Séminaire Lotharingien de Combinatoire **77** (2018), Article B77h

THE SANDPILE MODEL ON $K_{m,n}$ AND THE RANK OF ITS CONFIGURATIONS

MICHELE D'ADDERIO AND YVAN LE BORGNE

Abstract. We present an algorithm to compute the rank of a configuration of the sandpile model for the complete bipartite graph $K_{m,n}$ of complexity $O(m+n)$. Furthermore, we provide a formula for the generating function of parking sorted configurations on complete bipartite graphs $K_{m,n}$ according to rank, degree, and the sizes m and n.

The results in the present paper are similar to those found in a previous paper by Cori and Le Borgne [*Electron. J. Combin.* **23**(1) (2016), Paper 1.31, 47 pp.] for the complete graph K_{n+1}, and they rely on the analysis of certain operators on the stable sorted configurations of $K_{m,n}$ developed by Aval, D'Adderio, Dukes and Le Borgne in [*Adv. Appl. Math.* **73** (2016), 59–98].

Contents

Introduction

The sandpile model was introduced in 1987 by the physicists Bak, Tang and Wiesenfeld as a first example of a dynamical system displaying the remarkable property of self-organized criticality [BTW] (see also [D1, D2]). Since then, this model as well as its several variants

(among which the notable *chip-firing game* [M]) has proved to be a fertile ground from which novel and intriguing results in mathematics have emerged (see e.g. [BLS, M]).

In 2007, Baker and Norine, in the celebrated article [BN], introduced the notion of rank of a configuration of the sandpile model on a graph, which yielded a surprising analogue of the classical Riemann–Roch theorem for divisors on curves. This interesting concept stimulated an increasing amount of research, not just in combinatorics, but even more in algebraic and tropical geometry (see e.g. [C, HKN] for some recent developments).

Unfortunately, computing the combinatorial rank directly from its very definition is non-trivial, even for small examples. Despite the attention that this notion has attracted since its appearance in [BN], the problem of deciding the complexity of its computation (first attributed to Hendrik Lenstra, cf. [HKN]) has been addressed only recently. Indeed, it has been shown in [KT] that computing the rank of a configuration on a general (in fact even Eulerian) graph is an NP-hard problem.

This situation motivates the search for efficient algorithms to compute the rank on some special classes of finite graphs.

One of the first investigations in this direction is contained in [CL], where an algorithm of linear complexity (in the number of vertices, if we assume constant time for any integer division) to compute the rank on complete graphs has been described. Moreover, in the same paper, several enumerative by-products have been provided, which turn out to have intrinsic combinatorial interest.

In the present article, we give several results about the rank of configurations on complete bipartite graphs, parallel to the ones proved in [CL] for complete graphs.

Given a connected simple graph $G = (V, E)$, where V is the set of its vertices, a *configuration* on G (a *divisor* in the terminology of [BN]) is simply a function $u : V \to \mathbb{Z}$, while the *degree* of a configuration u is simply the sum of its values, so $\mathsf{degree}(u) \in \mathbb{Z}$. The *rank* of a configuration is an integer $\mathsf{rank}(u) \geq -1$ associated with u (see Section 1 for the definition). In our *sandpile model* we will always distinguish a vertex of G, which we will call the *sink*.

The first of our main results is an algorithm of linear complexity (in the number of vertices) to compute the rank of configurations on complete bipartite graphs: see Section 2 for its pseudocode. The analysis of this algorithm relies heavily on the combinatorial study in [ADDL] of compact (and in particular stable) sorted configurations on complete bipartite graphs.

The design of our algorithm leads to several enumerative by-products. One of them is the introduction of the *r-vector* $r(u) = (r_1, r_2, \ldots, r_n) \in \mathbb{Z}^n$ of a compact *sorted configuration* (i.e., a configuration u up to an automorphism of the graph that fixes the sink) on the complete bipartite graph $K_{m,n}$.

The second of our main results is an explicit formula for the rank of a sorted configuration on $K_{m,n}$ that is also *parking* (a *reduced* divisor in the terminology of [BN]).

Theorem 11.2. *Let u be a parking sorted configuration on $K_{m,n}$, let $r(u) = (r_1, r_2, \ldots, r_n)$ be its r-vector, and let $u_{a_m} \geq 0$ be the value of u on the sink a_m. Let*

$$u_{a_m} + 1 = nQ + R, \qquad \textit{with } Q, R \in \mathbb{N}, \quad \textit{and } 0 \leq R < n.$$

Then

$$\mathsf{rank}(u) + 1 = \sum_{i=1}^{n} \max\{0, Q + \chi(i \leq R) + r_i - 1\},$$

where $\chi(\mathcal{P})$ *is* 1 *if the assertion* $\mathcal{P}$ *is true, and* 0 *otherwise.*

In turn, a deeper understanding of the combinatorics behind the algorithm will bring our third main result, which is a formula for the generating function of the parking sorted configurations on complete bipartite graphs according to rank, degree, and the sizes of the two parts of the set of vertices: given

$$\mathcal{F}(x, y, w, h) := \sum_{n \geq 1, m \geq 1} K_{m,n}(x, y) w^m h^n,$$

where

$$K_{m,n}(x, y) = x^{(m-1)(n-1)} y \widetilde{K}_{m,n}(x^{-1}, xy), \tag{0.1}$$

and

$$\widetilde{K}_{m,n}(d, r) := \sum_{u \text{ parking sorted on } K_{m,n}} d^{\mathsf{degree}(u)} r^{\mathsf{rank}(u)},$$

the following theorem relates $\mathcal{F}(x, y, w, h)$ with the generating function $P(q; w, h)$ of *parallelogram polyominoes* according to their *area* (counted by q), their *width* (counted by w) and their *height* (counted by h) computed in [BV].

Theorem 15.7. *We have*

$$\mathcal{F}(x, y, w, h) = \frac{(1 - xy)(hw - P(x; w, h)P(y; w, h))}{(1 - x)(1 - y)(1 - h - w - P(x; w, h) - P(y; w, h))}.$$

The paper is organized as follows.

In the first section we recall some basic definitions and results about the sandpile model, but we restrict our attention mostly to the complete bipartite graph $K_{m,n}$. In particular, we define the notion of the rank of a configuration following [BN].

In the second section we state a first version of our algorithm to compute the rank of a configuration on $K_{m,n}$.

We use Sections 3–7 to recall some key results from [DL] and [ADDL]. In particular, we introduce our graphical interpretation of stable sorted configurations on $K_{m,n}$, and the corresponding graphical interpretation of the operators φ, ψ, $\mathsf{T_a}$ and $\mathsf{T_b}$, already studied in [ADDL]. All these results are used all along the rest of the paper. Notice that in these sections we use a notation slightly different (but lighter) from the one used in [ADDL].

In Section 8 we prove the correctness of our algorithm.

In Section 9 we provide a new formulation of our algorithm that uses the operators $\mathsf{T_a}$ and $\mathsf{T_b}$. We prove in this section that indeed the two algorithms are equivalent, by providing a graphical interpretation of the original algorithm.

This last analysis enables us to introduce the useful notion of a cylindric diagram of a parking sorted configuration on $K_{m,n}$ in Section 10. This provides a very efficient way to compute the rank of such a configuration.

Indeed, in Section 11, we introduce the notion of the r-vector of a stable sorted configuration on $K_{m,n}$, which, together with the cylindric diagram, gives us the formula of Theorem 11.2.

In Section 12 we provide a detailed analysis of the complexity of our algorithm, making more explicit some of its steps, proving in this way the announced linear bound.

In Section 13 we introduce two new statistics, xpara and ypara, on the parking sorted configurations on $K_{m,n}$. The strict relation between the bistatistics $(\mathsf{xpara}, \mathsf{ypara})$ and $(\mathsf{degree}, \mathsf{rank})$ is encoded in the change of variables (0.1).

In Section 14 we show that the bistatistic $(\mathsf{xpara}, \mathsf{ypara})$ is symmetrically distributed, and its symmetry is explained by the Riemann–Roch theorem of Baker and Norine [BN, Theorem 1.12], cf. Theorem 14.1, but also combinatorially via our graphical interpretation.

Finally, in Section 15 we prove the formula in Theorem 15.7.

1. Basic definitions and results

The *complete bipartite graph* $K_{m,n}$ is the graph (V, E) whose vertex set $V = A_m \sqcup B_n$ is the disjoint union of the sets $A_m := \{a_1, a_2, \dots, a_m\}$ and $B_n := \{b_1, b_2, \dots, b_n\}$, and where there is exactly one unoriented edge $\{a_i, b_j\} \in E$ for any $a_i \in A_m$ and $b_j \in B_n$.

To work in the sandpile model framework, we often distinguish one vertex, that is called the *sink*. For $K_{m,n}$, we typically choose a_m to be the sink. For this reason, the notation $A_{m-1} := A_m \setminus \{a_m\}$ will be sometimes useful.

A *configuration* u on the graph $K_{m,n}$ is a function $u : V \to \mathbb{Z}$, i.e., $u \in \mathbb{Z}^V$, that we also denote as

$$u := \begin{pmatrix} u_{a_1}, u_{a_2}, \dots, u_{a_{m-2}}, u_{a_{m-1}}; u_{a_m} \\ u_{b_1}, u_{b_2}, \dots, u_{b_n} \end{pmatrix},$$

where for any vertex $c \in A_m \cup B_n$, $u_c := u(c) \in \mathbb{Z}$ denotes the *value* of the configuration u at the vertex c. Notice the semicolon which distinguishes the value at the sink a_m.

The *degree* of a configuration u is defined as

$$\mathsf{degree}(u) := \sum_{c \in A_m \cup B_n} u_c.$$

We need some more notation. For any $c \in V$, let $e_{c,\tilde{c}}$ be 1 if $\{c, \tilde{c}\} \in E$ and 0 otherwise. Furthermore, let $d_c := \sum_{\tilde{c} \in V} e_{c,\tilde{c}}$ be the degree of the vertex c. Also, let us denote by $\epsilon^{(c)} \in \mathbb{Z}^V$ the *Dirac configuration* which has value 1 on c and 0 elsewhere.

Given a vertex $c \in V$, the *toppling operator* $\Delta^{(c)} \in \mathbb{Z}^V$ is defined as

$$\Delta^{(c)} = d_c \epsilon^{(c)} - \sum_{\tilde{c} \in V} e_{c,\tilde{c}} \epsilon^{(\tilde{c})}.$$

The *toppling* of the vertex $c \in A_m \cup B_n$ in the configuration u corresponds to computing $u - \Delta^{(c)}$, which means that an amount of 1 is sent from c to every neighbour along the corresponding edge.

Remark 1.1. In graph theory, the matrix defined by the rows $(\Delta^{(c)})_{c \in V}$ is called the *Laplacian matrix* of the underlying graph. It is well known and also clear from the previous observation

that

$$\sum_{c\in A_m\cup B_n} \Delta^{(c)} = 0 \in \mathbb{Z}^V.$$

So, for example,

$$\Delta^{(c)} = -\sum_{\substack{\tilde{c}\in A_m\cup B_n\\ \tilde{c}\neq c}} \Delta^{(\tilde{c})}.$$

We say that the configurations u and v are *toppling equivalent*, also denoted by $u \equiv_\Delta v$, if there exists a finite sequence of topplings specified by the vertices $c_1, c_2, \ldots, c_k$ such that $v = u - \sum_{i=1}^{k} \Delta^{(c_i)}$.

Using Remark 1.1, it is easy to prove the following proposition.

Proposition 1.2 (Dhar)**.** *The toppling equivalence is an equivalence relation.*

A vertex c is *unstable* in a configuration u if $u_c \geq d_c$, otherwise this vertex is *quasi-stable*. Toppling once an unstable vertex c preserves the non-negativity of its value, as the only decremented value is at vertex c. Such a toppling is called a *legal* toppling. A configuration is *quasi-stable* if any vertex distinct from the sink is quasi-stable. The *relaxation process* of a configuration u consists of iteratively performing a legal toppling of an unstable vertex distinct from the sink, as long as there exists one.

The following proposition is due to Dhar [D2] (cf. also [CR, Proposition 2.1]).

Proposition 1.3 (Dhar)**.** *Given a configuration u, the relaxation process terminates at a quasi-stable configuration independent of the ordering of legal topplings.*

A configuration u is *non-negative* if $u_c \geq 0$ for every $c \in A_m \cup B_n$.

A configuration u is *non-negative outside the sink* if for any vertex $c \in A_{m-1} \cup B_n$, hence any c distinct from the sink a_m, the value u_c is non-negative. Otherwise, the configuration u has a negative value outside the sink.

In this paper, we define a *stable configuration* to be a quasi-stable configuration which is also non-negative outside the sink.

Remark 1.4. Notice that what we called "quasi-stable configuration" is sometimes called "stable configuration" in the literature.

A configuration u is *effective* if u is toppling equivalent to a non-negative configuration.

The *rank* of a configuration u is defined as

$$\mathsf{rank}(u) := -1 + \min\{\mathsf{degree}(f) \mid f \text{ is non-negative and } u - f \text{ is non-effective}\}.$$

In other words, the rank incremented by one is the minimal cumulative decrease of values in the configuration u needed to reach a non-effective configuration.

It is clear from the definitions that if $u \equiv_\Delta v$ then $\mathsf{rank}(u) = \mathsf{rank}(v)$.

A non-negative configuration f such that $u - f$ is non-effective and $\mathsf{degree}(f) = \mathsf{rank}(u)+1$ is called a *proof for the rank* of u.

A configuration u is *parking* (with respect to the sink a_m) if u is non-negative outside the sink, and for any non-empty subset C of $A_{m-1}\cup B_n$ the configuration $u - \Delta^{(C)}$ has a negative

value outside the sink, where

$$\Delta^{(C)} := \sum_{c \in C} \Delta^{(c)}.$$

For a proof of the following proposition see [BN, Proposition 3.1] or [CL, Corollary 33].

Proposition 1.5 (Dhar). *Any configuration u is toppling equivalent to exactly one parking configuration (with respect to the sink a_m), which we denote by* $\mathsf{park}(u)$.

Baker and Norine observed that the test that a configuration u is effective may be reduced to the fact that the value of a_m is non-negative in the configuration $\mathsf{park}(u)$.

For a proof of the following theorem see [BN, Theorem 3.3] or [CL, Proposition 37].

Theorem 1.6 (Baker–Norine). *A configuration u is effective if and only if* $\mathsf{park}(u)$ *is non-negative.*

2. A greedy algorithm for the rank on $K_{m,n}$: statement

We first describe the following non-deterministic greedy algorithm which computes a proof for the rank of any configuration u of $K_{m,n}$.

All the algorithms in this paper are written in a pseudocode resembling the programming language PYTHON. (Compare this algorithm with the one in [CL, Section 2.2].)

```
def compute_rank(u):
    u = park(u)
    rank = -1
    f = 0 # f is the 0 configuration
    while u_{a_m} >= 0:
        let i be such that u_{b_i} = 0
        u = park(u - ε^{(b_i)})
        f = f + ε^{(b_i)}
        rank = rank + 1
    return (rank, f)
```

The first main result of our paper is the following theorem.

Theorem 2.1. *The algorithm returns in f a proof for the rank of the input u, hence* $\mathsf{degree}(f) - 1 = rank = \mathsf{rank}(u)$.

