sciendo
PU. M. A. Vol. **30** (2022), No.1, pp. 106 – 112

On the exhaustive generation of discrete figures with connectivity constraints

Hugo Tremblay
Department of Computer Science and Mathematics
Université du Québec à Chicoutimi
Canada
email: hugo_tremblay2@uqac.ca

and
Julien Vernay
Université de Lyon
UJM-Saint-Étienne
France
and
Department of Computer Science and Mathematics
Université du Québec à Chicoutimi
Canada
email: julien.vernay1@uqac.ca

(Received: March 31, 2022, and in revised form May 15, 2022.)

Abstract. This paper deals with a generalization of polyominoes called (a, b)-*connected discrete figures*, where a and b respectively denotes the connectivity of the foreground (i.e. *black pixels*) and background (i.e. *white pixels*). Formally, a finite set of pixels P is (a, b)-connected if P is a-connected and $\overline{P}$ is b-connected. By adapting a combinatorial structure enumeration algorithm due to Martin, we successfully generate (a, b)-connected discrete figures up to size $n = 18$.

Mathematics Subject Classification(2020). 05A15, 05B50.

Keywords: algorithm, exhaustive generation, discrete figure, digital geometry.

1 Introduction

Since their introduction by Golomb in [3], *Polyominoes*, that is finite sets of connected pixels, have proven to be of particular interest with many applications ranging from crystallography [10], robotics [6] and signal processing [2], among others. They appear in the literature under various monikers: animals [8, 7], clusters [9], polyominoes and pseudo-polyominoes [4] and self-avoiding polygons [5]. However we may call them, the problem of enumerating such discrete figures remains at the core of several research interests. This is not without reason as the efficient enumeration of polyominoes is still an unsolved problem. Let G_4 (respectively G_8) be the graph whose vertices is the set $\mathbb{Z}^2$ of pixels and where two pixels are adjacent if they share a side (respectively, a side or a vertex). A polyomino is then a finite subgraph of G_4 or G_8.

DOI: 10.2478/puma-2022-0016
© 2022 Tremblay, Vernay. This is an open access article licensed under the Creative Commons Attribution-NonCommercial-NoDerivs License (http://creativecommons.org/licenses/by-nc-nd/3.0/).

In this paper, we concern ourselves with a generalization of polyominoes called (a, b)-*connected discrete figures*, where a and b respectively denotes the connectivity of the foreground (i.e. *black pixels*) and background (i.e. *white pixels*). Such objects were first introduced in [1] in order to study a digital image transformation problem. Formally, a discrete figure $P \subset \mathbb{Z}^2$ is (a, b)-connected if and only if P is a-connected and $\overline{P}$ is b-connected. We refer to 4-connected (resp. 8-connected) figures as $(4, 0)$-connected (resp. $(8, 0)$-connected) and we denote $S_{a,b}$ the set of (a, b)-connected discrete figures. It is easy to deduce that $S_{a,4} \subset S_{a,8} \subset S_{a,0}$ and $S_{4,b} \subset S_{8,b}$ for $a \in \{4, 8\}$ and $b \in \{0, 4, 8\}$.

Some families $S_{a,b}$ have been considered in the literature: $S_{4,0}$ and $S_{8,0}$ are respectively polyominoes and pseudo polyominoes [4]. Also, $(4, 4)$-connected figures are frequently called *self-avoiding polygons* due to the fact that they can be generated by a looping self-avoiding walk on the grid $\mathbb{Z}^2$ [5]. To the best of our knowledge, $(4, 8)$, $(8, 4)$ and $(8, 8)$-connected figures have never been previously studied.

In [8], J. L. Martin introduces an algorithm for the enumeration of lattice graph structures by considering a canonical ordering on the neighbourhood of each vertex. This paper presents a modified version of this algorithm by considering the connectivity of both P and $\overline{P}$ in order to generate (a, b)-connected figures. To the best of our knowledge, this paper constitutes the first enumeration of $(4, 8)$, $(8, 4)$ and $(8, 8)$-connected discrete figures. The enumeration of $(8, 0)$-connected figures up to size 18 is also a novel result. The application of Martin's algorithm also yields new perspectives for the algorithmic complexity for generating $(4, 4)$, $(4, 8)$ and $(8, 4)$-connected polyominoes. Finally, we provide a modern implementation of Martin's Algorithm in C++.

