Mining Frequent Patterns in 2D+t Grid Graphs for Cellular Automata Analysis

Romain Deville, Elisa Fromont, Baptiste Jeudy, Christine Solnon

Mining Frequent Patterns in 2D+t Grid Graphs for Cellular Automata Analysis

Romain Deville[1,2], Elisa Fromont[1], Baptiste Jeudy[1], and Christine Solnon[2]

[1] UJM, CNRS, LaHC UMR 5516, F-42000, Saint-Etienne, France
[2] Université de Lyon, INSA-Lyon, LIRIS UMR 5205, F-69621, Villeurbanne, France

Abstract. A 2D grid is a particular geometric graph that may be used to represent any 2D regular structure such as, for example, pixel grids, game boards, or cellular automata. Pattern mining techniques may be used to automatically extract interesting substructures from these grids. 2D+t grids are temporal sequences of grids which model the evolution of grids through time. In this paper, we show how to extend a 2D grid mining algorithm to 2D+t grids, thus allowing us to efficiently find frequent patterns in 2D+t grids. We evaluate scale-up properties of this algorithm on 2D+t grids generated by a classical cellular automaton, *i.e.*, the game of life, and we show that the extracted spatio-temporal patterns may be used to analyze this kind of cellular automata.

1 Introduction

A 2D grid is a particular geometric graph that may be used to model any 2D regular structure such as, for example, grids of pixels (*i.e.*, images), game boards, or cellular automata. To characterize these grids, we may mine them to extract recurrent patterns [6]. In some applications, we use temporal sequences of grids (*i.e.*, 2D+t grids) to model the evolution of grids through time. This is the case, for example, of videos, or sequences of actions in board games. In this paper, we motivate and illustrate our work on Cellular Automata (CA) used to model the temporal evolution of ecosystems [3,13,12]. Indeed, biodiversity of ecosystems is increasingly recognized as an important element of global change. CA-based models are used to understand, predict and control spatio-temporal spread of species which is a key issue to preserve biodiversity [9]. A CA is a regular grid of cells. Each cell has a state which evolves through time, depending on the state of its neighbours in the grid. One of the most famous CA is the Game of Life [5]. In this CA, the grid is in 2 dimensions (on toric grids), and each cell has 8 neighbours (horizontally, vertically, and diagonally). Initially (at time $t = 0$), each cell is either alive or dead. The state at time $t + 1$ of a cell depends on its state and on the state of its 8 neighbours at time t. It is computed by applying the following rules: 1) if the cell is alive at time t and has 2 or 3 living neighbours, then it is alive at time $t + 1$, otherwise it becomes dead; 2) if the cell is dead at time t and has exactly 3 living neighbours, then it becomes alive at time $t + 1$, otherwise it stays dead. When executing a CA from a given initial state, one may observe the emergence of spatio-temporal patterns, and these patterns are

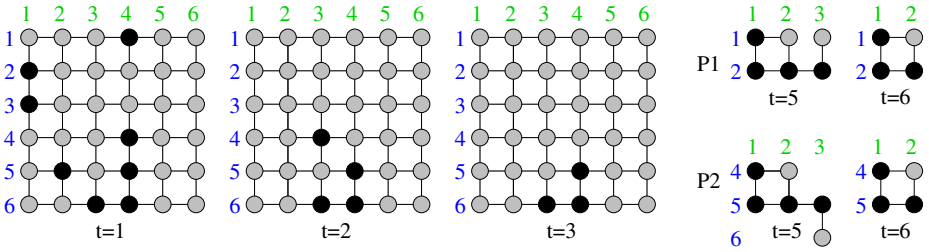


Fig. 1: Left: First three states of a 6×6 game of life modelled with a $2D + t$ grid. Living (resp. dead) cells are displayed in black (resp. gray). (x, y) coordinates are displayed in green and blue, respectively. Temporal edges are not displayed, but there is a temporal edge between each pair of nodes (i, j) such that $x_i = x_j$, $y_i = y_j$, and $|t_i - t_j| = 1$. Right: Two examples of spatio-temporal patterns (temporal edges are not displayed). P1 is isomorphic to a subgrid of the grid on the left (with translation $T = (2, 4, -4)$ and rotation $\theta = -\pi/2$). P2 is also isomorphic to a subgrid of the grid on the left. P2 is not isomorphic to P1 because the angle between edges (a, b) and (b, c) with $x_a = y_a = y_b = 2, x_b = x_c = 3$, and $y_c = 1$ in P1 is not preserved in P2.

characteristic of different ecosystem outcomes. [12] distinguishes four possible outcomes: (1) development of a homogeneous fixed pattern, (2) development of a periodic pattern, (3) development of a chaotic pattern, and (4) development of patterns composed of homogeneous regions and regions containing complex localized structures.