In order to prove this theorem, we first need to introduce some important notions and to recall some results from [ADDL].

In a later section, we provide an optimization of this algorithm on improved data-structures to reach the announced linear arithmetic complexity.

3. Stable and sorted configurations

The complete bipartite graph $K_{m,n}$ admits numerous automorphisms, the vertex a_m being fixed or not.

Let S_k denote the set of permutations of $\{1, 2, \ldots, k\}$. We define the action of $\sigma = (\sigma^a, \sigma^b) \in S_m \times S_n$ on the configuration u on $K_{m,n}$ by

$$\sigma \cdot u := \begin{pmatrix} u_{a_{\sigma^a(1)}}, u_{a_{\sigma^a(2)}}, \ldots u_{a_{\sigma^a(m-1)}}; u_{a_{\sigma^a(m)}} \\ u_{b_{\sigma^b(1)}}, u_{b_{\sigma^b(2)}}, \ldots u_{b_{\sigma^b(n)}} \end{pmatrix}.$$

We sometimes restrict this action to the elements $\sigma = (\sigma^a, \sigma^b)$ such that $\sigma^a(m) = m$, so that the value on the distinguished sink a_m is preserved. Such elements are also written as elements of $S_{m-1} \times S_n$ instead of $S_m \times S_n$. Clearly they all act as automorphisms of $K_{m,n}$.

Most of our definitions fit well with these symmetries. For example, two configurations u and v are called *toppling and (S-)permuting equivalent*, where $S = S_m \times S_n$ or $S = S_{m-1} \times S_n$, if there exists an element $\sigma \in S$ such that $\sigma \cdot u$ and v are toppling equivalent ($\sigma \cdot u \equiv_\Delta v$). This binary relation is denoted by $u \equiv_{\Delta,S} v$.

All the properties claimed in the following lemma are easy to check directly from the definitions, so the proofs are left to the reader.

Lemma 3.1. *Let u and v be configurations of $K_{m,n}$, σ a permutation of $S_m \times S_n$, and τ a permutation of $S_{m-1} \times S_n$.*

1. *The binary relations $u \equiv_{\Delta,S_m \times S_n} v$ and $u \equiv_{\Delta,S_{m-1} \times S_n} v$ are equivalence relations;*
2. *$\sigma \cdot u$ is non-negative if and only if u is non-negative;*
3. *$\sigma \cdot u$ is effective if and only if u is effective;*
4. *$\tau \cdot u$ is non-negative outside the sink if and only if u is non-negative outside the sink;*
5. $\mathsf{park}(\tau \cdot u) = \tau \cdot \mathsf{park}(u)$*;*
6. *f is a proof for the rank of u if and only if $\sigma \cdot f$ is a proof for the rank of $\sigma \cdot u$;*
7. $\mathsf{degree}(\sigma \cdot u) = \mathsf{degree}(u)$*;*
8. $\mathsf{rank}(\sigma \cdot u) = \mathsf{rank}(u)$*.*

On $K_{m,n}$, the degree of any vertex a_i is n and the degree of any vertex b_j is m, hence a *stable* configuration u on $K_{m,n}$ (with respect to the sink a_m) is a configuration such that $0 \leq u_{a_i} < n$ for $i = 1, 2, \ldots, m-1$ and $0 \leq u_{b_j} < m$ for $j = 1, 2, \ldots, n$. Notice that we have no conditions on the value u_{a_m} of u on the sink a_m. We will sometimes identify configurations that differ only by their value on the sink. In the sequel, in these situations, the value u_{a_m} of u on the sink will be denoted by the symbol $*$.

A *sorted* configuration u on $K_{m,n}$ (with respect to the sink a_m) is a configuration such that $u_{a_i} \leq u_{a_{i+1}}$ for $i = 1, 2, \ldots, m-2$ and $u_{b_j} \leq u_{b_{j+1}}$ for $j = 1, 2, \ldots, n-1$. Notice again that there are no conditions on u_{a_m}. Clearly, given a configuration u on $K_{m,n}$, there exists a unique sorted configuration $\mathsf{sort}(u)$ for which there exists a $\tau \in S_{m-1} \times S_n$ such that $\mathsf{sort}(u) = \tau \cdot u$.

Example 3.2. Let $m = 7$, $n = 5$, and consider the configuration on $K_{7,5}$

$$u := \begin{pmatrix} 2, 0, 2, 2, 0, 0; * \\ 4, 4, 0, 0, 4 \end{pmatrix},$$

where the reader should remember that $*$ denotes the value at the sink. This is a stable configuration. The corresponding sorted configuration is

$$\mathsf{sort}(u) = \tau \cdot u = \begin{pmatrix} 0, 0, 0, 2, 2, 2; * \\ 0, 0, 4, 4, 4 \end{pmatrix},$$

where $\tau = (\tau^a, \tau^b) \in S_6 \times S_5$, $\tau^a = (1,\,5)(3,\,6) \in S_6$ and $\tau^b = (1,\,3)(2,\,4) \in S_5$ written as products of transpositions.

Since we are interested in computing the rank, because of Lemma 3.1, we may restrict ourselves to work with sorted configurations. More precisely, the rank of any configuration u is also the rank of $\mathsf{sort}(\mathsf{park}(u))$. Indeed, the first step of our algorithm consists in efficiently computing this latter configuration. Then the second step of our algorithm computes the rank of this parking sorted configuration, which is a particular case of a stable sorted configuration. In order to work with stable sorted configurations, it is convenient to introduce a pictorial interpretation of our objects.

4. A pictorial interpretation

Given a stable sorted configuration $u := \binom{u_{a_1}, u_{a_2}, \dots u_{a_{m-2}}, u_{a_{m-1}}; u_{a_m}}{u_{b_1}, u_{b_2}, \dots u_{b_n}}$, we draw its *diagram*, i.e., two lattice paths in an $m \times n$ square grid, from the southwest corner to the northeast corner, moving only by north or east unit steps, which encode u: a green path whose north step in the i-th row is $u_{b_i} + 1$ units away from the west edge of the grid, and a red path whose east step in the j-th column is $u_{a_j} + 1$ units away from the south edge of the grid, except for the m-th column (the one corresponding to the sink), which is always at distance n, i.e., on the upper edge of the $m \times n$ square grid.

Example 4.1. Let $m = 7$, $n = 5$, and consider the stable sorted configuration on $K_{7,5}$

$$u := \binom{0, 0, 0, 2, 2, 2; *}{0, 0, 4, 4, 4}.$$

Its diagram is drawn in Figure 1.

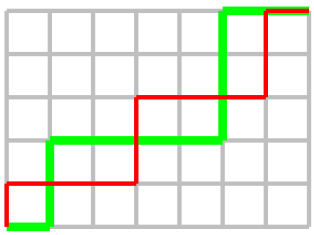

Figure 1. The stable sorted configuration $u = \binom{0,0,0,2,2,2;*}{0,0,4,4,4}$.

Notice that in this representation the red path always starts at the southwest corner with a north step, and it always ends with an east step (because of our convention). Similarly, the green path always starts at the southwest corner with an east step.

5. The operators φ and ψ on $K_{m,n}$

In [ADDL], two operators on stable sorted configurations have been studied. We recall here some of the corresponding results, but possibly with a slightly different (and lighter) notation. We refer to [ADDL] for details and proofs.

We define an operator ψ on stable sorted configurations in the following way: fix a total order on the vertices $A_{m-1} \cup B_n$ of $K_{m,n}$ different from the sink a_m, say $a_1 < a_2 < \cdots < a_{m-1} < b_1 < b_2 < \cdots < b_n$.

Given a stable configuration u on $K_{m,n}$, if for every nonempty $C \subseteq A_{m-1} \cup B_n$ the configuration $u + \Delta^{(C)}$ is not stable, then define $\psi(u) := u$.

Otherwise, consider the following order on the subsets of $A_{m-1} \cup B_n$: if C and D are two distinct subsets of $A_{m-1} \cup B_n$, we say that C is smaller than D if $|C| < |D|$, or if $|C| = |D|$ and C is smaller then D in the lexicographic order induced by the fixed order on the vertices; i.e., if the smallest element of $C \setminus D$ is smaller than any element of $D \setminus C$.

Now let $C \subseteq A_{m-1} \cup B_n$ be minimal (in this order) such that $u + \Delta^{(C)}$ is still stable, and define $\psi(u) := \mathsf{sort}(u + \Delta^{(C)})$.

The fixed points of this operator are called *recurrent* sorted configurations (cf. [ADDL, Theorem 2.4]).

We define another operator, φ, which is a sort of a "dual" operator to ψ. We use the same total order on the vertices $A_{m-1} \cup B_n$ that we used for ψ. Then, given a stable sorted configuration u, if for every nonempty $C \subseteq A_{m-1} \cup B_n$ the configuration $u - \Delta^{(C)}$ is not stable, then define $\varphi(u) := u$. Otherwise, let $C \subseteq A_{m-1} \cup B_n$ be minimal (in the order that we defined above) such that $u - \Delta^{(C)}$ is still stable, and define $\psi(u) := \mathsf{sort}(u - \Delta^{(C)})$.

It can be shown that the fixed points of this operator are exactly the parking sorted configurations (cf. [ADDL, Theorem 2.6]).

It is known that, given a configuration u on $K_{m,n}$, there is exactly one recurrent sorted configuration v and exactly one parking sorted configuration w equivalent to u, i.e., such that $u \equiv_{\Delta,S} v$ and $u \equiv_{\Delta,S} w$ for $S = S_{m-1} \times S_n$ (cf. [CL, Theorem 28 and Corollary 33]).

It is also known that, if we start with a stable sorted configuration u, and if we apply iteratively the operator ψ, then in finitely many steps we get such a recurrent sorted configuration v, while, if we apply iteratively the operator φ, then in finitely many steps we get such a parking sorted configuration v (cf. [ADDL, Remark 5.16 and Proposition 5.18]).

6. A graphical interpretation of φ and ψ

Again, it is convenient to have a pictorial interpretation of these operators in terms of our diagrams. We refer to [ADDL, Section 5] for details and proofs.

We start by describing the operator φ. Consider the diagram of a stable sorted configuration u on $K_{m,n}$. We consider the area below the red path, and the area to the left of the green path. Their intersection (see Figure 2) will play an important role. We call it the *intersection area*.

A square in the $m \times n$ grid will be indexed by the coordinates of its northeast corner, so that the squares of the grid will have coordinates (x, y), with $1 \leq x \leq m$ and $1 \leq y \leq n$.

For example, the square in the southwest corner will be the $(1, 1)$ square, while the square in the northeast corner will be the (m, n) square.

Now, a square (i, j) will be in the intersection area if and only if $u_{a_i} \geq j-1$ and $u_{b_j} \geq i-1$.

Notice that this is always the case for the square $(1, 1)$.

In order to study the operator φ, consider the (open) connected components of the intersection area (here two squares with only one vertex in common are not considered connected).

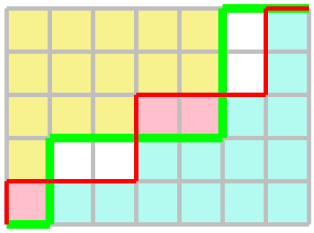

FIGURE 2. The stable configuration $u = \binom{0,0,0,2,2,2;*}{0,0,4,4,4}$. The yellow together with the pink area is the area to the left of the green path; the light blue together with the pink are is the area below the red path; the pink area is the intersection area.

We can order them by moving along the red (or equivalently the green) path starting from the southwest corner. So the first connected component will always be the one that contains the square $(1, 1)$.

If there are no connected components that contain two squares in the same row, then the configuration u is parking, hence $\varphi(u) = u$.

If there is at least one connected component that contains two squares in the same row, then consider the highest occurrence of two adjacent squares in the same row of the intersection area with a red north step crossing their leftmost vertical edge and a green east step crossing their rightmost lower horizontal unit edge, and let (i, j) and $(i + 1, j)$ be the coordinates of such squares. We have

$$\varphi(u) = \mathsf{sort}\left(u - \sum_{k=i}^{m-1} \Delta^{(a_k)} - \sum_{h=j}^{n} \Delta^{(b_h)}\right). \tag{6.1}$$

Let us see how this translates into pictures.

Given the diagram of a stable configuration $u := \binom{u_{a_1},u_{a_2},\dots u_{a_{m-2}},u_{a_{m-1}};u_{a_m}}{u_{b_1},u_{b_2},\dots u_{b_n}}$, we draw another $m \times n$ grid in such a way that the intersection of the new grid with the old one consists exactly of the upper edge of the square (m, n) of the new one and the lower edge of the square $(1, 1)$ of the old one. Then, in the new grid we reproduce precisely the red path as it appears in the old one, except for the last step, that we omit (so we get a long continuous red path); while we reproduce in the new grid the green path as it appears in the old grid, but shifted by one unit to the left. In this way we have a long red path and a long green path going through the two grids.

All this is better understood in an example: see Figure 3.

Once we have drawn the new grid with the new paths, we look for the rightmost occurrence of two adjacent squares in the intersection area on the same row, say (i, j) and $(i + 1, j)$, such that the squares $(i - 1, j)$ and $(i + 1, j - 1)$ are not in the intersection area. In other words, the left edge of the (i, j) square is a vertical step of the red path, and the lower edge

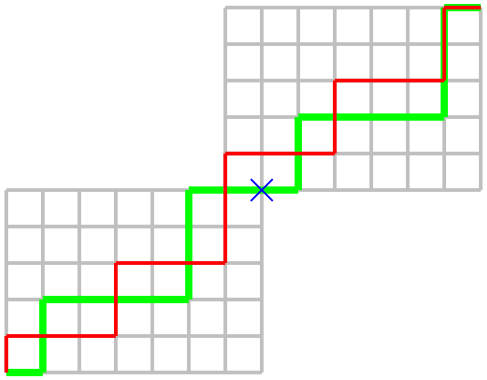

FIGURE 3. The stable configuration $u = \binom{0,0,0,2,2,2;*}{1,1,5,5,5}$, with the new grid and the new paths. The blue cross indicates the northeast corner of the new grid.

of the square $(i+1, j)$ is a horizontal step of the green path. We settle a new $m \times n$ grid with the northeast corner at the lower-right corner of the square (i, j).

If no such squares exist, then we put the new grid simply where the old one is.

We claim that the diagram that we get inside this last grid (completing the paths along the left and upper edges if necessary) will be the diagram of $\varphi(u)$ (cf. [ADDL, Theorem 5.11]). See Figure 4.

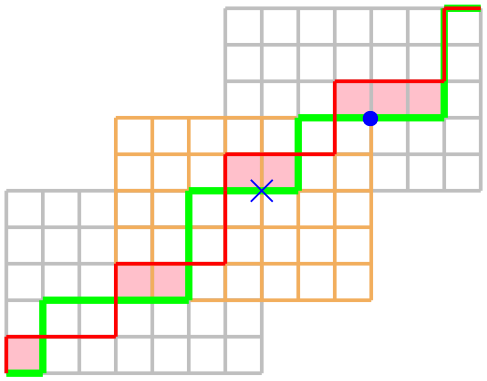

FIGURE 4. The orange grid is the last grid in the construction, while the blue spot indicates its northeast corner. The pink area is the intersection area of the long paths.