2 Generating (a, b)-connected figures

This section presents a modified version of Martin's enumeration algorithm for generating (a, b)-connected discrete figures. It proceeds by iteratively adding pixels to a figure until a prescribed size is attained. At each step, we ensure that both a and b connexities for black and white pixels are maintained. This is done in constant time for black pixels since it suffices to select the next pixel among the neighbourhood of the pixels of P. The same approach also yields constant time methods for checking white connectivity for the cases $(4, 4)$, $(4, 8)$ and $(8, 4)$. However, checking white-connectivity for $(8, 8)$-figures is linear in the size of the figure since paths may intersect without having any pixel in common.

Now, iteratively adding pixels one at a time defines an ordering on the pixels of P. Since this ordering is not unique (e.g. there are four ways of constructing the 2 by 2 square polyomino, starting at the bottom left pixel), this process potentially produces duplicates. The idea behind Martin's algorithm is to instead explicitly define a unique ordering on the pixels of P. Consequently, a figure is constructed by considering the so-called *canonical ordering* of its pixels. The following Algorithm 1 returns a canonical ordering of the pixels of a given figure F. The resulting ordering is illustrated in Figure 1.

Remark that this ordering is unique up to the choice of a starting pixel as well as an ordering on the neighbours of the points in the lattice graph.

Using the canonical ordering, we modify Martin's algorithm in order to generate (a, b)-connected discrete figures. The algorithm is based on a stack of size n containing *levels*. A level L_i is a list of *candidates*, that is pixels along with a certain state. Formally, a candidate is a triplet (p, k, s) where $p \in \mathbb{Z}^2$ is the coordinate of the pixel, $k \in \mathbb{N}$ is the level where the candidates first appear

Algorithm 1 CanonicalOrder

$p_1 \leftarrow \text{StartPoint}(F)$ ▷ p_1 is chosen according to a non-ambiguous criterion
$V \leftarrow \{p_1\}$ ▷ V is the set of visited pixels
$i \leftarrow 1$
$count \leftarrow 1$
while $count < n$ **do** ▷ We visit all n pixels
  for *direction* in ChosenOrder **do**
    ▷ For example, counterclockwise starting from the east
    $p' \leftarrow p_i + direction$ ▷ "+" denotes the vector sum
    if $p' \in F$ AND $p' \notin V$ **then**
      $count \leftarrow count + 1$
      $p_{count} \leftarrow p'$
      $V \leftarrow V \cup \{p'\}$
    end if
  end for
  $i \leftarrow i + 1$
end while
return $(p_i)_{1 \leq i \leq n}$ ▷ p_i is the ordered list of the pixels of F

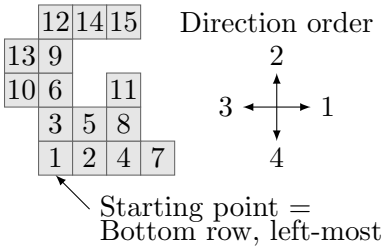


Figure 1: Example of canonical ordering using the left-most point on the bottom row as a starting pixel along with the cyclic order right-up-left-down.

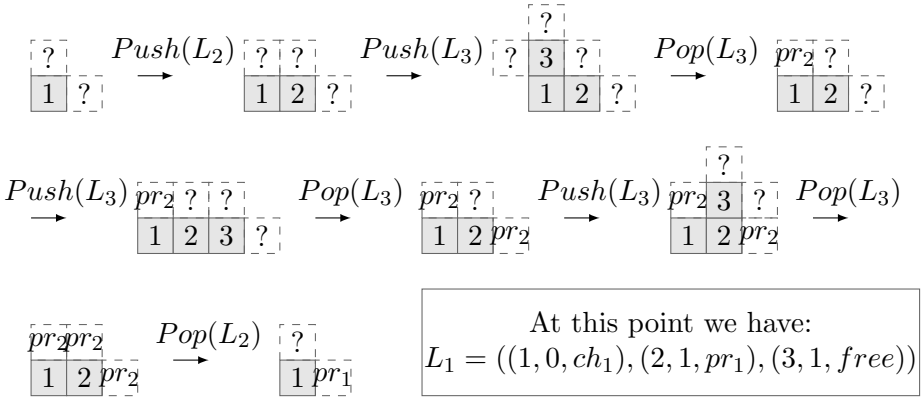


Figure 2: First steps of Martin's algorithm while generating 4-connected figures of size 3, with the canonical ordering of Figure 1. Free pixels are marked as "?" and prohibited pixels at level i are marked as "pr_i".

and $s \in \{free, ch_i, pr_i\}$ is the state of the candidate. Here, $free$ means the candidate has not yet been considered, ch_i means the candidate has been chosen at level i and pr_i means the candidate is prohibited at level i. As a first step, we set $L_0 = ((p_0, 0, free))$ where p_0 is the starting pixel. We now describe the two operations on the stack of levels. Remark that candidates are always added to levels according to the canonical ordering.