In this paper, we present an efficient algorithm for extracting spatio-temporal patterns in 2D+t grids. This algorithm may be used, for example, to extract meaningful spatio-temporal patterns in CA. When CA are used to model ecosystems, these patterns could be used by ecologists to better understand and control the dynamics of the ecosystems. For example, [3] explains that we can foresee the future of an ecosystem by identifying recurring patterns. Ecologists are also interested in understanding how dependent the patterns and the initial state are.

Our algorithm is an extension of GriMA [6], an algorithm for mining 2D grids which has been designed to tackle real-life applications such as image classification. This algorithm is recalled in Section 2. In Section 3, we show how to extend it to mine 2D+t grids. In Section 4, we evaluate scale-up properties of our new algorithm for mining game-of-life CA, and we show that the extracted spatio-temporal patterns are relevant for classification purposes.

2 Background on 2D Grids and GriMA

Definition of 2D grids, and 2D subgrid isomorphism. A 2D grid is a special case of graph such that each node has a 2D coordinate which is a couple of integer values, and each edge connects nodes which are neighbours on a grid. More formally, a **grid** is defined by $G = (N, E, L, x, y)$ such that N is a set of nodes, $E \subseteq N \times N$ is a set of edges, $L : N \cup E \to \mathbb{N}$ is a labeling function which

associates a label $L(c)$ with every component (node or edge) $c \in N \cup E$, and $x : N \to \mathbb{Z}$ and $y : N \to \mathbb{Z}$ map each node $u \in N$ to its 2D coordinates (x_u, y_u), and $\forall (u, v) \in E$, $|x_u - x_v| + |y_u - y_v| = 1$. A **subgrid** of a grid $G = (N, E, L, x, y)$ is a grid $G' = (N', E', L', x', y')$ such that $N' \subseteq N$, $E' \subseteq E \cap N' \times N'$ and L', x', and y' are the restrictions of L, x, and y to $N' \cup E'$, N', and N', respectively.

Looking for patterns in a grid amounts to searching for subgrid isomorphisms. Patterns should be invariant to translations and rotations. More formally, the **translation** of $G = (N, E, L, x, y)$ by a vector $T \in \mathbb{Z}^2$, denoted $G + T$, is the grid obtained by moving all its nodes with respect to T, *i.e.*, $\forall u \in N$, (x_u, y_u) becomes $(x_u, y_u) + T$. Let $\Theta = \{\left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right), \left(\begin{smallmatrix} 0 & -1 \\ 1 & 0 \end{smallmatrix}\right), \left(\begin{smallmatrix} -1 & 0 \\ 0 & -1 \end{smallmatrix}\right), \left(\begin{smallmatrix} 0 & 1 \\ -1 & 0 \end{smallmatrix}\right)\}$ be the set of rotation matrices of respective angles 0, $\pi/2$, π and $3\pi/2$. The **rotation** of G with respect to $\theta \in \Theta$, denoted θG, is the grid obtained by rotating all its nodes with respect to θ, *i.e.*, $\forall u \in N, (x_u, y_u)$ becomes $(x_u, y_u)\theta$. Two grids G_1 and G_2 are **grid isomorphic** if there exist a translation $T \in \mathbb{Z}^2$ and a rotation $\theta \in \Theta$ such that $G_1 = T + \theta G_2$. Finally, G_1 is **sub-grid-isomorphic** to G_2 if there exists a subgrid of G_2 which is isomorphic to G_2 (see Fig. 1 for an example).

