 8), S), ((8, 7), W))\). The sequence yields \(\alpha_7^9 = 8\). Moving fills the gaps.
Formation of a valid balanced pattern for $n = 10$. The 5-step sequence gives:

1. Initialize $(10, 8)$
2. Set PBC $(10, (2, 1), (1, 9), (9, 10), (10, 2))$
3. Set PBC $(10, (5, 3), (3, 6), (6, 8), (8, 5))$
4. Move P $(10, ((2, 1), \text{S}), ((1, 9), \text{W}), ((9, 10), \text{N}), ((10, 2), \text{E}))$
5. Move P $(10, ((1, 2), \text{N}), ((2, 10), \text{E}), ((10, 9), \text{S}), ((9, 1), \text{W}))$

The sequence yields $\sigma_{10} = 8$. Moving fills in gaps and relieve congested areas.

Cellular automata rules solving the wireless... 441
Fig. 22 Formation of a non-optimal pattern for $n = 11$. The 4-step sequence gives: Initialize (11, 9) → Set_PBC (11, (1, 7), (7, 11), (11, 5), (5, 1)) → Set_PBC (11, (3, 4), (4, 9), (9, 8), (8, 3)) → Add_P (11, (2, 2)), Add_P (11, (2, 10)), Add_P (11, (10, 10)), Add_P (11, (10, 2)). The sequence yields $\sigma'_{11} = 13$ (the required optimal result must be $\sigma'_{11} = 11$ as explained thereafter).
Fig. 23 Formation of a valid pattern for $n = 12$. The 5-step sequence gives:
Initialize $(12, 12) \rightarrow$ Set_PBC $(12, (1, 5), (5, 12), (12, 8), (8, 1)) \rightarrow$
Set_PBC $(12, (2, 10), (10, 11), (11, 3), (3, 2)) \rightarrow$ Add_P $(12, (1, 12))$
$\rightarrow$ Move_P $(12, (2, 10), E)$. The sequence yields $\sigma_{12} = 13$. The last move just gives a slightly better pattern.
Appendix A

Spectral detection of valid \( \min n \times n \) patterns for \( n = 11 \)

Fig. 24 Isospectral class \( C_{11} = \{ C_{11}^{1,2}, C_{11}^{1,5}, C_{11}^{2,4}, C_{11}^{1,14}, C_{11}^{3,5} \} \) of 11-circulants with their associated patterns. These patterns are not suitable for the minimum coverage problem. Last image: the pattern of \( C_{11}^{1,2} \) is decorated from the \( n_2 \)-prototile, with \( n \) prototiles and \( 2n \) gaps (\( n = 11 \)). Each cell is "decorated" from its von Neumann label of Fig. 5. The number of labels at site \((x, y)\) is the cover level \( v_{ij} = v(x, y) \) with one color per eigenvalue in \( M_{n_2} \).
Fig. 25 Isospectral class \( \mathcal{C}_2 = \{ C_{11}^{1,3}, C_{11}^{4,4}, C_{11}^{2,3}, C_{11}^{2,4}, C_{11}^{3,5} \} \) of 11-circulants with their associated patterns. These patterns are optimal for the minimum coverage problem. Last image: the pattern of \( C_{11}^{2,3} \) is decorated from the \( \mathbf{n}_2 \)-prototile, with \( n \) prototiles and \( 2n \) (colored) overlaps (\( n = 11 \)). Each cell is “decorated” from its von Neumann label of Fig. 5. The number of labels at site \( (x, y) \) is the cover level \( v_{ij} = v(x, y) \) with one color per eigenvalue in \( M_n \). 

