On the Exhaustive Generation of k-Convex Polyominoes

Stefano Brocchi and Giusi Castiglione and Paolo Massazza

Abstract

We present an algorithm to compute the degree of convexity of a convex polyomino P, defined as the smallest integer k such that any two cells of P can be joined by a path in P with at most k changes of direction. The algorithm is used to generate in amortized time $O(1)$ all k-convex polyominoes of size n.

1 Introduction

A polyomino (cf. [1]) of size n is a finite and connected union of n unitary squares (called cells) in the plane $Z \times Z$, considered up to translations. The class of *hv-convex* (or, more simply, convex) *polyominoes* is formed by polyominoes whose intersection with any vertical or horizontal line is connected. Such a class has been studied under several points of view as, for instance, the enumeration and the exhaustive generation [2,3,4]. In [5] the authors introduce a parameter related to the shape of a convex polyomino P: the *degree of convexity* of P is defined as the smallest number of changes of direction in a path internal to P needed to connect any pair of its cells. P is said to be *k-convex* if it has a degree of convexity at most k. An asymptotic estimate of a lower bound for the number of k-convex polyominoes with perimeter p has been given in [6]. Moreover, in [7,8] some subclasses of k-convex polyominoes has been enumerated. The simplest convex polyominoes have degree of convexity 1 and are called L-convex polyominoes; they have been defined in [5] and widely investigated in literature (cf. [9,10,11]). Higher in the hierarchy we have the Z-convex polyominoes with degree of convexity 2, for which some results have been presented in [12,13,14]. In [15] the authors give an efficient algorithm to determine the degree of convexity k of a convex polyomino P in time $O(\min(m, r*k))$, where m is the number of columns of P and r the number of its corners, extending a method originally used in [16]. Here, we define an alternative algorithm for determining k, having a computational complexity $O(m)$ but with some useful properties; in particular, it allows to generate a polyomino, column by column, by checking the degree of convexity reached at each step. Thanks to this procedure, we are able to define a CAT (Constant Amortized Time) algorithm for the exhaustive generation,

This paper is electronically published in
Electronic Notes in Theoretical Computer Science
URL: `www.elsevier.nl/locate/entcs`

by area, of the family of k-convex polyominoes.

2 Notation and preliminaries

Let P be a convex polyomino with a minimal bounding rectangle of size $m \times n$. We number the m columns and the n rows from left to right and from bottom to top, respectively. Thus, we consider the bottom (resp. top) row of P as its first (resp. last) row, and the leftmost (resp. rightmost) column of P as its first (resp. last) column. By (i, j) we denote a cell in the i-th row and j-th column of P. A *path* connecting two cells a and b of P, is a sequence $(a = (i_1, j_1),(i_2, j_2),\dots ,(i_r, j_r) = b)$ of distinct edge-adjacent cells all belonging to P. The step $((i_s, j_s),(i_{s+1}, j_{s+1}))$ is called an *East* step if $j_{s+1} = j_s + 1$ and $i_{s+1} = i_s$, a *North* step if $i_{s+1} = i_s + 1$ and $j_{s+1} = j_s$, a *West* step if $j_{s+1} = j_s - 1$ and $i_{s+1} = i_s$, a *South* step if $i_{s+1} = i_s - 1$ and $j_{s+1} = j_s$. A path is *monotone* if it is made of steps in only two directions: North and East (NE-path), North and West (NW-path), South and East (SE-path) or South and West (SW-path). A polyomino is convex if and only if every pair of its cells is joined by a monotone path (see [5]). Given $k \in \mathbb{N}$, a convex polyomino is said to be k-*convex* if every pair of its cells can be joined by a monotone path with at most k changes of direction. Obviously, a k-convex polyomino is also h-convex for every $h > k$. We define the *degree of convexity* of a convex polyomino P as the smallest $k \in \mathbb{N}$ such that P is k-convex. Let j be a column of P, by $\mathrm{low}(j)$ and $\mathrm{high}(j)$ we denote the row index of the bottom cell and of the top cell of j, respectively.

Furthermore, by $\mathrm{left}(i)$ we denote the column index of the leftmost cell of row i of P. We say that column i *includes* column j iff $low(i) \le low(j) \le high(j) \le high(i)$. Given a convex polyomino P with l columns, let h be the largest integer such that column h includes all preceding columns, and let k be the smallest value greater than h such that column k is not included in column h. If such a k does not exist $(k = \perp)$ then P is necessarily the union of a left stack L (the first h columns) with a right stack R (columns from $h + 1$ to l). Otherwise, P is called *descending* (*ascending*) if $\mathrm{low}(h) > \mathrm{low}(k)$ ($\mathrm{high}(k) > \mathrm{high}(h)$) and it can be decomposed as $L \cdot C \cdot R$ where L and R are stacks and C is a parallelogram polyomino. Notice that R consists of the last $l - j$ columns where j is the smallest index such that column j includes all subsequent columns. W.l.o.g., from here on we deal only with descending polyominoes. A cell $(i, j) \in P$ is a *SW-corner* (*SE-corner*) of P if and only if $(i - 1, j), (i, j - 1) \notin P$ $((i - 1, j), (i, j + 1) \notin P)$. Analogously, $(i, j) \in P$ is a *NE-corner* (*NW-corner*) of P if and only if $(i + 1, j), (i, j + 1) \notin P$ $((i+1, j), (i, j-1) \notin P)$. The following theorem states that in order to determine the degree of convexity of a descending polyomino we can restrict ourselves to consider suitable paths between corners.