Graph mining. Given a database D of graphs and a frequency threshold σ, the goal of the graph mining problem is to output all frequent subgraphs in D, *i.e.*, all graphs G such that there exist at least σ graphs in D to which G is sub-isomorphic. This problem may be solved by gSpan [14], and all similar general exhaustive graph mining algorithms [8]. However, as the subgraph isomorphic problem is $\mathcal{NP}$-complete, these algorithms do not scale well. On the other hand, Plagram [11] and FreqGeo [1] are graph mining algorithms dedicated to special cases of graphs for which subgraph isomorphism becomes polynomial, *i.e.*, plane graphs for Plagram and geometric graphs for FreqGeo. These algorithms have better scale-up properties. However, Plagram only mines patterns composed of faces and the smallest possible subgraph pattern is a single face, *i.e.*, a cycle with 3 nodes. Using Plagram to mine grids is possible but the problem needs to be transformed such that each grid node becomes a face in the graph tackled by Plagram. This transformation artificially increases the number of nodes and edges which causes a scalability problem for Plagram. Also, grids are special cases of geometric graphs. Therefore, FreqGeo may be used to mine grids. However, it has a higher time-complexity than GriMA, the 2D grid mining algorithm introduced in [6].

Description of GriMA. GriMA follows the same basic principle as gSpan, Plagram, and FreqGeo to avoid generating the same pattern multiple times: It uses codes to represent grids. This code is a list of edges encountered when performing a traversal of the grid. A grid may have several codes but one of them is chosen as the signature: The canonical code, which is the largest code wrt lexicographic order. GriMA explores the search space of all canonical codes in a depth-first recursive way. It first computes all frequent edges and then calls an `Extend` function for each of these frequent extensions. `Extend` has one input parameter: A pattern code P which is frequent and canonical. It outputs all frequent canonical codes P' such that P is a prefix of P'. To this aim, it first

computes the set E of all possible valid extensions of all occurrences of P in the database D of grids: A valid extension is the code e of an edge such that $P.e$ occurs in D. Finally, `Extend` is recursively called for each extension e such that $P.e$ is frequent and canonical. Hence, at each recursive call, the pattern grows.

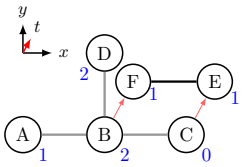
3 2D+t Grid Mining Algorithm

A $2D+t$ grid is defined by a tuple (N, E, L, x, y, t) such that (N, E, L, x, y) is a 2D grid graph and $t : N \to \mathbb{Z}$ is a function that maps nodes to temporal coordinates, *i.e.*, $\forall u \in N$, t_u is the temporal coordinate of node u. Also, edges are enforced to connect neighbour nodes in the grid, *i.e.*, $\forall (u, v) \in E$, $|x_u - x_v| + |y_u - y_v| + |t_u - t_v| = 1$. We distinguish two different kinds of edges: spatial edges (such that $|x_u - x_v| + |y_u - y_v| = 1$) and temporal edges (such that $|t_u - t_v| = 1$).

2D grid isomorphism is defined so that isomorphism is invariant to translations and rotations. When extending this definition to 2D+t grids, we still ensure that isomorphism is invariant to translations wrt all axis. However, as time is an oriented dimension, we allow rotations only along the temporal axis. Hence, we consider the set $\Theta = \{\left(\begin{smallmatrix}1&0&0\\0&1&0\\0&0&1\end{smallmatrix}\right), \left(\begin{smallmatrix}0&-1&0\\1&0&0\\0&0&1\end{smallmatrix}\right), \left(\begin{smallmatrix}-1&0&0\\0&-1&0\\0&0&1\end{smallmatrix}\right), \left(\begin{smallmatrix}0&1&0\\-1&0&0\\0&0&1\end{smallmatrix}\right)\}$ of rotation matrices of respective angles 0, $\pi/2$, π and $3\pi/2$ along the temporal axis.

