NEW UPPER BOUND FOR THE CONNECTIVE CONSTANT FOR SQUARE-LATTICE SELF-AVOIDING WALKS

Olivier Couronné

NEW UPPER BOUND FOR THE CONNECTIVE CONSTANT FOR SQUARE-LATTICE SELF-AVOIDING WALKS

OLIVIER COURONNÉ

Abstract. By modifying the automaton used by Pönitz and Tittman [4], and considering loops of length up to 26, we obtain 2.662343 as an upper bound for the connective constant in the lattice $\mathbb{Z}^2$.

1. Introduction

Let the number of walks on $\mathbb{Z}^2$ of length n starting at the origin and that never visit twice the same vertex be denoted as $c_2(n)$. As $c_2(n+m) \leq c_2(n)c_2(m)$, the quantity $\ln(c_2(n))/n$ converges to a finite limit, called the connective constant and denoted μ_2. There are three standard questions concerning μ_2: obtaining a good estimate, an upper bound and a lower bound. At present μ_2 is estimated to 2.63815853032790 [3], the best upper bound [4] is 2.679192495, and the current lower bound [1] is 2.62002.

In this article, we modify the automaton used in [4] and, with of course the help of more powerful computers than in 2000 which allows us to consider loops of length up to 26 instead of 22, we obtain

Theorem. *The connective constant for the lattice $\mathbb{Z}^2$ verifies*

$$\mu_2 \leq 2.662342426.$$

This article is organized as follows. In section 2 we describe the technique used in [4], with the paths without loops up to a certain size and the corresponding automaton. We also precise some definitions and notations. In section 3, we expose our main improvement, which consists of considering multiple ways to shorten too big states. In section 4, we give on one hand two improvements based on the planarity of $\mathbb{Z}^2$, and on the other hand two cases where we allow the states to be longer. These increase the number of states considered, but they are efficient considering the upper bound on μ_2 obtained. The algorithm is explained in section 5 and the results are given in section 6.

2. Sketch of the initial method

We recall here the automaton used in [4]. In order to get an upper bound for μ_2, the authors considered paths without cycles up to a certain length k. The cardinal of these paths of length n is of course greater than $c_2(n)$. Starting from $k = 14$ up to $k = 22$ (and note that there must be an even number of steps in a loop), they obtained upper bounds ranging from 2.8312 to 2.6792. We give now their automaton for $k = 4$.

We define state 1 as one step to the right. We can go in three directions: up, right or down.

Date: November 2022.

2010 *Mathematics Subject Classification.* 82B41; 05A15.

Key words and phrases. self-avoiding walk; connective constant.

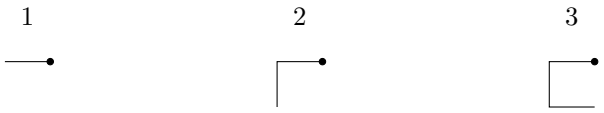
The author is supported by the Labex MME-DII funded by ANR, reference ANR-11-LBX-0023-01.

- When we go up, we obtain a new state, called state 2.
- When we go right, we are transferred back to state 1 again, as the first edge will be useless for a loop of size 4.
- When we go down, by symmetry, we identify the new state to state 2.

From the state 2 as it is represented below :

- When we go up, we are again in state 2.
- When we go right, we obtain the state 1.
- When we go down, we obtain a new state, called state 3.

From the state 3 there is two possibilities. One to state 1, and the other to state 2. So the automaton is composed of the three states represented here:



and the associated matrix is :

$$\begin{pmatrix} 1 & 2 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

By iteratively multiplying this matrix with the vector initially composed of ones, we approach [2] its largest eigenvalue, which gives 2.8312 as an upper bound for μ_2. Applying this method for $k = 22$, Pönitz and Tittman obtained 2.679192495 for the upper bound.

Similarly as in [4], we will use rotations and symmetry such that the last step of the state is to the right, and the first vertical step is downward. We shall call A the most recent vertex, and B the oldest. For a given state, we call its *size-loop* to be the number of vertex of the state plus the L_1 distance between A and B minus 1, that is the size of a loop using all the vertices of the state and a direct path between A and B (this path possibly using vertices of the state).

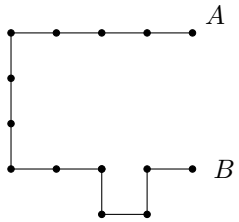
3. Simplifications of a state

Starting from a state, it is possible that the state obtained by adding a vertex in one of the three directions (up, right and down) is too big to make a loop of length k between A and B. In this section we present different ways to simplify such a state.