(i) Push level L_{i+1}: This operation first adds the next free candidate of L_i to L_{i+1} and sets its state to ch_{i+1}. Then, its neighbours are added to L_{i+1} in the predefined canonical direction order, setting $k = i + 1$ and the state to $free$.

(ii) Pop level L_{i+1}: First, the state of all prohibited candidates at level $i + 1$ is changed to $free$. Next, all candidates for which $k = i + 1$ are removed from the level. Finally, the candidate chosen at level $i + 1$ is marked as pr_i.

By repeatedly alternating $Pop(L_i)$ and $Push(L_i)$ stack operations, one can output all discrete figures of size i. Figure 2 shows the first steps of Martin's algorithm.

3 Implementation and results

S. Redner proposed in 1982 a FORTRAN implementation of Martin's algorithm for the generation of 4-connected figures without considering white connectivity [9]. We propose a thoroughly documented and modern C++ implementation of Martin's algorithm, freely available on GitHub [11]. The implementation has two versions: a simpler one which is a direct implementation of the notions presented in Section 2 and a more optimized one which makes use of multithreading.

In addition to the Push and Pop operations described in the previous section, we define additional operations on the stack of levels so we are able to access each figure up to size n, including the intermediate smaller ones: (a) adding the last chosen point's neighbours as candidates, (b) adding the next free candidate to the figure, and (c) replacing the last chosen point by the next free candidate.

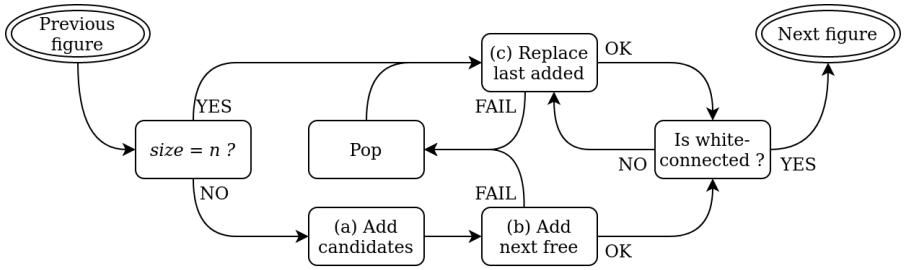
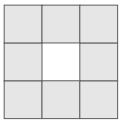
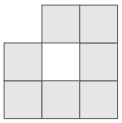


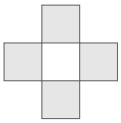
Figure 3: Flowchart of Martin's algorithm for generating (a, b)-connected figures. (a) The neighbours of the last chosen point are added as candidates. (b) The next free candidate is added to the figure provided it exists. (c) The last chosen point is replaced by the next free candidate provided it exists.



(a) $P \in S_{4,0}$ but $P \notin S_{4,8}$



(b) $P \in S_{4,8}$ but $P \notin S_{4,4}$



(c) $P \in S_{8,8}$ but $P \notin S_{8,4}$

Figure 4: Examples of discrete figures explaining the difference in number between the families $S_{a,b}$.

(a) followed by (b) is equivalent to a single Push operation and (c) is equivalent to a Pop operation followed by a Push operation. Note that operation (a) may add no candidates (for example in a snail-like figure) and operations (b) and (c) may fail due to a lack of non-explored free candidates. Doing a Pop operation while the current figure's size is one marks the enumeration's end. The flowchart in Figure 3 illustrates the process of generating a new figure.

Checking the white connectivity in the case $(8, 8)$ requires to look at the entire figure since a black closed path does not necessarily partition the white pixels, thus it is linear in time. For the other connectivities, we only check in the candidate's neighbourhood whether it closes a black path, thus it is in constant time.

We implemented both versions of Martin's algorithm in C++. The results for the number of figures of each connectivity type are in Table 1. It took several hours on a laptop PC to generate each column. Also, Figure 4 presents some examples of figures explaining the difference for the cardinality of the various families $S_{a,b}$.

4 Conclusion

This paper deals with (a, b)-connected discrete figures, that is finite sets of a-connected pixels such that the background is b-connected. By modifying Martin's enumeration algorithm, we generate all such figures as well as their number up to size 18 for most families $S_{a,b}$. Due to a lack of space, many details are omitted including proofs for the complexity of checking black and white connectivity at each step. We also considered different random versions of Martin's algorithm and plan to tackle this aspect in future publications.