To extend GriMA to 2D+t grids, we have to define the canonical code of a 2D+t grid. A **code** $C(G)$ of a 2D+t grid G is a sequence of n edge codes ($C(G) = \langle ec_0, ..., ec_{n-1} \rangle$) which is associated with a depth-first traversal of G starting from a given initial node. During this traversal, each edge is traversed once, and nodes are numbered: The initial node has number 0; each time a new node is discovered, it is numbered with the smallest integer not already used in the traversal. Each edge code corresponds to a different edge of G and the order of edge codes in $C(G)$ corresponds to the order edges are traversed. Hence, ec_k is the code associated with the k^{th} traversed edge. This edge code ec_k is the tuple $(\delta, i, j, a, L_i, L_j, L_{(i,j)})$ where :

- i and j are the numbers associated with the nodes of the k^{th} traversed edge.
- $\delta \in \{0, 1\}$ is the direction of the k^{th} traversed edge:
  - $\delta = 0$ if it is forward, *i.e.*, j is a new node reached for the first time;
  - $\delta = 1$ if it is backward, *i.e.*, j already appears in $\langle ec_0, ..., ec_{k-1} \rangle$.
- $a \in \{-2, -1, 0, 1, 2, 3\}$ is the angle value of the k^{th} traversed edge (i, j):
  - if (i, j) is a temporal edge, then $a = -2$ if $t_i = t_j + 1$, and $a = -1$ if $t_i = t_j - 1$;
  - else, (i, j) is a spatial edge.
    - If (i, j) is the first spatial edge encountered since the beginning of the traversal, then $a = 0$.
    - Else, let (l, m) be the first spatial edge in $\langle ec_0, ..., ec_{k-1} \rangle$ such that $x_i = x_m$ and $y_i = y_m$. We have $a = 2A/\pi$ where $A \in \{0, \pi/2, \pi, 3\pi/2\}$ is the angle between (l, m) and (i, j) in the x, y plane.
- L_i, L_j, $L_{(i,j)}$ are labels of i, j, and (i, j), respectively.



Code 1								Code 2								Code 3							
edge	δ	i	j	a	L_i	L_j	L_{ij}	edge	δ	i	j	a	L_i	L_j	L_{ij}	edge	δ	i	j	a	L_i	L_j	L_{ij}
(D,B)	0	0	1	0	2	2	0	(D,B)	0	0	1	0	2	2	0	(C,E)	0	0	1	-1	2	1	0
(B,C)	0	1	2	3	2	0	0	(B,A)	0	1	2	1	2	1	0	(E,F)	0	1	2	0	1	1	0
(C,E)	0	2	3	-1	0	1	0	(B,C)	0	1	3	3	2	0	0	(F,B)	0	2	3	-2	1	2	0
(E,F)	0	3	4	0	1	1	0	(C,E)	0	3	4	-1	0	1	0	(B,D)	0	3	4	1	2	2	0
(F,B)	1	4	1	-2	1	2	0	(E,F)	0	4	5	0	1	1	0	(B,A)	0	3	5	2	2	1	0
(B,A)	0	1	5	1	2	1	0	(F,B)	1	5	1	-2	1	2	0	(B,C)	1	3	0	3	2	0	0

Fig. 2: Left: A 2D+t grid (temporal edges are displayed in red, node labels are displayed next to nodes, and all edges have the same label 0). Right: 3 codes for this grid (other codes may be built by changing the traversal).

For example, let us consider code 1 in Fig. 2. Let us explain how the code of the fourth traversed edge (E,F) is built. $\delta = 0$ because (E,F) is a forward edge (F has not been reached before). (E,F) is a spatial edge, and the first spatial edge (l,m) such that m has the same spatial coordinates as E is (B,C). The angle between (E,F) and (B,C) is 0. So, $a = 0$. For the fifth edge of code 1, (F,B), $\delta = 1$ because B has already been reached before (backward edge). As (F,B) is a temporal edge, $a = -2$.

Given a code, we can reconstruct the corresponding grid since edges are listed in the code together with angles and labels. However, there exist different possible codes for a given grid, as illustrated in Fig. 2: Each code corresponds to a different traversal (starting from a different initial node and choosing edges in a different order). As we did for GriMA, we define a total order on the set of all possible codes that may be associated with a given grid by considering a lexicographic order (all code components have integer values). Among all the possible codes for a grid, the largest one according to this order is the canonical code of this grid and it is unique. For example, in Fig. 2, code 1 is canonical: It is greater than codes 2 and 3, and it is also greater than all other possible codes for this grid (not shown here).