The fact that this is indeed the diagram of what we stated in the previous section to be $\varphi(u)$ is now quite clear from the picture. This is understood easily by looking at an example: in Figure 4, we see that $\varphi(u) = \binom{0,0,0,3,3,3;*}{1,1,1,4,4}$.

The following remark is crucial.

Remark 6.1. If iterating the operator φ we keep track of the diagrams that we draw, what we are really doing is to draw periodically the green path, and to draw periodically the red path with its last (east) step removed. Then acting by φ corresponds simply to moving our $m \times n$ grid down (weakly southwest) stopping at all the stable sorted configurations that we see along this periodic picture (cf. [ADDL, Theorem 5.11]). The last of such configurations is precisely our parking sorted configuration. See Figure 7 for an example.

To make this remark clearer, we complete our graphical computation by iterating this construction in our running example, until we get a parking configuration.

So we read in the diagram of the orange grid of Figure 4 that $\varphi(u) = \binom{0,0,0,3,3,3;*}{1,1,1,4,4}$.

Iterating we get Figure 5, hence $\varphi(\varphi(u)) = \binom{0,0,0,2,2,2;*}{0,0,4,4,4}$.

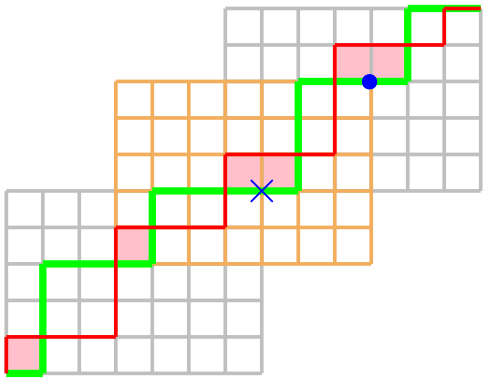

FIGURE 5. The diagram of the construction of $\varphi(\varphi(u))$.

Iterating again we get Figure 6, hence $\varphi(\varphi(\varphi(u))) = \binom{0,0,0,3,3,3;*}{0,0,0,3,3}$, which is parking on $\mathcal{K}_{7,5}$. Indeed, $\varphi(\varphi(\varphi(u))) = \mathsf{sort}(\mathsf{park}(u))$.

Remark 6.2. Notice that we could follow all the steps of this computation on the periodic diagram mentioned in Remark 6.1, just starting with u and sliding down (weakly southwest) our $m \times n$ grid, reading out all the stable configurations: see Figure 7.

Observe that our pictorial description gives the following characterization of the parking sorted configurations on $\mathcal{K}_{m,n}$ (cf. [ADDL, Corollary 5.15]).

Proposition 6.3. *The parking sorted configurations on $\mathcal{K}_{m,n}$ are precisely the stable sorted configurations on $\mathcal{K}_{m,n}$ whose intersection area does not contain two squares in the same row.*

It can be shown that the action of the operator ψ is essentially the inverse of the action of φ: in the periodic diagram mentioned in Remark 6.1, the action of ψ corresponds precisely to moving our $m \times n$ grid up (weakly northeast) stopping at all the stable sorted configurations that we see along this periodic picture (cf. [ADDL, Theorem 5.13 and Remark 5.16]).

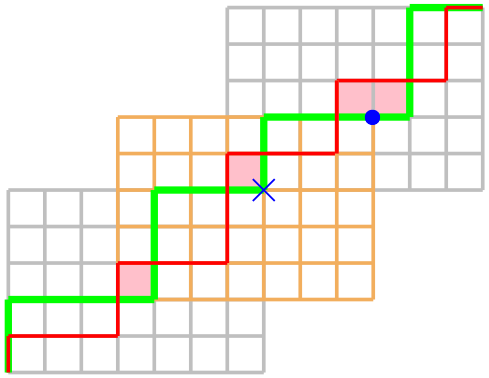

FIGURE 6. The diagram of the construction of $\varphi(\varphi(\varphi(u))) = \binom{0,0,0,3,3,3;*}{0,0,0,3,3}$.

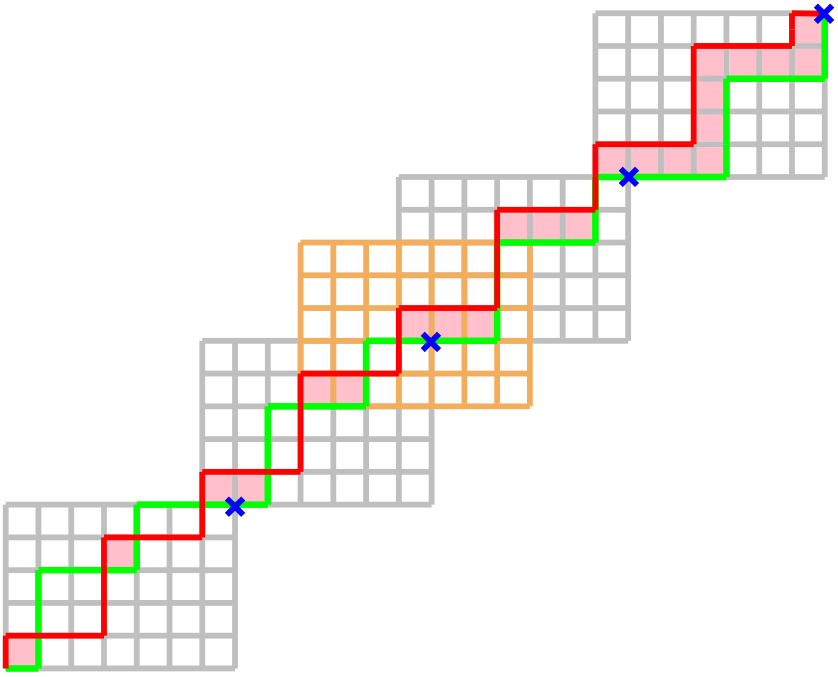

FIGURE 7. The periodic diagram of the stable sorted configuration $u = \binom{0,0,0,2,2,2;*}{1,1,5,5,5}$ (which can be seen in the orange grid).

The last of such configurations will be the corresponding recurrent sorted configuration. See Figure 7 for an example.

This gives the following characterization of recurrent sorted configurations on $K_{m,n}$ (cf. [DL, Theorem 3.7]).

Proposition 6.4. *The recurrent sorted configurations on $\mathcal{K}_{m,n}$ are precisely the stable sorted configurations on $\mathcal{K}_{m,n}$ whose intersection area is connected and touches both the southwest and the northeast corners of the $m \times n$ grid.*

7. The operators $\mathsf{T_a}$ and $\mathsf{T_b}$

For our analysis, it is convenient to introduce two operators that will explain the meaning of moving our $m \times n$ grid on the periodic diagrams. We refer again to [ADDL] for details and proofs (though we use a slightly different notation here).

For any sorted configuration u we define two operators:

$$\mathsf{T_a}(u) := \mathsf{sort}(u + \Delta^{(a_1)}) \text{ and } \mathsf{T_b}(u) := \mathsf{sort}(u + \Delta^{(b_1)}).$$

A configuration u is *compact* if

$$\max_{a_i \in A_{m-1}} u_{a_i} - \min_{a_i \in A_{m-1}} u_{a_i} \leq n \quad \text{and} \quad \max_{b_i \in B_n} u_{b_j} - \min_{b_j \in B_n} u_{b_j} \leq m.$$

The main examples of compact configurations that we will encounter in our work will be configurations with $-1 \leq u_{a_i} \leq n-1$ and $-1 \leq u_{b_i} \leq m-1$ for all i and j.

The following lemma follows immediately from the definitions.

Lemma 7.1. *The operators $\mathsf{T_a}$ and $\mathsf{T_b}$ are invertible on compact sorted configurations. More precisely, for any compact sorted configuration u, we have $\mathsf{T_a}^{-1}(u) = \mathsf{sort}(u - \Delta^{(a_{m-1})})$ and $\mathsf{T_b}^{-1}(u) = \mathsf{sort}(u - \Delta^{(b_n)})$. In formulae,*

$$\mathsf{T_a}(u) = \begin{pmatrix} u_{a_2}, u_{a_3}, \ldots, u_{a_{m-1}}, u_{a_1} + n; u_{a_m} \\ u_{b_1} - 1, u_{b_2} - 1, \ldots, u_{b_n} - 1 \end{pmatrix};$$

$$\mathsf{T_a}^{-1}(u) = \begin{pmatrix} u_{a_{m-1}} - n, u_{a_1}, u_{a_2}, \ldots, u_{a_{m-1}}; u_{a_m} \\ u_{b_1} + 1, u_{b_2} + 1, \ldots, u_{b_n} + 1 \end{pmatrix}$$

and

$$\mathsf{T_b}(u) = \begin{pmatrix} u_{u_{a_1}} - 1, u_{a_2} - 1, \ldots, u_{a_m} - 1; u_{a_m} - 1 \\ u_{b_2}, u_{b_3}, \ldots, u_{b_n}, u_{b_1} + m \end{pmatrix};$$

$$\mathsf{T_b}^{-1}(u) = \begin{pmatrix} u_{u_{a_1}} + 1, u_{a_2} + 1, \ldots, u_{a_m} + 1; u_{a_m} + 1 \\ u_{b_n} - m, u_{b_1}, u_{b_2}, \ldots, u_{b_{n-1}} \end{pmatrix}.$$

In terms of our graphical description, the operator $\mathsf{T_a}$ corresponds to moving our $m \times n$ grid by one unit in direction east. Similarly, the operator $\mathsf{T_b}$ corresponds to moving the grid by one unit in direction north. The graphical meaning of $\mathsf{T_a}^{-1}$ and $\mathsf{T_b}^{-1}$ is now clear (cf. [ADDL, Lemma 5.7], where $\mathsf{T_a}^{-1}$ and $\mathsf{T_b}^{-1}$ are denoted by $T^{>n}$ and $T^{\leq n}$, respectively). Therefore our operators φ and ψ can be easily seen as suitable iterations of the operators $\mathsf{T_a}^{-1}$ and $\mathsf{T_b}^{-1}$, and $\mathsf{T_a}$ and $\mathsf{T_b}$, respectively. For example, if u is a stable sorted configuration on $K_{m,n}$, and (i,j) is the cell of the diagram of u described right before equation (6.1), then $\varphi(u) = \mathsf{T_a}^{-m+i}\mathsf{T_b}^{-n+j-1}(u)$.

We will use these observations later in our paper.

We conclude this section by recalling that any compact sorted configuration u on $K_{m,n}$ is well described by $\mathsf{sort}(\mathsf{park}(u))$ and two integers (cf. [ADDL, Proposition 5.12]).

Proposition 7.2. *For any compact sorted configuration u on $K_{m,n}$, we have a unique pair $(k_a, k_b) \in \mathbb{Z}^2$ such that*

$$u = \mathsf{T_a}^{k_a}\mathsf{T_b}^{k_b}\mathsf{sort}(\mathsf{park}(u)).$$

8. A greedy algorithm for the rank on $K_{m,n}$: correctness

The aim of this section is to prove Theorem 2.1, i.e., the correctness of our algorithm.

We start with an auxiliary result.

Lemma 8.1. *If u is a parking configuration on $K_{m,n}$ (with respect to the sink a_m), then there exists $b_i \in B_n$ such that $u_{b_i} = 0$.*

Proof. The complete bipartite graph $K_{m,n}$ is a particular case of a simple graph without loops, i.e., there is at most one edge between two of its vertices and the endpoints of any of its edges are distinct.

We prove the more general result by contradiction that, in a parking configuration u on a simple graph without loops, at least one vertex c among the neighbours $N(s)$ of the sink s has value $u_c = 0$.

Since in $K_{m,n}$ the neighbours $N(a_m)$ of the sink a_m form exactly the set B_n, this would imply our result.

Let C be the set of vertices of our graph, and for any $c \in C$ let us denote by $\Delta^{(c)}$ the corresponding toppling operator of the vertex c. Notice that Remark 1.1 is still valid in this more general setting.

Now, if $u_c \neq 0$ for all $c \in N(s)$, since u is parking, it is non-negative outside the sink s, hence $u_c \geq 1$ for $c \in N(s)$. But in this case the configuration

$$u - \Delta^{(C-\{s\})} = u - \sum_{c \in C-\{s\}} \Delta^{(c)} = u + \Delta^{(s)}$$

is also non-negative outside the sink s, since only each neighbour of s loses exactly 1. This contradicts the assumption that u is parking. □

As we already observed in Section 3, *we* can always *assume that our configurations are sorted*. So from now on we will do it. We will observe later (in Section 12) that sorting does not increase the overall complexity of the algorithm.

Remark 8.2. Notice that Lemma 8.1 is an immediate consequence of our pictorial description of parking sorted configurations: the square $(1, 1)$ is always in the intersection area, and there cannot be two squares in the same row, therefore the first two steps of the green path (starting from the southwest corner) must be east and then north, hence the value at b_1 must be 0. We gave anyway our proof, since it works in a more general setting.

It follows from Lemma 8.1 that in a parking sorted configuration u on $K_{m,n}$ we must have $u_{b_1} = 0$. Hence in the **while** loop of our algorithm we can always take $i = 1$.

So, for our analysis we need to understand how to compute $\mathsf{sort}(\mathsf{park}(u - \epsilon^{(b_1)}))$ for a parking sorted configuration u on $K_{m,n}$. As usual, a pictorial interpretation of this operation will be very useful.

First of all, recall that in the intersection area of the diagram of a parking sorted configuration there cannot be two squares in the same row, and there is always the square $(1, 1)$.

Now notice that the diagram of $u - \epsilon^{(b_1)}$ can be obtained from the diagram of u by simply replacing the first two steps (starting from the southwest corner) of the green path, which have to be east and then north, by the steps north and then east. See Figure 8 for an example.

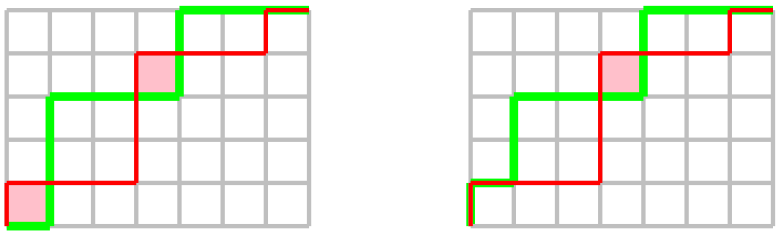

FIGURE 8. On the left, the parking sorted configuration $u = \binom{0,0,0,3,3,3;*}{0,0,0,3,3}$. On the right, the configuration $u - \epsilon^{(b_1)} = \binom{0,0,0,3,3,3;*}{-1,0,0,3,3}$.

Notice that the graphical interpretation of the operators φ, ψ, $\mathsf{T_a}$ and $\mathsf{T_b}$ works also for the diagrams of $u - \epsilon^{(b_1)}$, as this is still a compact configuration. In particular, recall from Remark 6.1 that, in order to find the parking sorted configuration equivalent to $u - \epsilon^{(b_1)}$, we need to look at the periodic diagram obtained by reproducing periodically the green path and the red path without its last east step (next to the northeast corner) of the diagram of $u - \epsilon^{(b_1)}$. Since u was parking, it is clear from the graphical interpretation that the parking sorted configuration of the periodic diagram is obtained by moving up (weakly northeast) the frame to reach the first stable configuration (moving down we will not encounter any stable configuration): this is going to be our $\mathsf{sort}(\mathsf{park}(u - \epsilon^{(b_1)}))$. See Figure 9 for the example of the computation of $\mathsf{sort}(\mathsf{park}(u - \epsilon^{(b_1)}))$ of Figure 8.