Theorem 2.1 *(cf. [15]) A descending convex polyomino P has degree of convexity k if and only if for any NW-corner a and for any SE-corner b there is a SE-path joining a to b (or, equivalently, a NW-path joining b to a) with at most k changes of directions, all occurring on the boundary of P.*

3 Computing the degree of convexity

In this section we give an algorithm for the computation of the degree of convexity. The algorithm is based on Theorem 2.1 and takes advantage of the following lemmas.

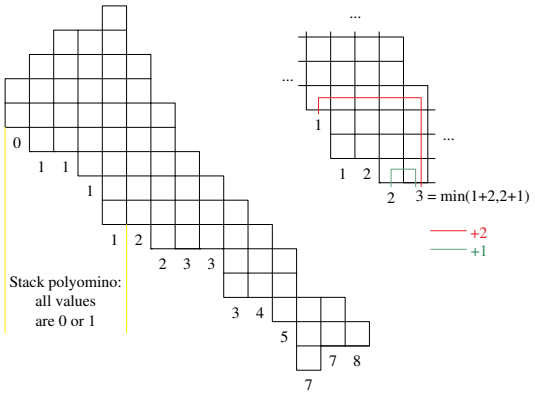

Fig. 1. Determining the SE-degree of convexity of P. Upper right: a depiction of the recursive rule.

To enhance the exposition, we first define the degree of convexity of a single column.

Definition 3.1 The NW-degree of convexity $\mathrm{Deg}(j)$ of column j of P, is the minimal number of changes of direction of a NW-path connecting the bottom cell of j to any NW-corner in the first j columns of P.

Notice that in a stack polyomino (i.e. a polyomino where column i is always included in column $i+1$) the NW-degree of convexity of any column is either 0 or 1. Let $P = L \cdot C \cdot R$ the decomposition of P defined in previous section, where C and R could be empty. The, one has:

Lemma 3.2 *Let P be a descending polyomino. For each column j of $C \cdot R$ with $low(j) < low(j-1)$ one has $Deg(j) = Deg(left(high(j)) + 2$.*

Lemma 3.3 *Let P be a descending polyomino. For each column j of $C \cdot R$ we have*

$$Deg(j) = \min(Deg(left(high(j))) + 2, Deg(left(low(j)) + 1)).$$

Let us consider the polyomino in Fig.1. Trivially, $Deg(1) = 0$. Each column j of L has $Deg(j) = 0$ if $low(j) = low(j-1) = high(j-1)$, and $Deg(j) = 1$ otherwise. For each column j to the right of L we compute $\mathrm{Deg}(\mathrm{left}(\mathrm{high}(i)))$ (red line in the figure on the right), $\mathrm{Deg}(\mathrm{left}(\mathrm{low}(i)))$ (green line) and we choose

$$\mathrm{Deg}(j) = \min(\mathrm{Deg}(\mathrm{left}(\mathrm{high}(j))) + 2, \mathrm{Deg}(\mathrm{left}(\mathrm{low}(j)) + 1)),$$

by Lemma 3.3.

The previous lemmas directly lead to Algorithm 1 that runs in time $O(m)$.

Algorithm 1 Algorithm for the computation of the degree of convexity of descending polyominoes.

```
Deg = [0, ..., 0];
for i = 2 to columnNumber do
  if column i ∈ L then
    if low(i − 1) = high(i − 1) = low(i) then Deg(i) = 0; else Deg(i) = 1;
  else
    Deg(i) = Deg(left(high(i))) + 2;
    if low(i − 1) ≤ low(i) then Deg(i) = min(Deg(i), Deg(left(low(i))) + 1);
  end if
end for
return max(Deg);
```

4 Exhaustive generation

The algorithm presented in Section 3 can be modified to generate all (descending) k-convex polyominoes of size n by an inductive approach. We suppose that a k-convex polyomino P_i with i columns and size $r < n$ has been generated and we see how to generate all k-polyominoes with $i+1$ columns that extend P_i.