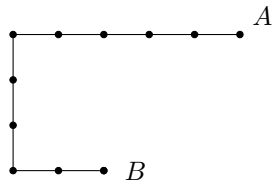
We stress that the techniques presented in this section do not change the set of states, but the relations between them and ultimately the transition matrix.

3.1. **Erasing the old vertices.** Initially, when a potential state was too big, we would forget the most ancient points until the state was not too big. This is the method used in [4].

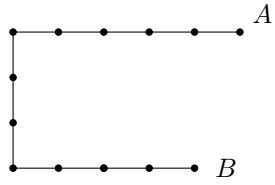
3.2. **Small bridges.** Consider $k = 16$ and the state S_1 represented below :



When we go right, with the initial procedure 3.1, we would get the state S_2 :



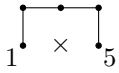
It seems unfortunate to lose vertices close to A. In fact, the rule we must observe for a child state is that the obtained state must not forbid paths that were previously allowed. In that respect, the following state S_3 can also be a right-child of S_1 :



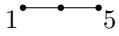
It is so because vertices of S_3 are included in the state obtained from S_1 to which we add a vertex at the right of A. Furthermore, the state S_3 is clearly a better choice than S_2 as S_3 contains S_2. The situation will not always be as clear, but nevertheless, each time the configuration 

is present, and that we call a *small bridge*, we can simplify it with a single edge between 1 and 4.

3.3. **Large bridges.** We call *large bridges* portions of a state like:

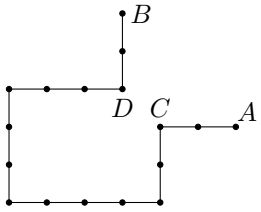


If this configuration is present in a state, where 1 and 5 are both not end-vertices of the state, and the × is not a vertex of the state, then we can simplify this portion to

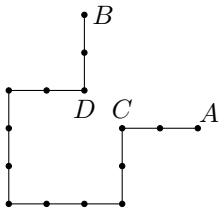


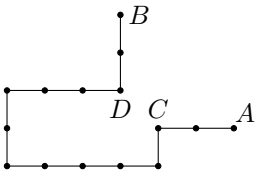
This transformation is valid due to the topological property of $\mathbb{Z}^2$, as a self-avoiding walk should not visit the ×. We could not do that for example in $\mathbb{Z}^3$.

3.4. **Small loops.** Consider the state S_1, which is supposed too big:



As C and D, which are not A, are such that $d_\infty(C, D) \leq 2$, we should never go in the area enclosed by the loop delimited by C and D. Hence the following states are valid simplifications of S_1 :





The procedure is as follows. Check if the state has what is called a small loop, that is two points C and D, whose indexes in the state are at least 9 apart, such that $d_\infty(C, D) \leq 2$, and both different A if the distance is 2 and not 1. If that is the case, and that A is not inside the loop defined by C and D, visit the vertices from C to D, and verify if there exists a straight line of length at least 3 surrounded by two corners in the same orientation as the loop $[C, D]$. For each such line found, we can propose a simplification as shown above.

In order to not slow too much the algorithm, for A not being inside the loop, we simply verify if there is at least two straight lines starting from A and not intersecting any vertex of the state.

3.5. **Multiple choices for a child.** To sum up, when a child state is too big, we have the following choices :

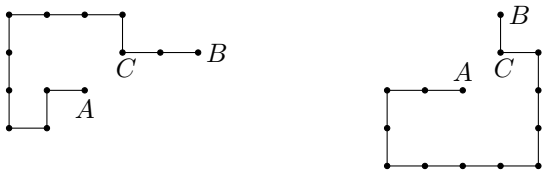
- forget the most ancient points until the state is not too big
- a new child state possible for each small bridge
- a new child state possible for each large bridge, not containing the extreme vertices.
- a new child state possible for each simplification of a small loop.

4. Other Improvements

The first following two points give conditions to allow a step in a direction based on the planarity of the graph, the second one giving a notable improvement on the upper bound. The next two points increase the number of states, but in an efficient way.

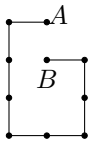
4.1. **Planar considerations for A.** When the vertex at the right of A is occupied, by topological consideration, we should not be able to go both upward and downward. Let C the vertex at the right of A. We count the algebraic number of corners, $+1$ to the right and -1 to the left, for the portion from C to A. If this is positive, we cannot go downwards, otherwise we cannot go upwards.

We implement similar considerations if the vertex at the top-right of A is occupied: in that case, either we cannot go upwards, or we cannot go both to the right and downwards:



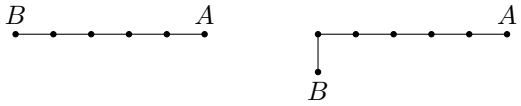
The situation where the vertex at the bottom-right of A is occupied is symmetric.