This provides the pictorial interpretation we were looking for.

In order to prove the correctness of our algorithm we need one more definition and two auxiliary results.

The *support* of a non-negative configuration f is the set of vertices $c \in A_m \cup B_n$ such that $f_c > 0$.

Lemma 8.3. *Let u be a configuration on $K_{m,n}$. Then there exists a proof f for the rank of u whose support is included in B_n.*

Proof. Let $H(u) = \min\{\sum_{a_i \in A_m} f_{a_i} \mid \ f \text{ is a proof for the rank of } u\}$.

We show by contradiction that $H(u) = 0$, implying the lemma. We assume that $H(u) > 0$ and we consider a proof f for the rank of u such that $\sum_{a_i \in A_m} f_{a_i} = H(u)$. Since $H(u) \geq 1$ there exists an i such that $f_{a_i} \geq 1$. Consider the configuration $g := f - \epsilon^{(a_i)}$, which is still non-negative, but it is not a proof for the rank of u. So, in particular, by Theorem 1.6, $\mathsf{park}(u-g)$ must be non-negative. Now, if $\mathsf{park}(u-g)-\epsilon^{(a_i)}$ is still non-negative, then, because

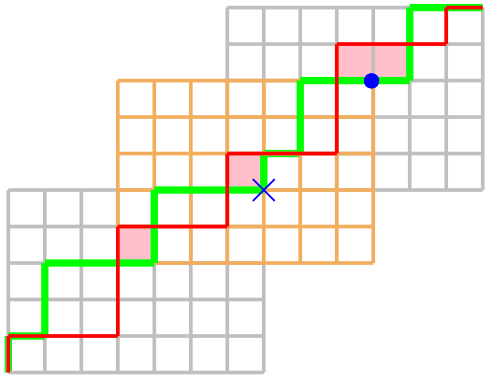

FIGURE 9. The diagram of the construction of $\mathsf{sort}(\mathsf{park}(u-\epsilon^{(b_1)})) = \binom{0,0,0,2,2,2,*}{0,0,3,4,4}$.

of Proposition 6.3, it must be still parking, and in fact $\mathsf{park}(u-g)-\epsilon^{(a_i)} = \mathsf{park}(u-f)$, contradicting the fact that f is a proof for the rank of u. Therefore the value of $\mathsf{park}(u-g)$ at a_i must be 0.

Set $v := \mathsf{sort}(\mathsf{park}(u-g))$. We just saw that $v_{a_1} = 0$. As v is a parking sorted configuration on $K_{m,n}$, by Lemma 8.1 we must have $v_{b_1} = 0$.

So, in the diagram of v the box $(1,1)$ constitutes its own connected component of the intersection area.

The crucial observation is that, when we compute $\mathsf{sort}(\mathsf{park}(v-\epsilon^{(a_1)}))$ and $\mathsf{sort}(\mathsf{park}(v-\epsilon^{(b_1)}))$, in both cases in our periodic diagram we are going to move our $m\times n$ grid up (strictly northeast), moving the southwest corner of the grid onto the southwest corner of the next connected component of the intersection area, and this corner will be exactly the same in the two computations. It turns out that the number removed from the sink in this operation is exactly the height of this corner: in fact this is the number of iterations of the operator $\mathsf{T_b}$ needed to reach the corner, each of which decreases the value at the sink by 1 (the number of iterations of the operator $\mathsf{T_a}$ needed to reach the corner is irrelevant, as $\mathsf{T_a}$ does not change the value at the sink).

All this shows that

$$\mathsf{sort}(\mathsf{park}(u-f)) = \mathsf{sort}(\mathsf{park}(v-\epsilon^{(a_1)}))$$

and

$$\mathsf{sort}(\mathsf{park}(u-f+\epsilon^{(a_i)}-\epsilon^{(b_j)})) = \mathsf{sort}(\mathsf{park}(v-\epsilon^{(b_1)}))$$

for a suitable j, and they both have the same value at the sink, so this value must be negative in both cases. Therefore also $\widetilde{f} := f-\epsilon^{(a_i)}+\epsilon^{(b_j)}$ is a proof for the rank of u. But $\sum_{a_i\in A_m}\widetilde{f} = H(u)-1$, giving a contradiction. Therefore we must have $H(u)=0$, as claimed. □

Lemma 8.4. *Let u be a configuration on $K_{m,n}$. If u is non-negative and $u_{b_i}=0$, then there exists a proof g for the rank of u such that $g_{b_i}>0$.*

Proof. Let f be a proof for the rank of u of support included in B_n, whose existence is guaranteed by Lemma 8.3. If $f_{b_i} > 0$, then $g = f$ has the stated properties. Otherwise, we have $f_{b_i} = 0$. Since $u - f$ is non-effective and the support of f is in B_n, there exists b_j ($\neq b_i$) such that $u_{b_j} - f_{b_j} < 0$. Set $\widetilde{f} := f - (f_{b_j} - u_{b_j})\epsilon^{(b_j)}$ and $w := u - \widetilde{f}$. Notice that $\widetilde{f}$ is still non-negative, and, by definition, we have $w_{b_i} = 0 = w_{b_j}$.

We claim that $(f_{b_j} - u_{b_j})\epsilon^{(b_j)}$ is a proof for the rank of w. Indeed, $w - (f_{b_j} - u_{b_j})\epsilon^{(b_j)} = u - f$ is non-effective, and if there is a non-negative configuration g with

$$\mathsf{degree}(g) < \mathsf{degree}((f_{b_j} - u_{b_j})\epsilon^{(b_j)}) = (f_{b_j} - u_{b_j})$$

for which $w - g$ is non-effective, then $h := f - (f_{b_j} - u_{b_j}) + g$ is non-negative,

$$\mathsf{degree}(h) < \mathsf{degree}(f) = \mathsf{rank}(u) + 1$$

and $w - g = u - h$ is non-effective, contradicting the definition of $\mathsf{rank}(u)$.

Now, since $w_{b_i} = 0 = w_{b_j}$, using the symmetry exchanging exactly b_i and b_j, we deduce that also $(f_{b_j} - u_{b_j})\epsilon^{(b_i)}$ is a proof for the rank of w. Therefore $g := \widetilde{f} + (f_{b_j} - u_{b_j})\epsilon^{(b_i)}$ is also a proof for the rank of u, and by construction $g_{b_i} > 0$, as we wanted. □

We are now ready to prove the correctness of our algorithm, i.e., to prove Theorem 2.1.

Proof of Theorem 2.1. The run of the algorithm is possible since the vertex b_i in the **while** loop always exists according to Lemma 8.1. This algorithm terminates since each loop iteration decreases the degree of the configuration by 1 and a configuration of negative degree is necessarily non-effective. Hence it remains to show that f is a proof for the rank of the input u, where $(rank, f)$ is the output of the algorithm. Since it is clear that f is non-negative and $rank = \mathsf{degree}(f) - 1$, it remains to show that $rank$ is the actual rank of the input u.

We prove the correctness by induction on the total number k of iterations of the **while** loop. If $k = 0$, then the first computed parking configuration is non-effective and $f = 0$ is the expected proof for the rank of u, which is indeed -1 in this case.

If $k > 0$, then the first loop iteration removes 1 from the value of $\mathsf{park}(u)$ at some vertex b_i, and, by the inductive hypothesis, the following $k-1$ iterations compute a proof $f' := f - \epsilon^{(b_i)}$ for the rank of $u' := \mathsf{park}(u) - \epsilon^{(b_i)}$. Let g be a proof for the rank of $\mathsf{park}(u)$ such that $g_{b_i} > 0$, which exists by Lemma 8.4.

Observe that $g' := g - \epsilon^{(b_i)}$ is a proof for the rank of $u' = \mathsf{park}(u) - \epsilon^{(b_i)}$. Indeed, on the one hand $u' - g' = \mathsf{park}(u) - g$ is non-effective, hence

$$\mathsf{rank}(u') \leq \mathsf{degree}(g') - 1 = (\mathsf{degree}(g) - 1) - 1 = \mathsf{rank}(\mathsf{park}(u)) - 1;$$

on the other hand, for any configuration w on $K_{m,n}$, we have the obvious inequality $\mathsf{rank}(w + \epsilon^{(b_i)}) \leq \mathsf{rank}(w) + 1$, hence

$$\mathsf{rank}(\mathsf{park}(u)) \leq \mathsf{rank}(\mathsf{park}(u) - \epsilon^{(b_i)}) + 1 = \mathsf{rank}(u') + 1.$$

Therefore $\mathsf{rank}(u') = \mathsf{rank}(\mathsf{park}(u)) - 1 = \mathsf{degree}(g') - 1$, so g' is a proof of u', as claimed.

Hence, by definition of proofs for the rank of $u' = \mathsf{park}(u) - \epsilon^{(b_i)}$, we must have $\mathsf{degree}(g') = \mathsf{degree}(f')$. As a consequence, $\mathsf{degree}(g) = \mathsf{degree}(f)$, so f is also a proof for the rank of $\mathsf{park}(u)$, as expected. □

9. A useful reformulation of the algorithm

Our main goal is to provide an efficient algorithm to compute the rank on $K_{m,n}$. To achieve this, we need to provide a detailed analysis of our original algorithm, so that we will be able to optimize it.

In order to do this, it is convenient to reformulate our algorithm in terms of the operators $\mathsf{T_a}$ and $\mathsf{T_b}$. We need some more definitions.

The cell of a north step of a path is the cell whose east side is the mentioned step. In our periodic diagram of a stable sorted configuration u on $K_{m,n}$, the periodic red path (whose period consists of $m+n-1$ steps) is called *the cut*, and it disconnects the cells of the underlying infinite grid into a *left component* and a *right component*.

We claim that, as long as we are interested in computing the rank of a configuration u on $K_{m,n}$, the following algorithm is equivalent to our original algorithm, i.e., it computes the rank of u using the same principles.

```
def new_compute_rank(u):
  u = sort(park(u))
  rank = -1
  while u_{a_m} >= 0:
    while u_{b_1} >= 0:
      u = T_a(u)
    u = T_b(u)
    if u_{a_{m-1}} >= n-1:
      rank = rank+1
    u_{a_m} = u_{a_m} - 1  # each iteration of T_b takes 1 from the sink
  return rank
```

Theorem 9.1. *This new algorithm computes the rank of a configuration u on $K_{m,n}$.*

In the rest of this section we will give a proof of Theorem 9.1, via the following pictorial interpretation of it.

Recall that, in terms of our periodic diagrams, the operators $\mathsf{T_a}$ and $\mathsf{T_b}$ correspond to moving our $m \times n$ grid by one unit in directions east and north, respectively.

First of all, we claim that what this algorithm does is to move the grid in such a way that its southwest corner moves along the periodic green path (whose period consists of $m+n$ steps) in direction northeast, following it step by step; it decreases the value u_{a_m} in the sink at every iteration of $\mathsf{T_b}$, and it increases the value in *rank* after every iteration of $\mathsf{T_b}$ for which the southwest corner of the $m \times n$ grid crosses a north step whose cell is in the right component, and in fact in the intersection area.

The only non-obvious statement is the last one, which is proved once we show that the condition in the **if**, i.e., $u_{a_{m-1}} \geq n-1$, corresponds exactly to the fact that the cell of the just crossed north step of the periodic green path is in the right component. But this follows simply from the observation that the period of the red path consists of $m+n-1$ steps, hence the horizontal red step in column $m-1$ of our $m \times n$ grid is precisely n cells higher than the

horizontal red step in the column to the left of the $m \times n$ grid. This is easily understood by looking at the diagrammatic example in Figure 10.

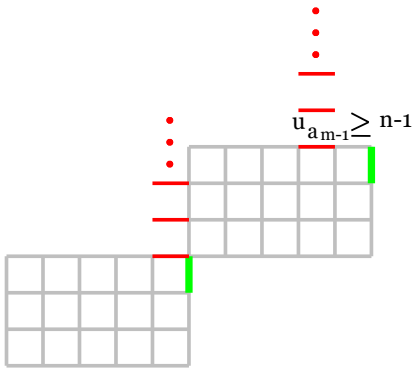

FIGURE 10. This diagrammatic example on $K_{5,3}$ explains how the condition $u_{a_{m-1}} \geq n-1$ translates into pictures.

Now that we understand the pictorial interpretation of this new algorithm, we need to understand why this is equivalent to our original algorithm.

Given a parking sorted configuration u on $K_{m,n}$, we set

$$\psi_0(u) := \mathsf{sort}(\mathsf{park}(u - \epsilon^{(b_1)})).$$

Recall that computing $\psi_0(u)$ corresponds to a loop iteration of our original algorithm. We already gave a graphical description of the action of the operator ψ_0 in Section 8. We saw that applying ψ_0 to a parking stable configuration u corresponds to replacing the first two steps (starting from the southwest corner) of the green path, which have to be east and then north, by the steps north and then east, and then, on the new periodic diagram, moving the $m \times n$ grid up (weakly northeast) in order to have the next available cell of the intersection area (there is at most one of them in each row) in the southwest corner. In our original algorithm, at every such iteration of ψ_0, we increased the local variable *rank* by 1, while we subtracted from the sink the number of north steps that we needed to move our grid.

We want to see what happens to the diagram when we apply ψ_0. To see this, it is useful to keep track of the periodic diagram that we built by iterating the operator ψ_0. In other words, we can construct a single diagram that encodes all the iterations of ψ_0. This diagram will be similar to the periodic diagram that we used to compute the iterations of ψ (or φ), but NOT identical. Indeed, this diagram will not be periodic. This is due to the fact that at every iteration of ψ_0 we first subtract $\epsilon^{(b_1)}$, and then we follow its own periodic diagram.

In particular, notice that u and $\psi_0(u)$ are not toppling equivalent, as they have different degrees, while this is obviously the case for u and $\psi(u)$.

We know that, without subtracting $\epsilon^{(b_1)}$ for each iteration of ψ_0, if we just looked at the periodic diagram and moved our $m \times n$ grid $m-1$ steps east and n steps north, we would see the same red path, but we would see the green path shifted by one unit towards east. Notice in particular that, starting with a parking sorted configuration u, we would not see a parking configuration, since, for example, in the first row the intersection area would contain exactly two cells.

Now iterating ψ_0, every time we have one cell in the intersection area, we modify the green path so that in the next periodic repetition, i.e., n rows to the north in the periodic diagram, we would still see exactly one cell, exactly in the same position: this is due to the subtraction of $\epsilon^{(b_1)}$. On the other hand, for any row that does not contain a cell in the intersection area, in the periodic repetition, i.e., n rows to the north in the periodic diagram, we will see the corresponding north step of the green path shifted by one unit towards east. At this point an example is in order.

Example 9.2. We already considered the parking sorted configuration $u = \binom{0,0,0,3,3,3;*}{0,0,0,3,3}$. Notice that there are only two rows in which there is a cell of the intersection area, so we want to compare u with $\psi_0(\psi_0(u))$. We already drew a picture of how to compute $\psi_0(u)$, see Figure 9. If we compute $\psi_0(\psi_0(u))$, we get $\psi_0(\psi_0(u)) = \binom{0,0,0,3,3,3;*}{0,1,1,3,4}$, see Figure 11.