The algorithm has to determine all integer pairs (a, b) such that by adding to P_i an $(i+1)$th column of size a with a bottom cell on row b, noted by $P_i|(a, b)$, one obtains another k-convex polyomino P_{i+1} Then, for each generated pair (a, b), it makes a recursive call to generate all $k-$convex polyominoes with prefix P_{i+1} and a suffix of size $n - r - a$. The algorithm can compute all of these pairs for a given P_i thanks to the following lemma:

Lemma 4.1 *Let P be a descending k-convex polyomino, and assume that for some integers (a, b) the polyomino $P|(a, b)$ is not k-convex. Then all polyominoes $P|(a_1, b_1)$ with $a_1 > a$ or with $a_1 = a$ and $b_1 > b$ are not k-convex.*

Lemma 4.1 leads to an iterator that, having as input P, generates (in sequence) all the pairs (a, b) such that $P|(a, b)$ is k-convex. It starts with the pair $(1, \hat{b})$ where $\hat{b}$ is the row of the top cell of the rightmost column of P. Then, if $P|(a, b)$ is k-convex the next pair is either $(a, b-1)$ (if $P|(a, b-1)$ is k-convex) or $(a+1, \hat{b}-a+1)$ (if $P|(a+1, \hat{b}-a+1)$ is k-convex). If both $P|(a, b-1)$ and $P|(a+1, \hat{b}-a+1)$ are not k-convex the iterator halts (all the pairs have been generated).

Therefore, at this point it is quite immediate to design a recursive algorithm that generates all k-convex polyominoes column by column. We first generate all left stacks of size not greater than n (these objects can be trivially generated in CAT time). Then, for each left stack, the algorithm proceeds by (recursively) adding columns which preserve k-convexity, using the iterator for the pairs (a, b). The execution of such an algorithm corresponds to a tree where each internal node at level i is associated with a suitable k-convex polyomino with i columns, say P_i, and has as many children as pairs (a, b) such that $P|(a, b)$ is k-convex with size at most n. The leaves of the tree correspond to k-convex polyominoes of size n, and we may prove that the algorithm runs in constant amortized time as the iterator runs in time $O(1)$ and each internal node has at least two children.

5 Conclusions and further work

We briefly introduced an algorithm for generating k-convex polyominoes by area, based on an algorithm for determining the degree of convexity k. There are many opportunities for further research arising from this work. An extension of the algorithm that we omitted for brevity's sake can generate polyominoes that are *exactly* k-convex, i.e. k-convex but not $(k-1)$-convex. Moreover, the results presented here might help to find some formulas for the enumeration of k-convex polyominoes. We plan to examine these research lines in future works.

References

[1] S. W. Golomb. Checker boards and polyominoes. *American Mathematical Monthly*, 61:675–682, 1954.

[2] M.P. Delest and G. Viennot. Algebraic languages and polyominoes enumeration. *TCS*, 34(1-2):169–206, 1984.

[3] M. Bousquet-Mélou. A method for the enumeration of various classes of column-convex polygons. *Discrete Mathematics*, 154(1-3):1–25, 1996.

[4] A. Del Lungo, E. Duchi, A. Frosini, and S. Rinaldi. On the generation and enumeration of some classes of convex polyominoes. *Electr. J. Comb.*, 11(1), 2004.

[5] G. Castiglione and A. Restivo. Reconstruction of l-convex polyominoes. *Electronic Notes in Discrete Mathematics*, 12:290–301, 2003.

[6] A. Micheli and D. Rossin. Counting k-convex polyominoes. *Electr. J. Comb.*, 20(2), 2013.

[7] D. Battaglino, J.Fedou, S. Rinaldi, and S. Socci. The number of k-parallelogram polyominoes. In *FPSAC*, volume AS of *Dicrete Mathematics and Theoretical Computer Science*, pages 1143–1154. Springer, 2013.

[8] A. Boussicault, S. Rinaldi, and S. Socci. The number of directed k-convex polyominoes. In *FPSAC*, volume AS of *Discrete Mathematics and Theoretical Computer Science*, pages 511–522. Springer, 2015.

[9] G. Castiglione, A. Frosini, E. Munarini, A. Restivo, and S. Rinaldi. Combinatorial aspects of *l*-convex polyominoes. *Eur. J. Comb.*, 28(6):1724–1741, 2007.

[10] G. Castiglione, A. Frosini, A. Restivo, and S. Rinaldi. A tomographical characterization of *l*-convex polyominoes. In *DGCI05*, volume 3429 of *LNCS*, pages 115–125. Springer, 2005.

[11] P. Massazza. On the generation of *l*-convex polyominoes. In *GasCom12, Bordeaux June 25-27*, 2012.

[12] G. Castiglione and P. Massazza. An efficient algorithm for the generation of *z*-convex polyominoes. In *IWCIA 2014*, volume 8466 of *Lecture Notes in Comput. Sci.*, pages 51–61. Springer, 2014.

[13] E. Duchi, S. Rinaldi, and G. Schaeffer. The number of *z*-convex polyominoes. *Advances in Applied Mathematics*, 40(1):54 – 72, 2008.

[14] A. Frosini, R. Pinzani, S. Rinaldi, and L. Vuillon. Tomographical aspects of 2-convex polyominoes. In *CET2011*, Proceedings of world congress on engineering and technology. IEEE press, 2011.

[15] S. Brocchi, G. Castiglione, and P. Massazza. On computing the degree of convexity of polyominoes. *Electron. J. Combin.*, 22(1):1–13, 2015.

[16] S. Brocchi, A. Frosini, R. Pinzani, and S. Rinaldi. A tiling system for the class of *l*-convex polyominoes. *Theor. Comput. Sci.*, 475:73–81, 2013.