This work constitutes a first step towards understanding this type of combinatorial objects and

n	$(4,0)$-connected	$(4,8)$-connected	$(4,4)$-connected	$(8,0)$-connected	$(8,8)$-connected	$(8,4)$-connected
1	1	1	1	1	1	1
2	2	2	2	4	4	4
3	6	6	6	20	20	20
4	19	19	19	110	110	109
5	63	63	63	638	638	622
6	216	216	216	3832	3832	3664
7	760	760	756	23592	23592	22094
8	2725	2724	2684	147941	147940	135609
9	9910	9898	9638	940982	940966	843941
10	36446	36358	34930	6053180	6053002	5310754
11	135268	134744	127560	39299408	39297724	33724862
12	505861	503065	468837	257105146	257105146	215793158
13	1903890	1889936	1732702	1692931066	1692811056	1389673091
14	7204874	7138286	6434322	11208974860	11208021976	8998648488
15	27394666	27086832	23993874	74570549714	74563162152	58548155506
16	104592937	103202581	89805691	498174818986	498118512909	382526638033
17	400795844	394625770	337237337	3340366308393		2508473632910
18	1540820542	1513810138	1270123530	22471158811164		16503616943998

Table 1: Number of (a,b)-connected figures. The results for $(4,0)$ and $(4,4)$ respectively corresponds to OEIS sequence #A001168 for the number of fixed 4-connected polyominoes and OEIS sequence #A006724 for the number of self-avoiding polygons. The result for $(8,0)$ corresponds to OEIS sequence #A006770 for the number of fixed 8-connected polyominoes, extended here to $n = 18$ for the first time. Generating $(8,8)$-connected figures of size $n \geq 17$ would have taken too much time on our computer.

opens up several new interesting research avenues. From a theoretical point of view, it would be interesting to find generating series for (a,b)-connected figures. Also, since they originated from a digital picture transformation problem in [1], the work done in this paper could help solve the following conjecture: Given two $(4,4)$-connected digital images I and I' of size n, there exists a sequence of $O(n^2)$ 8-adjacent pixel interchanges that transforms I into I'. A possible starting point for tackling this problem would be to study the density of (a,b)-connected discrete figures and its correlation to the preceding transformation. Recall that the *density* of a simple graph G on n vertices, and by extension a discrete figure, effectively measures how close G is to the complete graph K_n. It is relatively straightforward to show that the density tends to zero as n grows.

From an experimental point of view, it would be interesting to refine our implementation, namely by using a more powerful computer or by dispatching work among multiple computers. Experimenting with various segmentation values or probabilities would also surely lead to interesting results, both for the random and the exhaustive generation of discrete figures. In particular, the computation of the number of $(8,8)$-connected discrete figures of size 17 and 18 could be easily and quickly obtained by using a more powerful computer.

References

[1] P. Bose, V. Dujmovic, F. Hurtado and P. Morin, *Connectivity-preserving transformations of binary images*, Comput. Vis. Image Underst., 113 (2009) 1027–1038.

[2] Z. M. Du, F. Y. Ye, H. Shi and G. P. Zhu, *A fast recovery method of 2D geometric compressed sensing signal*, Circ. Syst. Signal Pr., 34 (2015) 1711–1724.

[3] S. W. Golomb, *Polyominoes*, Charles Scribners' Sons, 1965.

[4] S. W. Golomb, *Polyominoes: Puzzles, Patterns, Problems, and Packings*, Princeton University Press, 1994.

[5] I. Jensen and A. J. Guttmann, *Self-avoiding polygons on the square lattice*, J. Phys. A, 32 (1999) 4867–4876.

[6] M. Kobilarov, M. Desbrun, J. Marsden and G. Sukhatme, *A discrete geometric optimal control framework for systems with symmetries*, Proceedings of Robotics: Science and Systems (2007) Atlanta, GA, USA, June.

[7] P. M. Lam, *On Monte Carlo generation and study of anisotropy of lattice animals*, J. Phys. A, 19 (1986) L155.

[8] J. L. Martin, *Computer techniques for evaluating lattice constants*, in: Domb, C., Green, M. S. (Eds.), Phase Transitions and Critical Phenomena, vol. 3, Academic Press, London, 1974, pp. 97—112.

[9] S. J. Redner, *A Fortran program for cluster enumeration*, J. Stat. Phys., 29 (1982) 309–315.

[10] T. Sunada, *Topological Crystallography*, Springer, 2013.

[11] J. Vernay, *GitHub Repository: discrete-figures*, version 1.1.0 (2022), https://github.com/J-Vernay/discrete-figures.