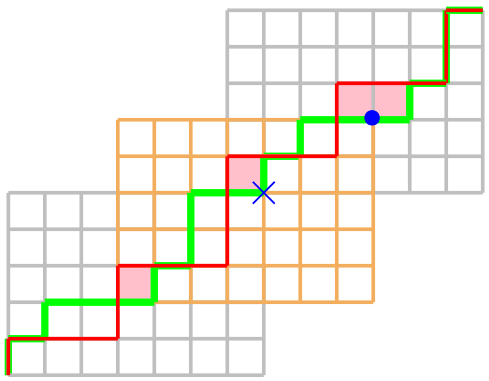

FIGURE 11. The diagram of the construction of $\psi_0(\psi_0(u)) = \binom{0,0,0,3,3,3;*}{0,1,1,3,4}$.

As claimed, the same picture could be obtained by producing a single diagram that keeps track of the iterations of ψ_0. In order to draw this diagram, for every iteration of ψ_0 we move the $m \times n$ grid to the next (weakly northeast) stable configuration along the usual periodic diagram of the configuration appearing in the $m \times n$ grid, like we would do in the computation of ψ, but in this new diagram we modify the green path at the n-th row (we start counting the rows from the bottom row of the $m \times n$ grid, which will be row 0), by keeping the cell of the intersection area in that row where it used to be. This accounts for the subtraction of $\epsilon^{(b_1)}$.

Then we repeat the procedure with $\psi_0(u)$ in place of u. The diagram for $\psi_0(\psi_0(u))$ of our example is shown in Figure 12: the pink cells are the intersection area of the diagram, and the numbers in them are the number of the corresponding row (the bottom row is row 0); the yellow cells are in the positions in which the iterations of ψ_0 modify the green path, and the number in them corresponds to the number of the bottom row of the $m \times n$ grid at the time of its creation, i.e., at the time of the iteration of ψ_0 that modified the green path.

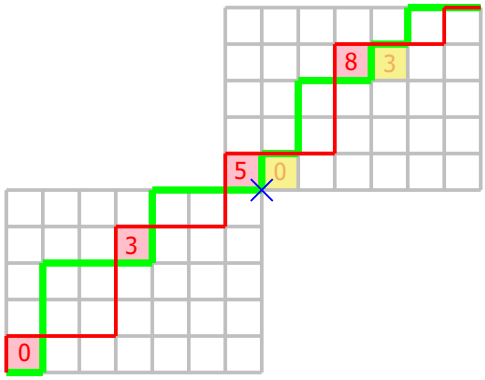

FIGURE 12. The diagram of the construction of $\psi_0(\psi_0(u)) = \binom{0,0,0,3,3,3;*}{0,1,1,3,4}$. The yellow cells underscore the positions in which the iterations of ψ_0 modify the green path.

As expected, looking at the diagram n rows to the north, the rows corresponding to the ones in which u had a cell in the intersection area are unchanged, while in the other rows the green path got shifted by one unit in direction east.

Observe that in general the red path in this diagram will be periodic of a period consisting of $m+n-1$ steps (as in the case of the periodic diagram for computing ψ), while in general the green path will not be periodic. In fact, it becomes periodic when, at a given iteration of ψ_0, in the (parking sorted) configuration every row has precisely one cell in the intersection area: from then on the green path will also be periodic of a period consisting of $m+n-1$ steps, so that every n iterations of ψ_0 we will see always the same configuration outside the sink.

See Figure 13 for more iterations of ψ_0 for the same example $u = \binom{0,0,0,3,3,3;*}{0,0,0,3,3}$. The meaning of the coloured cells and the numbers inside them is the same as the one explained for Figure 12.

Notice that indeed, after we get a configuration with precisely one cell in the intersection area in every row, the rest of the diagram becomes periodic (look at the top 10 rows of Figure 13).

Notice also that, with our enhanced diagram with the yellow cells, like the one in Figure 13, one could also deduce the proof f given by the algorithm by looking at the last iteration of ψ_0: the proof f for the configuration in the $m \times n$ diagram of the last iteration of ψ_0 will be the configuration with $f_{a_j} = 0$ for all j, and f_{b_i} equal to the number of yellow cells in row $i+1$ (remember that the bottom row is row 0) for $i = 1, 2, \ldots, n$.

Coming back to our new algorithm, let us see why it is equivalent to our original one. In our new algorithm we move the southwest corner of the $m \times n$ grid along the green path in the northeast direction, subtracting 1 from the sink at every north step, and adding 1 to the local variable *rank* every time we cross a north step whose cell is in the right component, i.e., in the intersection area. The observation is that, since we are moving in our original

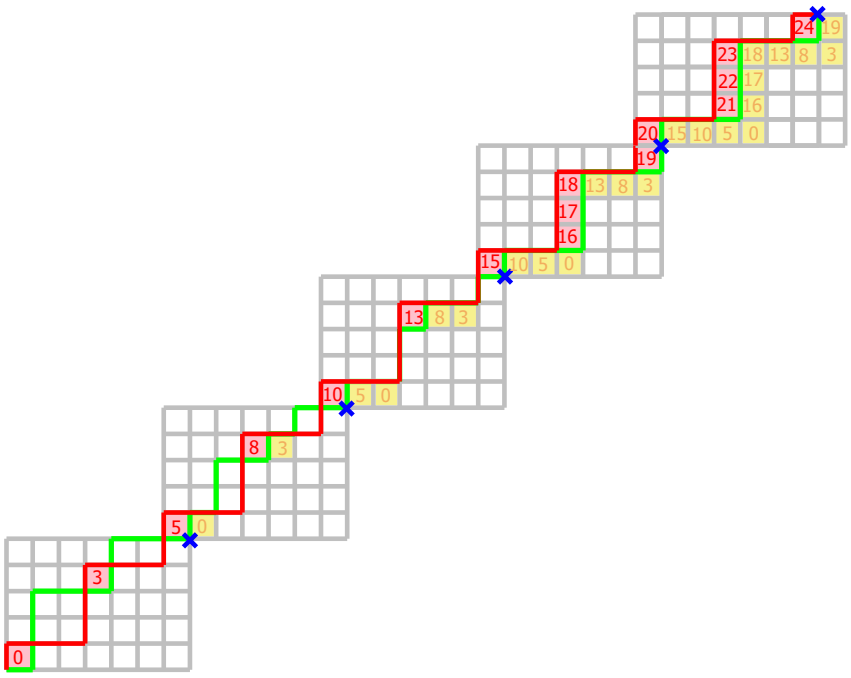

FIGURE 13. The diagram of the first few iterations of ψ_0 on $u = \binom{0,0,0,3,3,3,*}{0,0,0,3,3}$.

periodic diagram, every n rows each vertical green step gets shifted by one unit towards east. Now, once such a green vertical step falls in the right component, i.e., the corresponding cell is in the right component, then it remains in that component indefinitely. Observe that our new algorithm ignores such a vertical green step until it falls in the right component, and subsequently it always increases the local variable *rank* after crossing it by 1.

This corresponds precisely to the situation of our original algorithm, where the operator ψ_0 skips the rows where the vertical green steps are in the left component, until they fall in the right one, so that their cell is in the intersection area, and subsequently the algorithm always stops at that cell, increasing the local variable *rank* by 1 at each visit.

Since obviously both algorithms decrease the value at the sink by 1 at every vertical move of the $m \times n$ grid, this proves the equivalence of the two algorithms.

10. CYLINDRIC DIAGRAMS

Given a parking sorted configuration u on $K_{m,n}$, we associate with it a *cylindric diagram*, which allows to efficiently compute the rank of u given the diagram of u and the value u_{a_m} at the sink.

Consider a parking sorted configuration u on $K_{m,n}$, look at the corresponding diagram, and consider all the cells to the right of the green path in its n rows. If the value u_{a_m} at the

sink is negative, then of course $\mathsf{rank}(u) = -1$, and there is nothing else to do. If $u_{a_m} \geq 0$, start writing the numbers $0, 1, 2, \ldots, u_{a_m}$ in the cells to the left of the vertical steps of the green path, from the bottom row to the top row. After writing $0, 1, \ldots, n-1$, write $n+i$ in the cell to the right of i for $i = 0, 1, \ldots, n$. Then write $2n+i$ in the cell to the right of $n+i$ for $i = 0, 1, \ldots, n-1$. And so on, until one writes the number u_{a_m}. The only thing to recall is to write the number j in red if the cell containing j is in the right component (i.e., to the right of the red path), and in green if it is in the left component (i.e., to the left of the red path). This will be the *cylindric diagram* of u. See Figure 14 for an example.

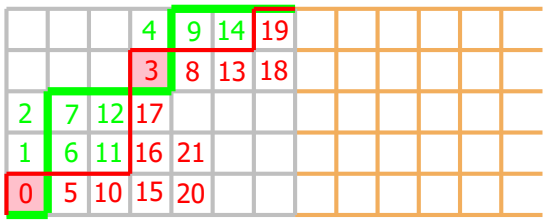

FIGURE 14. The parking sorted configuration $u = \binom{0,0,0,3,3,3;21}{0,0,0,3,3}$ with the corresponding cylindric diagram.

We claim that $\mathsf{rank}(u)$ is equal to -1 plus the number of red numbers in the cylindric diagram of u.

To see this, imagine that, running our new algorithm on the periodic diagram of u, any time the southwest corner of the $m \times n$ grid crosses a vertical green step, we write to its left a label with the number i of the vertical steps that we already crossed. So, for example, next to the first vertical green step we write 0. Because of the periodicity of the diagram, since every n rows the vertical green steps get shifted by one unit in direction east, we can imagine to "roll up" (this is why the name "cylindric" diagram) our periodic diagram modulo n in the north-south direction and modulo $m-1$ in the west-east direction. In this way we get always the same red path, but the green path would be reproduced shifted by i units in direction east at the $(i+1)$-st cycle of n rows: instead of drawing it every time, we just draw the first original green path in our picture, but we keep track of the numerical labels by recording them in our cylindric diagram where they should appear. So this is going to give us precisely the cylindric diagram of u, except for the colours of the numbers.

In this way, it is clear from our pictorial description of the algorithm that we increase the local variable *rank* by 1 precisely when the cell of the vertical green step that we just crossed is in the right component, i.e., when in our cylindric diagram we recorded a red number, while ignoring all the other vertical green steps, i.e., when we recorded a green number.

Since the labels count -1 plus the number of vertical steps that we performed in our new algorithm, and since we stop writing our labels at the number u_{a_m}, this means that, when we wrote the last red number, it was the last time that the value in our sink was non-negative,

and at the next red number, i.e., at the next iteration of our original algorithm, it would become negative. This and the correctness of our new algorithm prove our claim.

11. The r-vectors and a formula for the rank

In this section we introduce a useful tool that will enable us to optimize our algorithm. We need some notation.

Consider a stable sorted configuration u on $K_{m,n}$. For i with $1 \leq i \leq n$, let

$$r_i := u_{b_i} + 1 - |\{u_{a_j} \mid j \neq m, u_{a_j} + 1 \leq i - 1\}|.$$

Notice that r_i is simply the difference between the distance from the vertical step of the red upper path in row i (the bottom row is row 1) from the left edge of the $m \times n$ grid, and the distance from the vertical step of the green lower path in row i from the left edge of the $m \times n$ grid. We call $r(u) := (r_1, r_2, \ldots, r_n)$ the *r-vector* of the stable sorted configuration u on $K_{m,n}$.

We remark that for a stable sorted configuration, we always have $r_1 \geq 1$, and $-m + 2 \leq r_i \leq m$ for all i. Moreover, for a parking sorted configuration, Proposition 6.3 says that we always have $r_1 = 1$, and $-m + 2 \leq r_i \leq 1$ for all i. This is the translation in terms of the r-vector of the fact that in each row of the diagram of a parking sorted configuration there is at most one box in the intersection area.

Example 11.1. For example, for the parking sorted configuration $u = \binom{0,0,0,3,3,3;*}{0,0,0,3,3}$ on $K_{7,5}$ we have $r(u) = (1, -2 - 2, 1, -2)$, cf. Figure 8.

Recall that in Section 9 we introduced the operator ψ_0, which is defined on any parking sorted configuration u by $\psi_0(u) = \mathsf{sort}(\mathsf{park}(u - \epsilon^{(b_1)}))$. In that same section, we saw how the action of ψ_0 can be seen as a suitable composition of the operators $\mathsf{T_a}$ and $\mathsf{T_b}$. Moreover, we drew a diagram to explain the iterations of the operator ψ_0, cf. Figure 13.

We want to understand the relation between $r(u)$ and $r(\psi_0(u))$. In order to do this, we define an operator $\widetilde{\psi_0}$ on the set of vectors $r = (r_1, r_2, \ldots, r_n)$ with $r_i \in \mathbb{Z}$ and $-m+2 \leq r_i \leq 1$ for all i: given such a vector $r = (r_1, r_2, \ldots, r_n)$, we set

$$\widetilde{\psi_0}(r) := \begin{cases} (r_2, r_3, \ldots, r_{n-1}, r_n, 1), & \text{if } r_1 = 1, \\ (r_2, r_3, \ldots, r_{n-1}, r_n, r_1 + 1), & \text{if } r_1 < 1. \end{cases}$$

Now we can translate all the observations that we made in the discussion after Example 9.2 in terms of r-vectors. For example, it is now clear that, if $r(u)$ is the r-vector of a parking sorted configuration u on $K_{m,n}$, then

$$r(\psi_0(u)) = \widetilde{\psi_0}^{h-1}(r(u)),$$

where h is the smallest integer $h \geq 2$ such that $r_h = 1$, if such an integer exists, and $h := n+1$ otherwise. Moreover, what $\widetilde{\psi_0}^{n}$ does to $r(u)$ is to increase all the r_i's with $r_i < 1$ by 1 and leave the other r_i's fixed. The remark about the eventual periodicity of the green path of the diagram of ψ_0 can be translated into the following observation.

We can sort the set $\{r(u) \mid u \text{ is a stable sorted configuration on } K_{m,n}\}$ of all possible r-vectors with respect to the *reverse degree-lexicographic order*: given two distinct vectors

$r(u) := (r_1, r_2, \dots, r_n)$ and $r(u') := (r'_1, r'_2, \dots, r'_n)$, we set $r(u) < r(u')$ if $|r(u)| := \sum_i r_i < \sum_i r'_i =: |r(u')|$, or if $|r(u)| = |r(u')|$ and for the largest $i \in \{1, 2, \dots, n\}$ such that the difference $r'_i - r_i$ is non-zero we have $r'_i - r_i > 0$.

Then, for a parking sorted configuration u, we have $r(\widetilde{\psi}_0(u)) \geq r(u)$, so in particular $r(\psi_0(u)) \geq r(u)$, and the equality $r(\psi_0(u)) = r(u)$ holds if and only if $r(u) = (1, 1, \dots, 1)$.

We are in the position to give a formula for the rank of a parking sorted configuration u on $K_{m,n}$ in terms of u_{a_m} and the r-vector $r(u)$ (compare the following theorem with [CL, Theorem 12]).