Note that it is not necessary to exhaustively build all codes when computing a canonical code. We use heuristics to first build large codes (by first choosing spatial edges with $3\pi/2$ angles, such as for (D,B) and (B,C), for example). Also, when building a code, we stop the traversal as soon as the corresponding code becomes smaller than the largest current code.

This canonical code for 2D+t allows us to extend GriMA to mine 2D+t grids in a straightforward way, and we can show that the resulting mining algorithm, called GriMA2D+t, is both *correct* (it only outputs frequent subgrids) and *complete* (it cannot miss any frequent subgrid). The proof (not detailed due to lack of space) basically shows that every prefix of a canonical code is canonical.

GriMA2D+t enumerates all frequent patterns in $\mathcal{O}(kn^2.|P|^2) = \mathcal{O}(kn^4)$ time per pattern P, where k is the number of grids in the set D of input grids, n the size of the largest grid $G_i \in D$ (in number of edges) and $|P|$ the number of edges in a pattern P.

Node-induced GriMA*2D+t.* In our application, the mined grids are complete and have no label on edges. Thus, we designed a variant of GriMA2D+t, called node-induced-GriMA2D+t, which computes node-induced grids, i.e. grids induced by their node sets. This corresponds to a "node-induced" closure operator on graphs where, given a pattern P, we add all possible edges to P without adding new nodes. We have shown in [6] that this optimization decreases the number of extracted patterns and the extraction time.

Limitation on edge extension. Moreover, to avoid mining patterns that only contain dead cells, we also limit the extension procedure of our mining process. In the `Extend` function, we forbid extension with edges linking two dead cells. As a consequence, every edge (i, j) in a mined pattern is such that either i, or j, or both i and j correspond to living cells.

4 Experiments

We study the scale-up properties of GriMA$2D + t$ and assess the relevance of the mined patterns on a classification task related to the behavior of a CA, *i.e.*, the *Game of Life* described in Section 1. More precisely, given the k first cell states, with $k \in \{1, 2, 5, 10, 20\}$, the goal is to forecast the outcome at time $t = 1000$, where we only consider two possible outcomes: *dead* (if all cells are dead at time $t = 1000$), or *alive* (if at least one cell is alive at time $t = 1000$).

Dataset. We consider four sizes of grids $n \times n$, with $n \in \{20, 30, 40, 50\}$. For each size n, we randomly choose the initial state (dead or alive) of each $n \times n$ cell wrt to a cell probability p. We have chosen p in such a way that the outcome at time $t = 1000$ is *dead* or *alive* with equal probabilities. This way, we ensure during our dataset generation process that there is no bias towards one of the two classes. This imposes a cell probability p of 74%, 78%, 80%, and 81% for $n = 20, 30, 40$, and 50, respectively. Besides, to avoid trivial predictions of the class *dead*, due to the fact that all cells may be dead before the k^{th} iteration, we only select initial states such that there is at least one cell alive at the 50^{th} iteration. For each size $n \in \{20, 30, 40, 50\}$, we generate a set S_n of 2000 initial states such that the outcome at time $t = 1000$ is *dead* for half of them (S_n^d), and *alive* for the other half (S_n^a). We split each set S_n^d and S_n^a into two equal parts for learning (L_n^d and L_n^a) and training (T_n^d and T_n^a).

2D+t grids. For each state $s_i \in S_n$ (with $n \in \{20, 30, 40, 50\}$), and for each temporal horizon $k \in \{1, 2, 5, 10, 20\}$, we build a 2D+t grid $G(s_i, k)$ which is a temporal sequence of k $2D$ grids: The first one corresponds to the state s_i, and the next $k - 1$ ones correspond to states obtained by iteratively applying the game-of-life rules starting from s_i. Each node is labeled with either 0 (*dead* cell) or 1 (cell *alive*), and all edges have the same label.