Cellular automata rules solving the wireless...
Funding Open Access funding enabled and organized by Projekt DEAL.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article’s Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article’s Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

Ab NA, Aziz AKA, Ismail WZW (2009) Coverage strategies for wireless sensor networks, world academy of science. Eng Technol 3(2):171–176

Achasova S, Bandman O, Markova V, Piskunov S (1994) Parallel substitution algorithm, theory and application. World Scientific, Singapore

Alia OM, Al-Ajouri A (2017) Maximizing wireless sensor network coverage with minimum cost using harmony search algorithm. IEEE Sensors J 17(3):882–896

Boesch F, Tindell R (1984) Circulants and their connectivities. J Graph Theory 8(4):487–499

Cardei M, Thai MT, Li Yingshu, Wu Weiil (2005) Energy–Efficient Target Coverage in Wireless Sensor Networks, INFOCOM – Proc. 24th Conf. IEEE Comm. Society 1976–1984

Charr J, Deschinkel K, Mansour RH, Hakem M (2019) Optimizing the lifetime of heterogeneous sensor networks under coverage constraintAê–>–A MILP and genetic based approaches. WiMob 2019 - Int. Conf Wireless and Mobile Computing, Networking and Comm. 2019:1–6

Cheng MX, Ruan J, Wu W (2005) Achieving Minimum Coverage Breach under Bandwidth Constraints in Wireless Sensor Networks, INFOCOM – Proc. 24th Conf. IEEE Comm. Society 2638–2645

Church RL, ReVelle CS (1976) Theoretical and computational links between the p-median, location set-covering, and the maximal covering location problem. Geogr Anal 8(4):406–415

Davis PJ (1970) Circulant matrices. Wiley, New York

Désérable D (1999) A family of Cayley graphs on the hexavalent grid. Discr Appl Math 93(2–3):169–189

Gasior J, Seredynski F, Hoffmann R (2018) Towards Self-organizing Sensor Networks: Game-Theoretic r-Learning Automata-Based Approach, ACRI 2018 – Cellular Automata, Mauri G., El Yacoubi S., Dennunzio A., Nishinari K., Manzoni L. (eds) LNCS 11115:125–136

Gomes FC, Meneses CN, Pardalos PM, Valdisio G, Viana R (2006) Experimental analysis of approximation algorithms for the vertex cover and set covering problems. Computers Op Res 33(12):3520–3534

Hakimi SL (1965) Optimum distribution of switching centers in a communication network and some related graph theoretic problems. Op Res 13(3):462–475

He X, Fu X, Yang Y (2019) Energy-efficient trajectory planning algorithm based on multi-objective PSO for the mobile sink in wireless sensor networks. IEEE Access 7:176204–217

Hoffmann R (2022) Forming Point Patterns by a Probabilistic Cellular Automata Rule, eprint http://arxiv.org/abs/2202.06656 based on a talk presented at Summer Solstice Conference on Discrete Models of Complex Systems, Dresden 2019. http://arxiv.org/abs/2202.06656

Hoffmann R, Désérable D, Seredyński F (2021) A cellular automata rule placing a maximal number of dominoes in the square and diamond. J Supercomput 77:9069–9087

Hoffmann R, Désérable D, Seredyński F (2019) A Probabilistic Cellular Automata Rule Forming Domino Patterns, PaCT 2019, Malyshkin V. (ed.) Parallel Computing Technologies, LNCS 11657:334–344

Hoffmann R, Désérable D, Seredyński F (2021) Minimal Covering of the Space by Domino Tiles, PaCT 2021, Malyshkin V. (ed.) Parallel Computing Technologies, LNCS 12942:453–465

Hoffmann R, Seredyński F (2020) Covering the Space with Sensor Tiles, ACRI 2020, Gizdalla T.M., Manzoni L., Sirakoulis G.C., Bandini S., Podlaski K. (eds) Cellular Automata, LNCS 12599:156–168

Huber KP (1994) Codes over Eisenstein–Jacobi integers, Finite Fields: Theory, Applications & Algorithms, G.L. Mullen, P.J.S. Shiue, eds., Contemporary Mathematics168:165–179

Jawad HM, Jawad AM, Nordin R, Gharghan SK, Abdullah NF, Ismail M, Abu-AlShaer MJ (2020) Accurate empirical path-loss model based on particle swarm optimization for wireless sensor networks in smart agriculture. IEEE Sensors J 20(1):552–561

Jia J, Dong C, He X, Li D, Yu Y (2012) Sensor scheduling for target coverage in directional sensor networks. Int J Distrib Sens Networks 13(6):1–12