Theorem 11.2. *Let u be a parking sorted configuration on $K_{m,n}$, let $r(u) = (r_1, r_2, \dots, r_n)$ be its r-vector, and let $u_{a_m} \geq 0$ be the value of u on the sink a_m. Let*

$$u_{a_m} + 1 = nQ + R, \qquad \text{with } Q, R \in \mathbb{N}, \quad \text{and } 0 \leq R < n.$$

Then

$$\mathsf{rank}(u) + 1 = \sum_{i=1}^{n} \max\{0, Q + \chi(i \leq R) + r_i - 1\}, \tag{11.1}$$

where $\chi(\mathcal{P})$ is 1 if the assertion $\mathcal{P}$ is true, and 0 otherwise.

Proof. The proof of this theorem follows immediately from our analysis of the cyclic diagram of u: we claim that, for every i with $1 \leq i \leq n$, the i-th summand of the right-hand side of (11.1) is precisely the number of red labels in the i-th row of our cylindric diagram (the bottom row being row 1). Indeed, the max takes care of the fact that all the labels of the row could be to the left of the red path, i.e., that they are all green; the term $Q + \chi(i \leq R)$ takes care of the fact that all the $u_{a_m} + 1$ labels (both green and red) wrap around the rows Q times with a remainder of R; finally the term $r_i - 1$ is there to remove the green labels from the counting. □

Example 11.3. Consider the parking sorted configuration $u = \binom{0,0,0,3,3,3;21}{0,0,0,3,3}$ on $K_{7,5}$. In this case, we have $r(u) = (1, -2-2, 1, -2)$, $21 + 1 = 4 \cdot 5 + 2$, hence $Q = 4$ and $R = 2$, so that the summands of the right-hand side of (11.1) are (in order) $5, 2, 1, 4, 1$, which are precisely the number of red labels in rows $1, 2, 3, 4, 5$, respectively, of the cyclic diagram of u, see Figure 14.

12. An algorithm for the rank on $K_{m,n}$ of linear arithmetic complexity

In this section we make explicit all the steps involved in our original algorithm to compute the rank of a configuration on $K_{m,n}$. Along the way, we make an analysis of the complexity of this algorithm.

In order to do this, in our complexity model we make the following two assumptions:

(1) the four elementary binary operations, i.e., addition, subtraction, product, and Euclidean division, each cost 1 (*linear arithmetic complexity model*);
(2) we can access the position of an array in constant time (this will be needed for sorting).

We use the usual O-notation to estimate the number of operations of the algorithm: we say that we performed $O(n)$ operations if there exists a constant $C > 0$ such that for every $n \geq 1$ we actually performed $\leq C \cdot n$ operations.

In our algorithm, we need to perform the following steps.

STEP 1: Given any configuration $u = \binom{u_{a_1}, u_{a_2}, \ldots, u_{a_{m-1}}; u_{a_m}}{u_{b_1}, u_{b_2}, \ldots, u_{b_n}}$ on $K_{m,n}$, compute a stable configuration $u' = \binom{u'_{a_1}, u'_{a_2}, \ldots, u_{a'_{m-1}}; u'_{a_m}}{u'_{b_1}, u'_{b_2}, \ldots, u'_{b_n}}$ toppling-equivalent to u.

In order to do this, what we do in practice is to use Euclidean division to compute first

$$u_{b_i} := q_i m + u'_{b_i}, \qquad \text{with } q_i, u'_{b_i} \in \mathbb{Z} \text{ and } 0 \leq u'_{b_i} < m \text{ for } i = 1, 2, \ldots, n,$$

and then we compute

$$u_{a_i} + \sum_{j=1}^{n} q_i = \widetilde{q}_i n + u'_{a_i}, \qquad \text{with } \widetilde{q}_i, u'_{a_i} \in \mathbb{Z} \text{ and } 0 \leq u'_{a_i} < n \text{ for } i = 1, 2, \ldots, m-1;$$

finally we set

$$u_{a_m} := \mathsf{degree}(u) - \sum_{i=1}^{n} u'_{b_i} - \sum_{j=1}^{m-1} u'_{a_j}.$$

Claim 1. *The configuration u' is toppling equivalent to u.*

Proof. It is straightforward to check that

$$u' = u - \sum_{i=1}^{n} q_i \Delta^{(b_i)} - \sum_{j=1}^{m-1} \widetilde{q}_j (\Delta^{(a_j)} - \Delta^{(a_m)}). \qquad \square$$

Notice that to compute u' we used $n + (m-1)$ Euclidean divisions and two sums with $\leq m+n$ terms each. Hence, using our assumption (1) on the complexity model, we performed a total of $O(m+n)$ operations.

STEP 2: Given a stable configuration u (with respect to the sink a_m), we sort it, i.e., we compute $\mathsf{sort}(u)$. To do this, we use the so called *counting sort* (cf. [CLRS, Section 8.2]), which is an algorithm that takes an integer m and a list $w = (w_1, w_2, \ldots, w_n)$ of n integers w_i with $0 \leq w_i \leq m$, and it returns a list *out* with the w_i in increasing order. This algorithm uses an array of lists: we give a pseudocode for completeness.

```
def counting_sort(m,w): # the input w = [w_1, w_2, ..., w_n] is a list of integers 0 ≤ w_i ≤ m
    for i in [0,1,2,...,m]:
        ℓ_i = [ ] # each ℓ_i is initialized as an empty list
    ℓ = < ℓ_0, ℓ_1, ℓ_2, ..., ℓ_m > # an array of m+1 empty lists
    for i in [1,2,...,n]:
        ℓ_{w_i}.append(w_i) # we append w_i to ℓ_{w_i}
    out = [ ] # out is initialized as an empty list
    for j in [0,1,2,...,m]: # we join the lists ℓ_i in the given order
        out = out joined to ℓ_j
    return out
```

This algorithm, using our assumption (2) on the complexity model, performs $O(m+n)$ operations.

Now we can use this algorithm to order first the u_{a_i}'s, where $0 \leq u_{a_i} \leq n-1$ for $i = 1, 2, \ldots, m-1$, and then the u_{b_j}'s, where $0 \leq u_{b_j} \leq m-1$ for $j = 1, 2, \ldots, n$. Therefore, to compute $\mathsf{sort}(u)$, we perform in total $O(m+n)$ operations.

Step 3: Given a stable sorted configuration $u = \binom{u_{a_1}, u_{a_2}, \ldots, u_{a_{m-1}}; u_{a_m}}{u_{b_1}, u_{b_2}, \ldots, u_{b_n}}$ on $K_{m,n}$, we compute $u'' = \mathsf{sort}(\mathsf{park}(u))$.

Recall the definition of the r-vector $r(u) = (r_1, r_2, \ldots, r_n)$ of a stable sorted configuration u: for $i = 1, 2, \ldots, n$, we set

$$r_i := u_{b_i} + 1 - |\{u_{a_j} \mid j \neq m, u_{a_j} + 1 \leq i - 1\}|.$$

By Proposition 6.3, we know that u is parking if and only if $r_i \leq 1$ for $i = 1, 2, \ldots, n$. In particular, we must have $r_1 = 1$.

Now let h be the minimal integer i such that $1 \leq i \leq n$ and $r_i = \max\{r_j \mid j = 1, 2, \ldots, n\}$, and let $k := u_{b_h} - r_h + 2 = 1 + |\{u_{a_j} \mid j \neq m, u_{a_j} + 1 \leq h - 1\}|$.

Remark 12.1. For $h = 1$, we have $k = 1$, while, for $h \geq 2$, we have also $\max\{r_j \mid j = 1, 2, \ldots, n\} \geq 2$ (since $r_1 \geq 1$), hence $k \leq 1 + (m-2) = m-1$.

Now we set

$$u' := u + (r_h - 2)\Delta^{(a_m)} - \sum_{s=k}^{m-1} \Delta^{(a_s)} - \sum_{t=h}^{n} \Delta^{(b_t)}, \tag{12.1}$$

and

$$u'' := \mathsf{sort}(u').$$

We claim that $u' = \mathsf{park}(u)$, so that $u'' = \mathsf{sort}(u') = \mathsf{sort}(\mathsf{park}(u))$.

In fact, before proving the claim, we can give explicit formulae for the values of u' and u'' on the vertices.

It is straightforward to check that, given $u' = \binom{u'_{a_1}, u'_{a_2}, \ldots, u'_{a_{m-1}}; u'_{a_m}}{u'_{b_1}, u'_{b_2}, \ldots, u'_{b_n}}$,

$$\begin{aligned} u'_{b_i} &= u_{b_i} - u_{b_h} + \chi(i \leq h-1) \cdot m, && \text{for } i = 1, 2, \ldots, n, \\ u'_{a_j} &= u_{a_j} - (h-1) + \chi(j \leq k-1) \cdot n, && \text{for } j = 1, 2, \ldots, m-1, \end{aligned} \tag{12.2}$$

and

$$u'_{a_m} := \mathsf{degree}(u) - \sum_{i=1}^{n} u'_{b_i} - \sum_{j=1}^{m-1} u'_{a_j}.$$

Moreover, it can be shown that

$$u'' = \begin{pmatrix} u'_{a_k}, u'_{a_{k+1}}, \ldots, u'_{a_{m-1}}, u'_{a_1}, u'_{a_2}, \ldots, u'_{a_{k-1}}; u'_{a_m} \\ u'_{b_h}, u'_{b_{h+1}}, \ldots, u'_{b_n}, u'_{b_1}, u'_{b_2}, \ldots, u'_{b_{h-1}} \end{pmatrix}. \tag{12.3}$$

To see where the formula (12.1) comes from, we should look at our periodic diagram used to understand the action of φ, ψ, $\mathsf{T_a}$, and $\mathsf{T_b}$. Given the diagram of a stable sorted configuration u, the diagram of $u + \Delta^{(a_m)}$ has the same red path, while the green path is shifted by one unit towards west (notice that in this situation we may get some -1's on the

vertices b_i). On the other hand, the diagram of $\mathsf{sort}\left(u - \sum_{s=k}^{m-1} \Delta^{(a_s)} - \sum_{t=h}^{n} \Delta^{(b_t)}\right)$ is the diagram that we see in the $m \times n$ grid after having it moved $m-k$ units west and $n-h+1$ units south in the periodic diagram of u: this comes from the graphical interpretation of the operators $\mathsf{T_a}^{-1}$ and $\mathsf{T_b}^{-1}$ (cf. the comments after Lemma 7.1).

Now, by definition of h and r_h, in $u+(r_h-2)\Delta^{(a_m)}$ the h-th row is the lowest row in which the intersection area has precisely two squares, while in all the other rows the intersection area has at most 2 squares. Moreover, by definition of k, the leftmost box in the intersection area of the h-th row has coordinates (k,h). So we have to bring the northeast corner of our $m \times n$ grid on the southeast corner of the (k,h) cell. In order to do this, we need to move our $m \times n$ grid $m-k$ steps west and $n-h+1$ steps south.

In this way, by the periodicity of our diagram, we must have reached a parking sorted configuration. This explains our formula (12.1) for u' and proves the claim that $u' = \mathsf{park}(u)$, so that $u'' = \mathsf{sort}(u') = \mathsf{sort}(\mathsf{park}(u))$.

Let us look at an example.

Example 12.2. Consider the stable sorted configuration $u = \binom{0,1,2,3,3,3;*}{2,4,4,6,6}$ on $K_{7,5}$, cf. Figure 15.

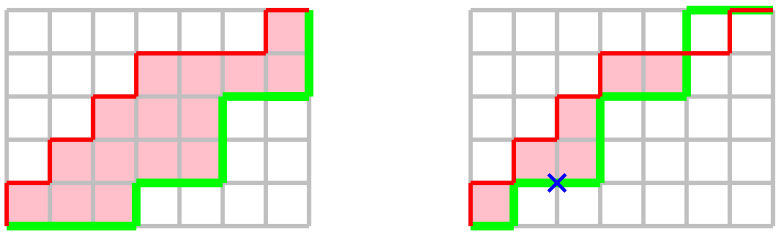

FIGURE 15. On the left, the stable sorted configuration $u = \binom{0,1,2,3,3,3;*}{2,4,4,6,6}$. On the right, the configuration $u + 2\Delta^{(a_7)} = \binom{0,1,2,3,3,3;*}{0,2,2,4,4}$.

In this case, $h=2$, $r_h=4$, and $k=2$. So $r_h-2=2$, and we have

$$\begin{aligned} u' &= u + 2\Delta^{(a_7)} - \sum_{s=2}^{6} \Delta^{(a_s)} - \sum_{t=2}^{5} \Delta^{(b_t)} \\ &= \binom{4,0,1,2,2,2;*}{5,0,0,2,2}, \end{aligned}$$

and

$$u'' = \binom{0,1,2,2,2,4;*}{0,0,2,2,5}.$$

See Figure 16.

Now that we have a formula for computing $\mathsf{sort}(\mathsf{park}(u))$, we want to count how many operations this requires.

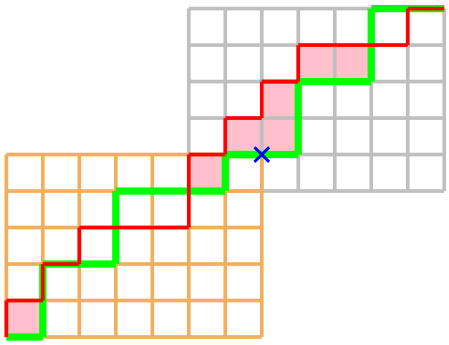

FIGURE 16. The diagram of the construction of $u'' = \binom{0,1,2,2,2,4;*}{0,0,2,2,5}$ from $u + 2\Delta^{(a_7)}$.

It is clear from the definition of the r-vector that, to compute the r-vector of a stable sorted configuration u on $K_{m,n}$, we can use the following algorithm written in pseudocode.

```
def compute_r_vector(u): # the input u = (u_{a_1},u_{a_2},...,u_{a_{m-1}};u_{a_m}) / (u_{b_1},u_{b_2},...,u_{b_n}) is stable and sorted
    out = [ ] # out is initialized as an empty list
    h = 1
    for i in [1,2,...,n]:
        while u_{a_h} + 1 ≤ i − 1 and h ≤ m − 1:
            h = h + 1
        out = out.append(u_{b_i} + 1 − (h − 1))
    return out
```

Clearly, this algorithm costs $m + n$ operations.

Moreover, we need n operations to find h, and m operations to find k. Finally, we need $2(n+m)$ operations to compute the u'_{b_i}'s and the u'_{a_j}'s. So, for computing $u'' = \mathsf{sort}(u')$, due to Step 3, or simply using formula (12.3), in total we performed again $O(m+n)$ operations.

STEP 4: Given a parking sorted configuration u, we compute $\mathsf{rank}(u)$. We can do this using the formula of Theorem 11.2: we first compute

$$u_{a_m} + 1 = nQ + R, \qquad \text{with } Q, R \in \mathbb{N}, \quad \text{and } 0 \le R < n,$$

and then

$$\mathsf{rank}(u) = -1 + \sum_{i=1}^{n} \max\{0, Q + \chi(i \le R) + r_i - 1\}.$$

All this clearly requires $O(m+n)$ operations as well.

All the discussion of this section shows that our algorithm to compute the rank of a configuration u on $K_{m,n}$ performs $O(m+n)$ operations under our assumptions on the complexity model.