Mining process. For each size $n \in \{20, 30, 40, 50\}$ and each temporal horizon $k \in \{1, 2, 5, 10, 20\}$, we mine frequent patterns in the learning sets. This is done for each class separately: We compute the set $F^d_{n,k}$ (resp. $F^a_{n,k}$) of frequent patterns in all $G(s_i, k)$ with $s_i \in S^d_n$ (resp. $s_i \in S^a_n$). We consider two different frequency threshold $\sigma \in \{50\%, 100\%\}$: When $\sigma = 50\%$ (resp. $\sigma = 100\%$), a pattern is frequent if it is present in half of the grids (resp. all the grids). Note that, the higher the frequency, the lower the number of mined patterns and the more efficient the mining process. Each mining process has been limited to 12 hours of CPU time: If the mining process is not completed after 12 hours, we stop it and consider the subset of patterns that have been extracted within this time limit.

Classification process. For each size $n \in \{20, 30, 40, 50\}$ and each temporal horizon $k \in \{1, 2, 5, 10, 20\}$, we build the set $F_{n,k} = F^a_{n,k} \cup F^d_{n,k}$ that contains all frequent patterns (in the two classes). Then, for each state $s_i \in L^d_n \cup L^a_n$, we count the number of occurrences of each pattern of $F_{n,k}$ in $G(s_i, k)$, and build a frequency vector that gives the frequency of each pattern. Hence, each state is represented by a histogram of frequent substructures.

We report two sets of experiments: One with histograms created using all the patterns mined on both classes (which can be very sparse) and one with a selected subset of 100 patterns. This post-processing selection is performed using the relevance score and the greedy selection algorithm presented in [7]. To fasten the preprocessing step, we delete at each of the 100 iterations of the greedy algorithm, the patterns with the 10% lowest scores.

Frequency vectors (of length $|F_{n,k}|$ or 100) are used to train a binary Support Vector Machine (SVM) to discriminate between the two classes. We use the Libsvm [4] library with the intersection kernel presented in [10] (known to be good on histograms).

Finally, we use the trained model to forecast the class of each state in our training set: For each state $s_i \in T^d_n \cup T^a_n$, we count the number of occurrences of each pattern of $F_{n,k}$ (or the 100 selected patterns of $F_{n,k}$) in $G(s_i, k)$, and build a frequency vector which is used by the SVM model to forecast an outcome (dead or alive) which is compared to the true outcome (dead for states coming from T^d_n and alive for states coming from T^a_n). We report accuracy results, *i.e.*, the percentage of states for which the forecasted outcome is equal to the true outcome.

Accuracy results. We report accuracy results in Table 1. When increasing the temporal horizon k (*i.e.*, the temporal size of the mined grids), accuracy results are improved. This shows the relevance of the GriMA2D+t algorithm compared to GriMA. However, when increasing k, the mining process needs more time and we often had to stop the mining process after 12 hours (red cells) for the largest values of k. In this case, we only explored part of the substructure search space.

Also, the larger the grid size n, the better the results. It is well known that, for the game of life, large grids have higher probabilities of containing stable

k		1		2		5		10		20	
n	σ	50%	100%	50%	100%	50%	100%	50%	100%	50%	100%
20	$\lvert F_{n,k}\rvert$	673	22	23589	64	824616	2478	707743	96762	417724	213861
	All Pat	72.40	70.70	77.30	72.50	83.40	85.20	85.00	88.70	88.30	91.30
	100 Pat	72.70		75.80		83.80	83.50	84.10	87.80	85.60	89.50
30	$\lvert F_{n,k}\rvert$	662	18	28795	68	783701	2472	688546	99827	355381	252891
	All Pat	77.00	68.40	81.60	76.60	84.80	87.80	88.00	89.10	92.70	92.80
	100 Pat	74.10		79.90		84.40	86.90	88.40	89.40	91.80	92.20
40	$\lvert F_{n,k}\rvert$	667	27	38103	77	786619	4620	634501	178710	403411	246885
	All Pat	79.10	70.90	86.50	82.10	89.90	92.50	91.60	93.10	95.50	96.00
	100 Pat	77.10		84.40		86.80	90.10	89.40	93.60	94.30	96.80
50	$\lvert F_{n,k}\rvert$	740	26	46235	79	906209	4171	720779	206508	373600	282003
	All Pat	78.60	72.70	82.80	79.70	89.30	90.90	91.80	93.20	96.40	95.80
	100 Pat	77.90		84.20		88.50	89.40	89.90	92.90	91.90	96.40