Jiao Z, Zhang L, Xu M, Cai C, Xiong J (2019) Coverage control algorithm-based adaptive particle swarm optimization and node sleeping in wireless multimedia sensor networks. IEEE Access 7:170096–105

Liao C, Ting C (2018) A novel integer-coded memetic algorithm for the set k-cover problem in wireless sensor networks. IEEE Trans Cybern 48(8):2245–2258

Lin Y, Wang X, Hao F, Wang L, Zhang L, Zhao R (2018) An on-demand coverage based self-deployment algorithm for big data perception in mobile sensing networks. Fut Gener Computer Syst 82:220–234

Manju, Chand S, Kumar B (2018) Genetic algorithm–based metaheuristic for target coverage problem, IET Wireless Sensor Syst 8(4):170–175

Martínez C, Stafford E, Beivide R, Gabidulin EM (2008) Modeling hexagonal constellations with Eisenstein-Jacobi graphs. Probl Inf Transm 44(1):1–11

Mehrez A (1987) Facility location problems, review, description, and analysis, in. Geogr Res Forum 8:113–129

Monakhova EA, Romanov A, Lezhnev E (2020) Shortest path search algorithm in optimal two-dimensional circumulant networks: implementation for networks-on-chip. IEEE Access 8:215010–19

Mostafaei H, Meybodi MR (2013) Maximizing lifetime of target coverage in wireless sensor networks using learning automata. Wireless Pers Comm 71:1461–1477

Plénét T, El Yacoubi S, Raievsky C, Lefèvre L (2021) Observability of Affine Cellular Automaton Through Mobile Sensors, ACRI 2020 – Cellular Automata, Gizdalla T.M., Manzoni L., Sirakoulis G.C., Bandini S., Podlaski K. (eds) LNCS 12599:6–45

Patan R, Razi A, Hua KA, Majidi A (2017) NQ–GPLS: N-queen inspired gateway placement and learning automata-based gateway...
selection in wireless mesh network, MobiWaC ’17 – Proc.15th ACM Int. Symp. on Mobility Management and Wireless Access 41–44

Richter S, Helmert M, Gretton C (2007) A Stochastic Local Search Approach to Vertex Cover, In: Hertzberg J., Beetz M., Englert R. (eds) KI 2007: Advances in Artificial Intelligence LNCS 4667:412–426

Saadi N, Bounceur A, Euler R, Lounis M, Bezoui M, Kerkar M, Pottier B (2020) Maximum lifetime target coverage in wireless sensor networks. Wireless Personal Comm 111:1525–1543

Seredyn´ski F, Gasior J, Hoffmann R (2021) The Second Order CA–Based Multi-agent Systems with Income Sharing, ACRI 2020 – Cellular Automata, Gwizdalla T.M., Manzoni L., Sirakoulis G.C., Bandini S., Podlaski K. (eds) LNCS 12599:34–145

Sierpiński W (1916) O krzywej, której każdy punkt jest punktem rozgałęzienia, Prace Mat.–Fiz. 27 (1):77–86

Snyder LV (2011) Covering problems, In Foundations of Location Analysis 6:109–135. Springer.

Thai My T, Wang Feng, Du D Hongwei, Jia X (2008) Coverage problems in wireless sensor networks: designs and analysis. Int J Sensor Netw 3(3):191–200

Tret’yakova A, Seredyński F, Bouvry P (2016) Graph cellular automata approach to the maximum lifetime coverage problem in wireless sensor networks. Simulation 92(2):153–164

Yebra JLA, Fiol MA, Morillo P, Alegre I (1985) The diameter of undirected graphs associated to plane tessellations. Ars Combinatoria 20–B:159–171

Yetgin H, Cheung KTH, El-Hajjar M, Hanzo LH (2017) A survey of network lifetime maximization techniques in wireless sensor networks. IEEE Comm Surv Tutorials 19(2):828–854

Zhong J, Huang Z, Feng L, Du W, Li Y (2020) A hyper-heuristic framework for lifetime maximization in wireless sensor networks with a mobile sink. IEEE/CAA J Automatica Sinica 7(1):223–236

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.