4.2. **Planar considerations for B.** Now that we do not systematically erase the oldest vertices, the following configuration can happen:



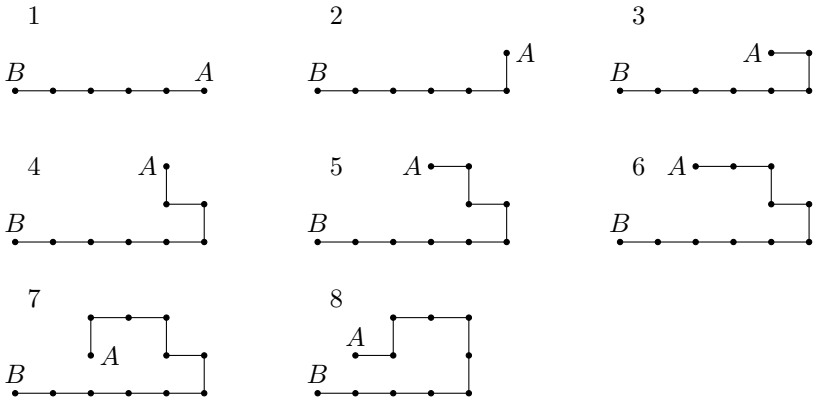
So at each step, we verify if the point B is not surrounded, that is if there can be an infinite path starting from B not using the vertices of the state. This verification gives a notable improvement on the result.

4.3. **States similar to a line.** When considering the eigenvector of the matrix, one can observe that the largest coordinates are for states that are nearly lines, such as:



With that in mind, we allow states, whose portion starting from B and of length $k/2$ is similar to a line, to be larger, that is the allowed size-loop is now $k+2$.

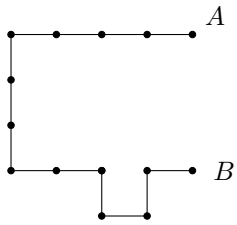
Furthermore, again for these states, while A goes toward B, but not too close to B, we allow an extra $+2$ for the length of the loop. Here is an example for $k = 10$, and the initial state the line with 6 vertices (we do not apply the rules about the first step to the right and the first vertical step downwards).



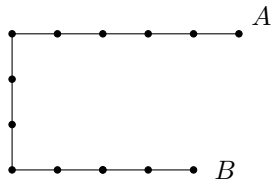
The state 1 is particular, since by definition its size-loop is 10, but we cannot make a loop of size less than 12. States 2 and 3 have a size-loop of 12, which is allowed since the portion starting from B is similar to a line. Then state 4 is of size-loop 14, which is also allowed since we always have a "line" starting from B, and A is not considered close to B. States 5, 6 and 7 are similar to the state 4. Finally when we go left from state 7, we consider that A is close to B, and we allow only a size-loop of 12. State 8 is one of the possible simplification. It can be of size-loop 12 as the criteria we take for a "line" starting from B is valid for this state.

This technique allows us to keep the former point B in more situations, and to consider longer loops for specific states.

4.4. **States lacking simplifications.** When we go right from the following state:

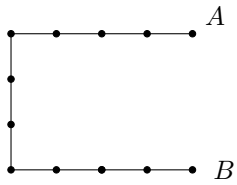


we can choose the following child:

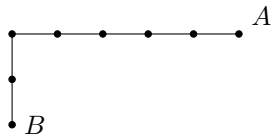


and that is totally fine (this is the example of 3.2).

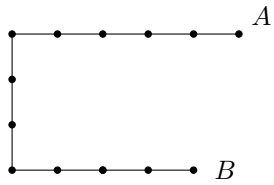
But when we go right from the following state S_1:



as there is no simplification, we have to choose



Remark that the states considered here are not concerned by the preceding point 4.3. Now, in order to improve the second situation, when a state has no simplification in the half close to B, no small loop, and that there is a direct path from A to B, we allow an extra $+2$ for the size of the loop. Hence a possible child of S_1 will be



4.5. **Iterative simplifications.** This is not *per se* an improvement, but something we have to take care of as a consequence of the two preceding techniques. It may happen that a state was allowed to be of size-loop $k+2$, but when we go right for example, then a candidate child, after one simplification, is also of size-loop $k+2$, without necessarily being allowed to be so. In that case, we create from this child a new list of simplified states, which will have a strictly lesser size-loop. We do this for each candidate child that is too large, and obtain in that way a correct list of children.