Table 1: Accuracy results for the classification of states in $T_n^d \cup T_n^a$. For each size $n \in \{20, 30, 40, 50\}$, the first line reports the number of frequent patterns $|F_{n,k}|$, and the cell is colored in red if the 12 hour time-out has been reached and green otherwise; the second and third line report accuracy results with vectors of size $|F_{n,k}|$ and 100, respectively (if $|F_{n,k}| < 100$, results are not given for vectors of size 100). Each line gives results for $k = 1, 2, 5, 10$, and 20, and with $\sigma = 50\%$ and 100% .

patterns that may characterize *alive* outcomes. However, as with k, we often had to stop the mining process after 12 hours for the largest grids. This shows the necessity of efficient algorithms to tackle real-life problems.

The number of mined patterns $|F_{n,k}|$, is smaller when the frequency threshold $\sigma = 100\%$ than when it is 50%. For small temporal horizons $k \in \{1, 2\}$, the number of mined patterns is not large enough (smaller than 27 for $k = 1$ and than 79 for $k = 2$). In this case, the results obtained with $\sigma = 100\%$ are worse than those obtained with $\sigma = 50\%$. However, for larger time horizons $k \in \{5, 10, 20\}$, the number of mined patterns becomes large enough for $\sigma = 100\%$ while it becomes so large for $\sigma = 50\%$ that the mining process is never completed. As we only have a subset of the frequent patterns in this case, it may be possible that some relevant patterns have not be found. We observe that in this case the results are worse with $\sigma = 50\%$ than with $\sigma = 100\%$.

Finally, let us compare the results obtained when all patterns of $F_{n,k}$ are used for the classification (All Pat) with the results obtained when we only use the 100 first patterns selected by the post-processing process (100 Pat): The difference is usually rather small, and in some cases it improves results (*e.g.*, $k = 20, n = 40$) whereas in some other cases it degrades them (*e.g.*, $k = 20, n = 20$). However, the post-processing improves the efficiency of the counting step: The process of counting all occurrences of all patterns of $F_{n,k}$ (to create histograms used as inputs for the SVMs) takes on average 0.002 seconds when $k = 1$ and up to 45 seconds when $k = 20$, whereas it takes 0.0005 seconds when $k = 1$ and up

k		1		2		5		10		20	
n		All Pat	100 Pat	All Pat	100 Pat	All Pat	100 Pat	All Pat	100 Pat	All Pat	100 Pat
20	Depth	0(0)	0(0)	0.9(1)	0.9(1)	2.1(4)	2.0(4)	5.3(9)	4.5(9)	11.7(19)	9.0(19)
	NbCell	6.4(11)	6.6(10)	7.9(15)	7.2(11)	11.3(25)	11.8(20)	14.8(28)	15.3(24)	18.3(32)	14.7(25)
30	Depth	0(0)	0(0)	0.9(1)	0.9(1)	2.3(4)	2.4(4)	6.4(9)	6.0(9)	14.5(19)	11.1(19)
	NbCell	6.4(11)	6.4(9)	8.4(17)	7.7(12)	11.7(23)	11.0(18)	16.8(35)	15.3(29)	21.5(32)	16.6(25)
40	Depth	0(0)	0(0)	0.9(1)	0.9(1)	2.3(4)	2.2(4)	6.8(9)	6.4(9)	16.2(19)	14.1(19)
	NbCell	6.4(12)	6.3(9)	9.0(18)	7.6(13)	12.1(24)	11.1(17)	20.2(39)	21.3(35)	23.4(34)	19.6(27)
50	Depth	0(0)	0(0)	1.0(1)	0.9(1)	2.3(4)	2.3(4)	6.7(9)	6.5(9)	16.5(19)	15.6(19)
	NbCell	6.5(12)	6.7(11)	9.2(19)	7.5(13)	13.2(27)	12.3(21)	19.4(39)	20.5(33)	24.1(34)	21.8(30)

Table 2: Average and Maximum (in parenthesis) depth and number of cells for all patterns of $F_{n,k}$ or only those selected with post processing.

to 0.008 seconds when $k = 20$ if we only count occurrences of the 100 patterns selected by post-processing.