13. Two new statistics on parking sorted configurations on $K_{m,n}$

We consider the generating function of parking sorted configurations on $K_{m,n}$ according to degree and rank:

$$\widetilde{K}_{m,n}(d,r) := \sum_{u \text{ parking sorted on } K_{m,n}} d^{\mathsf{degree}(u)} r^{\mathsf{rank}(u)}.$$

This formal series is not a power series since the degree takes both arbitrary positive and negative values. It encodes the distribution of the bistatistic $(\mathsf{degree}, \mathsf{rank})$. In Figure 17, we display the coefficients of $d^{\mathsf{degree}(u)} r^{\mathsf{rank}(u)}$ for the configurations u on $K_{5,3}$ with degree between -3 and 17 in a table. To get the full table, we have to add an already started half diagonal of entries equal to 105 in the top right corner and an already started horizontal half line of entries equal to 105 in the bottom left corner.

	d^{-3}	d^{-2}	d^{-1}	d^0	d^1	d^2	d^3	d^4	d^5	d^6	d^7	d^8	d^9	d^{10}	d^{11}	d^{12}	d^{13}	d^{14}	d^{15}	d^{16}	d^{17}
r^9																					105
r^8																				105	
r^7																		1	105		
r^6																	3	104			
r^5															1	8	102				
r^4														3	15	97					
r^3												1	9	27	89						
r^2										1	6	20	39	75							
r^1							1	3	9	20	36	49	57								
r^0				1	3	8	15	27	39	49	48	35									
r^{-1}	105	105	105	104	102	97	89	75	57	35	15										

FIGURE 17. Partial table of coefficients of $\widetilde{K}_{5,3}(d,r)$.

There seems to be a symmetry in the distribution of the coefficients of $\widetilde{K}_{5,3}(d,r)$. For example, we can read the values $15, 35, 57, 75, 89, 97, 102, 104, 105, 105, 105, \ldots$ in the first row towards west but also in the rightmost diagonal towards northeast. We observe similar identities in the next rows and diagonals. These observations and some additional inspections suggest the following change of parameters: we set

$$\begin{aligned} \mathsf{xpara}(u) &= (m-1)(n-1) + \mathsf{rank}(u) - \mathsf{degree}(u), \\ \mathsf{ypara}(u) &= \mathsf{rank}(u) + 1. \end{aligned}$$

This leads us to consider the formal power series

$$K_{m,n}(x,y) := \sum_u x^{\mathsf{xpara}(u)} y^{\mathsf{ypara}(u)},$$

where u still runs through parking sorted configurations of $K_{m,n}$. Observe that

(13.1)
$$K_{m,n}(x,y) = x^{(m-1)(n-1)} y \widetilde{K}_{m,n}(x^{-1}, xy) \quad \text{and} \quad \widetilde{K}_{m,n}(d,r) = d^{(m-1)(n-1)} r^{-1} K_{m,n}(d^{-1}, dr),$$

whence the study of $\widetilde{K}_{m,n}(d,r)$ is equivalent to the study of $K_{m,n}(x,y)$.

We display a partial table of the coefficients of $K_{5,3}(x,y)$ in Figure 18. Notice that now this formal power series seems to be symmetric in x and y.

y^{10}	105									
y^9	105									
y^8	105	1								
y^7	104	3								
y^6	102	8	1							
y^5	97	15	3							
y^4	89	27	9	1						
y^3	75	39	20	6	1					
y^2	57	49	36	20	9	3	1			
y^1	35	48	49	39	27	15	8	3	1	
y^0	15	35	57	75	89	97	102	104	105	105	105
	x^0	x^1	x^2	x^3	x^4	x^5	x^6	x^7	x^8	x^9	x^{10}

FIGURE 18. Partial table of coefficients of $K_{5,3}(x,y)$.

In this section, we study the formal power series $K_{m,n}(x,y)$. In particular, we want to understand the apparent symmetry in x and y, and we will eventually provide a formula for the generating function of the $K_{m,n}(x,y)$ for $m,n \geq 1$.

The key observation is that the parameters xpara and ypara have a natural interpretation in the following extension of our preceding algorithm for computing the rank. We need some more notation.

Given a configuration u on $K_{m,n}$, we denote by $u[*]$ a *partial configuration* undefined on the sink a_m. Similarly, we denote by $u[s]$ the configuration on $K_{m,n}$ whose values outside the sink a_m are the ones of u, while it takes the value s on the sink a_m. We naturally define the r-vector $r(u) := r(u[0]) = (r_1, r_2, \ldots, r_n)$ of the partial configuration $u[*]$: indeed notice that $r(u[s]) = r(u[0])$ for all $s \in \mathbb{Z}$ as clearly the r-vector does not depend on the value at the sink.

Consider the bi-infinite strip of unit square cells in the plane for which the bottom left corner has coordinates $\{(i,j) \mid i \in \mathbb{Z}, j \in \{0,1,\ldots,n-1\}\}$. This is where we draw the cylindric diagram of $u[s]$ for $s \geq 0$ (cf. Section 10). The labelling of the cells of the cylindric diagram that we used to compute the rank can be naturally extended to the negative values s on the sink a_m: the cell $L(s)$ labelled by s will have the bottom left corner coordinatized by $(u_{b_{t+1}} + q, t)$ where $s = qn + t$ with $q, t \in \mathbb{Z}$ and $0 \leq t < n$ is given by Euclidean division. We define the set $V(u[s])$ of *visited cells* by the algorithm for the parking sorted configuration $u[s]$ as all the cells labelled by a number less than or equal to s. We recall that the red path of the diagram of u disconnects the strip of the cylindric diagram into the left component and the right component.

We define $\mathsf{xpara}(u)$, respectively $\mathsf{ypara}(u)$, as the number of unvisited left cells, respectively of visited right cells.

The following lemma proves the coherence of the two given definitions of xpara and ypara.

Lemma 13.1. *For any parking sorted configuration u on $K_{m,n}$, we have*

$$\mathsf{xpara}(u) = (m-1)(n-1) + \mathsf{rank}(u) - \mathsf{degree}(u)$$

and

$$\mathsf{ypara}(u) = \mathsf{rank}(u) + 1.$$

The proof of this lemma relies on the following one, which describes the effect of the increment by one of the value on the sink by either a decrement of xpara or by an increment of ypara.

Lemma 13.2. *For any parking sorted configuration $u[s]$ on $K_{m,n}$ we have*

$$\mathsf{xpara}(u[s+1]) = \mathsf{xpara}(u[s]) - \chi(L(s+1) \text{ is at left})$$

and

$$\mathsf{ypara}(u[s+1]) = \mathsf{ypara}(u[s]) + \chi(L(s+1) \text{ is at right}).$$

Proof. The difference between the visited cells for $u[s]$ and $u[s+1]$ is the extra cell $L(s+1)$ labelled by $s+1$, which is added to $V(u[s])$ to obtain $V(u[s+1])$:

$$V(u[s+1]) = V(u[s]) \cup \{L(s+1)\}.$$

Then $L(s+1)$ is either in the left component or in the right component of the cylindric diagram, which are determined by the cut of the red path of u.

In the first case, an unvisited left cell in $u[s]$ is now visited in $u[s+1]$ hence $\mathsf{xpara}(u[s+1]) = \mathsf{xpara}(u[s]) - 1$. In the second case, the extra visited cell in $u[s+1]$ adds one visited right cell, hence $\mathsf{ypara}(u[s+1]) = \mathsf{ypara}(u[s]) + 1$. □

Proof of Lemma 13.1. The relation $\mathsf{ypara}(u) = \mathsf{rank}(u) + 1$ is clear from our description of the computation of the rank in terms of the cylindric diagram.

To understand $\mathsf{xpara}(u)$ is a bit more involved. First we observe that, for any partial configuration u, the quantity

$$Q(u) := \mathsf{degree}(u[s]) + \mathsf{xpara}(u[s]) - \mathsf{ypara}(u[s])$$

is well defined, i.e., it does not depend on s. Indeed, according to Lemma 13.2, when s is incremented by one, $\mathsf{degree}(u[s])$ is incremented by one, and exactly one of $\mathsf{xpara}(u[s])$ and $-\mathsf{ypara}(u[s])$ is decremented by one.

Hence to prove the relation $\mathsf{xpara}(u) = (m-1)(n-1) + \mathsf{rank}(u) - \mathsf{degree}(u)$ it suffices to show that $Q(u) = (m-1)(n-1) - 1$. In order to do this, we will compute $Q(u) = \mathsf{degree}(u[0]) + \mathsf{xpara}(u[0]) - \mathsf{ypara}(u[0])$.

The general argument is better understood by looking at an example: we consider the parking sorted configuration $u[0] = \binom{1,1,2,2,2,4;0}{0,0,2,2,5}$ and draw its diagram, see Figure 19 on the left.

We recall that the length of the green arrow in Figure 19 in the row i is u_{b_i}, and the length of the red arrow in column j is u_{a_j}. We assume that u is parking, hence the (pink) intersection area contains at most one (pink) cell in each row. This implies that, if one shifts the green path by one unit to the west, the green and red arrows do not intersect, cf. Figure 19 on the right. Then we partition the $(m-1)(n-1)$ cells of the orange rectangle defined by the two opposite corners $(0,1)$ and $(m-1,n)$. There are three kinds of cells:

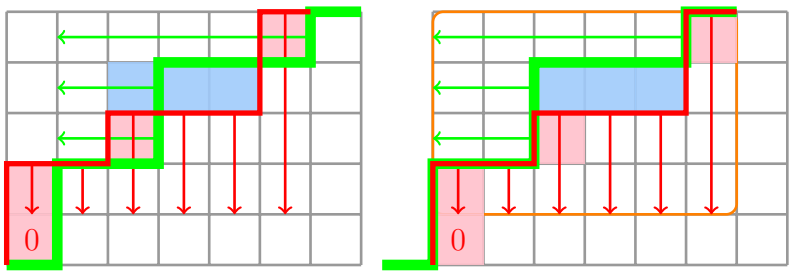

FIGURE 19. The diagram of the configuration $u[0] = \binom{1,1,2,2,2,4;0}{0,0,2,2,5}$ and its shifted version for the evaluation of $Q(u)$.

- the cells crossed by a horizontal green arrow counted by $\sum_{i=2}^{n} u_{b_i}$ (we recall that $u_{b_1} = 0$),
- the cells crossed by a vertical red arrow counted by $\sum_{j=1}^{m-1} u_{a_j}$, and
- the blue cells which are the unvisited left cells in $u[0]$, hence counted by $\mathsf{xpara}(u[0])$.

This partition of cells implies that

$$(m-1)(n-1) = \mathsf{degree}(u[0]) + \mathsf{xpara}(u[0]).$$

Since $\mathsf{rank}(u[0]) = 0$, we finally have

$$(m-1)(n-1) = \mathsf{degree}(u[0]) + \mathsf{xpara}(u[0]) - \mathsf{rank}(u[0]) = Q(u) + 1. \qquad \square$$

14. SYMMETRY OF $K_{m,n}(x, y)$

We are now in the position to prove the symmetry of $K_{m,n}(x, y)$ in x and y. It turns out that its explanation relies on the Riemann–Roch theorem for graphs. Compare the following theorem with [CL, Proposition 15 and Corollary 16].

Theorem 14.1. *For any bipartite complete graph $K_{m,n}$, we have $K_{m,n}(x, y) = K_{m,n}(y, x)$.*

Proof. The *canonical divisor* K_G on the bipartite complete graph $G = K_{m,n}$ is the configuration d such that $d_{a_i} = n - 2$ and $d_{b_j} = m - 2$ for all i and j. The Riemann–Roch theorem for graphs of Baker and Norine (cf. [BN, Theorem 1.12]) in the particular case of $K_{m,n}$ states that

$$\mathsf{rank}(u) - \mathsf{rank}(d-u) = \mathsf{degree}(u) - |E_{K_{m,n}}| + |V_{K_{m,n}}|,$$

where $|E_{K_{m,n}}| = mn$ and $|V_{K_{m,n}}| = m + n$.

We have

$$\begin{aligned} \mathsf{xpara}(d-u) &= (m-1)(n-1) + (\mathsf{rank}(d-u) - \mathsf{degree}(d-u)) \\ &= (m-1)(n-1) + (\mathsf{rank}(u) - mn + (m+n)) \\ &= \mathsf{rank}(u) + 1 = \mathsf{ypara}(u), \end{aligned}$$

where in the second equality we applied the Riemann–Roch theorem to the configuration $d - u$ in order to get $\mathsf{xpara}(d-u) = \mathsf{ypara}(u)$.

Similarly, for $u = d - (d-u)$ we obtain that $\mathsf{ypara}(d-u) = \mathsf{xpara}(u)$, hence the involution $u \leftrightarrow d - u$ shows the expected symmetry. $\square$

Remark 14.2. As on K_n [CL, Lemma 18], a *superposition principle* for the graphical interpretation for $K_{m,n}$ also gives a combinatorial proof of this symmetry. We only sketch the argument here, as the details already given in [CL] should help to fill in the gaps. It turns out that the cylindric diagram of $\mathsf{sort}(\mathsf{park}(d-u))$ may be superimposed with the cylindric diagram of $\mathsf{sort}(\mathsf{park}(u))$ with the algorithm computing the rank labelling the cells in the reverse order and the right and left components switched. There is also a shift to be fixed in this reverse labelling of cells, so that the first cell incrementing the rank from -1 is labelled by 0. It turns out that, via this involution, at any point of the algorithm, the notion of visited and unvisited cells is switched, as well as the notions of left and right cells. Hence this involution shows in terms of cells that $\mathsf{xpara}(\mathsf{sort}(\mathsf{park}(d-u))) = \mathsf{ypara}(\mathsf{sort}(\mathsf{park}(u)))$ and $\mathsf{ypara}(\mathsf{sort}(\mathsf{park}(d-u))) = \mathsf{xpara}(\mathsf{sort}(\mathsf{park}(u)))$, proving combinatorially the symmetry of $K_{m,n}(x,y)$.

15. The generating function $\mathcal{F}(x,y,w,h)$

We now turn our attention to the generating function

$$\mathcal{F}(x,y,w,h) := \sum_{n\geq 1, m\geq 1} K_{m,n}(x,y) w^m h^n.$$

In order to provide a formula for this formal power series, we start by considering the generating function

$$F_u(x,y) := \sum_{s\in\mathbb{Z}} x^{\mathsf{xpara}(u[s])} y^{\mathsf{ypara}(u[s])}$$

of $(u[s])_{s\in\mathbb{Z}}$ according to the bistatistic $(\mathsf{xpara}, \mathsf{ypara})$, where $u[*]$ is a parking sorted partial configuration on $K_{m,n}$. Notice that

$$K_{m,n}(x,y) = \sum_u F_u(x,y),$$

where the sum on the right-hand side is over all parking sorted partial configurations $u[*]$ on $K_{m,n}$.

We observe that Lemma 13.2 reveals some geometric sums which are hidden in $F_u(x,y)$. Indeed, this lemma is equivalent to saying that, if $L(s+1)$ is at the left of the cut (i.e., the red path) in the cylindric diagram of the partial configuration $u[*]$, then

$$x^{\mathsf{xpara}(u[s+1])} y^{\mathsf{ypara}(u[s+1])} = \frac{1}{x} \cdot x^{\mathsf{xpara}(u[s])} y^{\mathsf{ypara}(u[s])},$$

while, if $L(s+1)$ is at its right, then

$$x^{\mathsf{xpara}(u[s+1])} y^{\mathsf{ypara}(u[s+1])} = y \cdot x^{\mathsf{xpara}(u[s])} y^{\mathsf{ypara}(u[s])}.$$

This suggests to partition $\mathbb{Z}$ into the maximal intervals with the property that the cells in the cylindric diagram of u labelled by the elements of a given interval are all on the same side of the cut (i.e., they have all the same colour, red or green), and then to partition the series $F_u(x,y)$ into a sum of finitely many series according to those intervals. To make this more explicit, we introduce some notation.