5. The Algorithm

First we generate the states without the relations. The initial state S_0 is the line of length $k/2$ (the technique in 4.3 makes this state relevant, even if it is "too big"). For each direction (up, right, down), we verify if the vertex is empty, and also the planar conditions 4.1 and 4.2. If the move is allowed, we verify if the potential state was already discovered or if it is not too big. In the other case, we create a list of potential children using the methods exposed in section 3. Note that with the first method of erasing the oldest vertices, we proceed gradually, and we verify after each deletion if the state has not been already discovered. So for each state and each direction, we have a list of allowed children. The new states of these lists are append to the list of the states to proceed. As claimed, in this first step, we do not save the relations between the states.

Moreover, we found somewhat beneficial, when creating the list of children in a direction for a state, to begin with a list without the improvements of 4.3 and 4.4, then to add the states found with these two improvements. It increases the number of states, but in an efficient way concerning the upper bound. Unfortunately, this significantly increases the duration of the algorithm, and we did not use it for $k = 26$.

After the first run, we put in memory the list of all discovered states, and we run again the preceding part, this time recording the list of children. The reason for doing these two steps instead of just one is that the states lately discovered using the improvements in 4.3 and 4.4 can be used as children by previous states.

In the next step, we create a transition matrix by choosing the first element of each non-empty list of children. We then approximate by iteration an eigenvector (we divide at each step the vector by its maximal value).

Once we have a first eigenvector, we create a new transition matrix, by choosing the best child, that is the child having the minimal corresponding coordinate in the eigenvector, for each list of children. We approximate again an eigenvector, and we iterate multiple times this part.

6. Results and Discussion

We list the upper bounds obtained for different values of k, together with the number of states and the size of the file containing the lists of children :

k	upper bound	number of states	size of the file containing the children
14	2.682775686	20,313	590 ko
16	2.677352271	95,637	2.93 Mo
18	2.673036298	486,798	15.6 Mo
20	2.669575008	2,533,177	76.3 Mo
22	2.666665240	13,731,499	427.5 Mo
24	2.664196283	79,510,267	2.64 Go
26	2.662342426	430,365,791	12.86 Go

For $k = 26$, the program took one week and necessitated a computer with 32 Go of RAM. For $k = 22$, the number of states is about 60% larger than it was in [4]. Even with this difference, one can see that the new method is efficient, the crucial part being the different ways to shorten a state.

Below we present, for $k = 18$, the results using some combinations of the techniques described previously. The column *Two passes* indicates if we build the graph structure in two steps as described at the beginning of the algorithm.

Similar to a line	States lacking simplifications	Two passes	upper bound	number of states
			2.678392579	255, 961
x			2.678121527	306, 605
	x		2.676625088	309, 224
x	x		2.675854723	446, 832
	x	x	2.674975842	315, 029
x	x	x	2.673435562	447, 250

We can see that the technique concerning states lacking simplification 4.4 is particularly efficient, and that calculating the children in two steps is mandatory. The improvement concerning states similar to a line 4.3, while not being as efficient as the other two, is still better than increasing the size of the loop by 2. The comparison between the last line of this table and the line for $k = 18$ in the first table illustrates the interest to build the list of children first without 4.3 and 4.4, and then with them.

To finish, we discuss briefly how the simplification techniques can apply in dimension three or greater. Among the different simplifications of states, only the usual deletion of old vertices and the simplification of small bridges apply. Furthermore, it would not be useful to ensure that, as in 4.1 and 4.2, the points A and B are not surrounded by vertices of the state since, for the lengths of loops considered, it hardly ever happens in dimension three, and never in greater dimensions.

References

[1] W.A Beyer and M.B Wells. Lower bound for the connective constant of a self-avoiding walk on a square lattice. *Journal of Combinatorial Theory, Series A*, 13(2):176–182, 1972.

[2] Françoise Chatelin. *Eigenvalues of matrices*, volume 71 of *Classics in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2012. With exercises by Mario Ahués and the author, Translated with additional material by Walter Ledermann, Revised reprint of the 1993 edition [MR1232655].

[3] Jesper Lykke Jacobsen, Christian R. Scullard, and Anthony J. Guttmann. On the growth constant for square-lattice self-avoiding walks. *J. Phys. A*, 49(49):494004, 18, 2016.

[4] A. Pönitz and P. Tittmann. Improved upper bounds for self-avoiding walks in z^d. *Electron. J. Combin.*, 7:Research Paper 21, 10 pp., 2000.

Université Paris Nanterre, Modal'X, FP2M, CNRS FR 2036, 200 avenue de la République 92000 Nanterre, France.

Email address: olivier.couronne@parisnanterre.fr