Overall, those results show that taking into account the structural information along the spatio-temporal grids can be used for the prediction of the outcome of cellular automata and the extension to temporal dimension of our grid mining algorithm can be used to tackle spatio-temporal problems.

Patterns statistics. Table 2 reports some statistics about the mined patterns (all patterns in $F_{n,k}$, or the 100 ones selected by post-processing). We report the average and maximum (in parenthesis) number of nodes of each pattern as well as their depth, i.e., the number of temporal steps on which the patterns are present (spatial patterns have a depth of 0). The average number of nodes and the depth of the patterns selected by post-processing are usually less important than the same statistics for all the mined patterns. This may come from the fact that deep patterns are not diverse enough to be selected by the post-processing step which in turn suggests that, when the timeout is reached, the diversity of the mined pattern is not high enough. To further increase this diversity, stochastic search methods such as Monte-Carlo Tree Search [2] could be integrated in our algorithm.

5 Conclusion and future work

We have presented GriMA2D+t, an algorithm to mine temporal sequences of 2D regular structures called grids. We have shown on experiments on a classical cellular automaton, the game of life, that GriMA2D+t can effectively extract spatio-temporal patterns in temporal grids. We have also shown that those patterns can be used as new features for classification algorithms and, in particular, to successfully predict the outcomes of cellular automata. This opens interesting new paths in the automatic analysis of the evolution of ecosystems and, in particular, to predict and control spatio-temporal spread of species in order to preserve biodiversity.

To further increase the efficiency of the classification process and scale to larger problems, we proposed to use a post-processing step that allows us to select a good subset of the mined patterns. In future work, we planned to directly mined a relevant subset of the possible patterns by using Monte-Carlo tree search methods. We also plan to apply this algorithm to analyze other temporal structures such as videos.

Acknowledgements

This work has been supported by the ANR project SoLStiCe (ANR-13-BS02-0002-01)

References

1. H. Arimura, T. Uno, and S. Shimozono. Time and space efficient discovery of maximal geometric graphs. In *Int. Conf. on Discovery Science*, pages 42–55, 2007.
2. G. Bosc, C. Raïssi, J.-F. Boulicaut, and M. Kaytoue. Any-time diverse subgroup discovery with monte carlo tree search. *CoRR*, 2016.
3. Broder Breckling, Guy Pe'er, and Yiannis G. Matsinos. *Cellular Automata in Ecological Modelling*, pages 105–117. Springer Berlin Heidelberg, 2011.
4. C.-C Chang and C.-Jen Lin. LIBSVM: A library for support vector machines. *ACM-TIST*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.
5. J. Conway. The game of life. *Scientific American*, 223(4):4, 1970.
6. R. Deville, É. Fromont, B. Jeudy, and C. Solnon. Grima: a grid mining algorithm for bag-of-grid-based classification. In *S+SSPR*, pages 132–142, 2016.
7. B. Fernando, É. Fromont, and T. Tuytelaars. Mining mid-level features for image classification. *IJCV*, 108(3):186–203, 2014.
8. C. Jiang, F. Coenen, and M. Zito. A survey of frequent subgraph mining algorithms. *KER*, 28:75–105, 2013.
9. Diana E Marco, Sergio A Páez, and Sergio A Cannas. Species invasiveness in biological invasions: a modelling approach. *Biological Invasions*, 4(1):193–205, 2002.
10. F. Odone, A. Barla, and A. Verri. Building kernels from binary strings for image matching. *IEEE-TIP*, 14(2):169–180, 2005.
11. A. Prado, B. Jeudy, É. Fromont, and F. Diot. Mining spatiotemporal patterns in dynamic plane graphs. *IDA*, 17:71–92, 2013.
12. Stephen Wolfram. Cellular automata as models of complexity. *Nature*, 311(5985):419–424, 1984.
13. J Timothy Wootton. Local interactions predict large-scale pattern in empirically derived cellular automata. *Nature*, 413(6858):841–844, 2001.
14. X. Yan and J. Han. gSpan: graph-based substructure pattern mining. In *ICDM*, pages 721–724, 2002.