A configuration $u[s]$ is a *positive boundary configuration* if $L(s+1)$ is at the right of the cut (of the cylindric diagram of u) while $L(s)$ is at its left. A configuration $u[s]$ is a *negative*

boundary configuration if $L(s+1)$ is at the left of the cut (of the cylindric diagram of u) while $L(s)$ is at its right.

Let $s_+^{(0)} \le s_+^{(1)} \le \cdots \le s_+^{(k)}$ be the values for which $u[s_+^{(i)}]$ is a positive boundary configuration (there are clearly finitely many), and set $S_u^+ = \{s_+^{(0)}, s_+^{(1)}, \dots, s_+^{(k)}\}$. Define similarly $S_u^- = \{s_-^{(0)}, s_-^{(1)}, \dots, s_-^{(k-1)}\}$ as the set of values for which $u[s_-^{(i)}]$ is a negative boundary configuration. Observe that the indices of the elements of S_u^- go up to $k-1$: indeed, $|S_u^+| = |S_u^-|+1$ since $L(s)$ is at the left of the cut for $-s$ large enough, while it is at its right for s large enough.

Observe that the two sets S_u^+ and S_u^- define the partition of $\mathbb{Z}$ that we mentioned above:

$$\mathbb{Z} =]-\infty, s_+^{(0)}] \sqcup \coprod_{i=0}^{k-1} \Big([s_+^{(i)}+1, s_-^{(i)}] \sqcup [s_-^{(i)}+1, s_+^{(i+1)}] \Big) \sqcup [s_+^{(k)}+1, +\infty[\,.$$

Compare the following lemma with [CL, Lemma 20].

Lemma 15.1. *For any parking sorted partial configuration $u[*]$ on $K_{m,n}$, we have*

$$F_u(x,y) = \frac{1-xy}{(1-x)(1-y)} \left(\sum_{s_+ \in S_u^+} x^{\mathsf{xpara}(u[s_+])} y^{\mathsf{ypara}(u[s_+])} - \sum_{s_- \in S_u^-} x^{\mathsf{xpara}(u[s_-])} y^{\mathsf{ypara}(u[s_-])} \right).$$

Proof. We already observed that the two sets S_u^+ and S_u^- define a partition of $\mathbb{Z}$. By Lemma 13.2, in each interval I of this partition, the sum $\sum_{s\in I} x^{\mathsf{xpara}(u[s])} y^{\mathsf{ypara}(u[s])}$ is geometric with multiplier either x, reversing the order of summation in this case, or y; in both cases it can be evaluated easily by using the boundary terms defining I.

Using the more compact notation $W_u(s) := x^{\mathsf{xpara}(u[s])} y^{\mathsf{ypara}(u[s])}$, it is easy to check that, for $i = 0, 1, \dots, k-1$,

$$\begin{aligned}
&I =]-\infty, s_+^{(0)}] && \text{gives} && \sum_{s\in I} x^{\mathsf{xpara}(u[s])} y^{\mathsf{ypara}(u[s])} = \frac{W_u(s_+^{(0)})}{1-x},\\
&I = [s_+^{(i)}+1, s_-^{(i)}-1] && \text{gives} && \sum_{s\in I} x^{\mathsf{xpara}(u[s])} y^{\mathsf{ypara}(u[s])} = \frac{yW_u(s_+^{(i)}) - W_u(s_-^{(i)})}{1-y},\\
&I = [s_-^{(i)}, s_+^{(i+1)}] && \text{gives} && \sum_{s\in I} x^{\mathsf{xpara}(u[s])} y^{\mathsf{ypara}(u[s])} = \frac{W_u(s_+^{(i+1)}) - xW_u(s_-^{(i)})}{1-x}, \text{ and}\\
&I = [s_+^{(k)}+1, +\infty[&& \text{gives} && \sum_{s\in I} x^{\mathsf{xpara}(u[s])} y^{\mathsf{ypara}(u[s])} = \frac{yW_u(s_+^{(k)})}{1-y}.
\end{aligned}$$

Now we sum these series corresponding to the intervals, and we observe that in this sum for $i = 0, 1, \dots, k$ (uniformly!) the coefficient of $W_u(s_+^{(i)})$ is

$$\frac{1}{1-x} + \frac{y}{1-y} = \frac{1-xy}{(1-x)(1-y)},$$

while for $i = 0, 1, \dots, k-1$ (still uniformly), the coefficient of $W_u(s_-^{(i)})$ is

$$-\frac{1}{1-y} - \frac{x}{1-x} = -\frac{1-xy}{(1-x)(1-y)},$$

finally leading to the desired formula. □

So Lemma 15.1 reduces the computation of $F_u(x,y)$, and hence of $\mathcal{F}(x,y,w,h)$, to the computation of the generating function of the boundary configurations. In order to understand these, we provide a description of the boundary configurations and their bistatistic $(\mathsf{xpara},\mathsf{ypara})$ in terms of pairs of binomial paths.

We define a *pair of binomial paths in the $m\times n$ grid*, or simply a *pair*, as a pair of two binomial paths in the $m\times n$ grid starting at the southwest corner: a *red path*, which consists of the shuffling of n north steps and $m-1$ east steps, and a *green path*, which consists of the shuffling n north steps and m east steps. When needed, we will identify these binomial paths with words in the alphabet $\{\mathsf{E},\mathsf{N}\}$, where E indicates an east step and N indicates a north step.

The *east suffix* of a binomial path is the longest suffix of the path (i.e., of the corresponding word in $\{\mathsf{E},\mathsf{N}\}$) that contains only east steps. A pair is a *positive boundary pair* if the red path starts with a north step, the green path starts with an east step, and the length of the east suffix of the green path is longer than or equal to the length of the east suffix of the red path. A pair is a *negative boundary pair* if the red path starts with an east step, the green path starts with a north step, and the length of the east suffix of the green path is shorter than or equal to the length of the east suffix of the red path.

In a pair p of binomial paths in the $m\times n$ grid, we consider in each row of the grid the cells between the red and the green north steps crossing that row. If the red north step is at the left of the green step, then we say that these cells belong to the xarea, otherwise we say that they belong to the yarea. Notice that the xarea corresponds to what we called the *intersection area* in previous sections. Hence $\mathsf{xarea}(p)$, respectively $\mathsf{yarea}(p)$, denote the total number of cells in the xarea, respectively in the yarea. We also denote by $\mathsf{xrow}(p)$ the number of rows where the crossing red north step is weakly to the right of the crossing green north step, and by $\mathsf{yrow}(p)$ the number of the other rows, where the crossing red north step is strictly to the left of the crossing green step.

For a positive pair p_+, we set

$$\mathsf{xpara}_+(p_+) := \mathsf{xarea}(p_+) + \mathsf{xrow}(p_+) \text{ and } \mathsf{ypara}_+(p_+) := \mathsf{yarea}(p_+) - \mathsf{yrow}(p_+),$$

and, for a negative pair p_-, we set

$$\mathsf{xpara}_-(p_-) := \mathsf{xarea}(p_-) \text{ and } \mathsf{ypara}_-(p_-) := \mathsf{yarea}(p_-).$$

Example 15.2. Consider the pair p of binomial paths in Figure 20. Here we have a 7×5

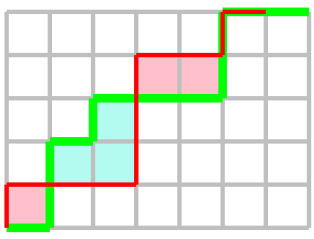

FIGURE 20. A positive boundary pair in the 7×5 grid.

grid, the red path is NEEENNNEENE, and the green path is ENNENEEENNEE. Observe that the red path starts with a north step, and its east suffix is E, while the green path starts with an east step, and its east suffix is EE. Hence p is a positive boundary pair. The xarea corresponds to the pink cells, hence $\mathsf{xarea}(p) = 3$. The yarea corresponds to the blue cells, hence $\mathsf{yarea}(p) = 3$. Observe also that $\mathsf{xrow}(p) = 3$ and $\mathsf{yrow}(p) = 2$, hence $\mathsf{xpara}_+(p) = \mathsf{xarea}(p) + \mathsf{xrow}(p) = 6$, while $\mathsf{ypara}_+(p) = \mathsf{yarea}(p) - \mathsf{yrow}(p) = 1$.

Remark 15.3. Notice that every positive boundary pair corresponds to a certain stable sorted configuration: for example, the pair in Figure 20 corresponds to the stable sorted configuration $u = \binom{0,0,0,3,3,4;*}{0,0,1,4,4}$. Notice that in this case u is stable but not parking, as it has two cells of its intersection area in the same row.

The following lemma explains our optimistic notations.

Lemma 15.4. *For any complete bipartite graph $K_{m,n}$ there exist two bijections, one denoted by f_+, from the positive boundary configurations into the positive boundary pairs (in the $m \times n$ grid), and another one denoted by f_-, from the negative boundary configurations into the negative boundary pairs, with the following properties:*

(1) *for any positive boundary configuration $u[s_+]$, if we set $p_+ = f_+(u[s_+])$, then*
$$(\mathsf{xpara}(u[s_+]), \mathsf{ypara}(u[s_+])) = (\mathsf{xpara}_+(p_+), \mathsf{ypara}_+(p_+));$$
(2) *for any negative boundary configuration $u[s_-]$, if we set $p_- = f_-(u[s_-])$, then*
$$(\mathsf{xpara}(u[s_-]), \mathsf{ypara}(u[s_-])) = (\mathsf{xpara}_-(p_-), \mathsf{ypara}_-(p_-)).$$

In fact, in order to prove the lemma, we can define the maps f_+ and f_- explicitly. Those bijections essentially consist in identifying the appropriate intersections of the red and green periodic paths of $u[*]$ embedded in $\mathbb{Z}^2$, each such intersection being related simultaneously to one boundary pair and one boundary configuration.

In a pair of periodic binomial paths in $\mathbb{Z}^2$, we define a *positive boundary intersection* i_+ to be a vertex of $\mathbb{Z}^2$ common to the red and the green periodic paths such that:

(1) i_+ is followed (going from southwest towards northeast) by a north red step and an east green step, and
(2) the last green north step before i_+ is weakly to the left of the red north step crossing the same row.

Similarly, a *negative boundary intersection* i_- is a vertex common to the red and green periodic paths such that:

(1) i_- is followed by an east red step and a north green step, and
(2) the last green north step before i_- is weakly to the right of the red north step crossing the same row.

Our construction is better understood by looking at an example.

Example 15.5. In Figure 21, we draw the periodic diagram of the parking sorted configuration $u = \binom{0,0,0;*}{0,0,1}$ on $K_{4,3}$, whose diagram can be recognized in the lowest blue dashed 4×3 grid. In the picture, we included the labelling of the cells to the left of the green north steps: recall that this diagram with this labelling is simply an unrolled version of our cylindric

diagram of u. Moreover, notice that, for graphical practicality, we wrote $\overline{n}$ instead of $-n$ for $n \in \mathbb{N} \setminus \{0\}$.

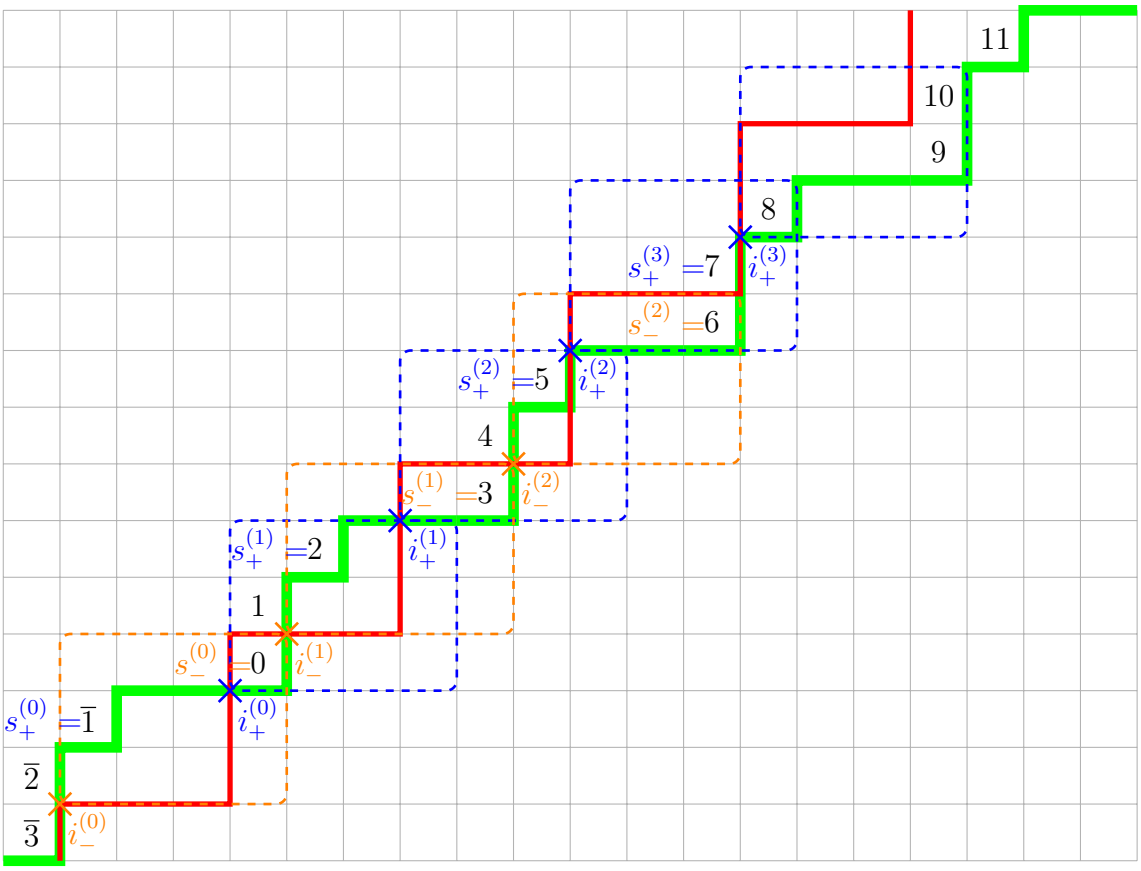

FIGURE 21. Boundary intersections and related boundary configurations and pairs.

Observe that we labelled the positive boundary intersections by $\{i_+^{(0)}, i_+^{(1)}, i_+^{(2)}, i_+^{(3)}\}$, and we marked them by a blue cross, while we labelled the negative boundary intersections by $\{i_-^{(0)}, i_-^{(1)}, i_-^{(2)}\}$, and we marked them by an orange cross.

Following the black labels of cells along the green path, we observe that

$$S_u^+ = \{s_+^{(0)}, s_+^{(1)}, s_+^{(2)}, s_+^{(3)}\} = \{-1, 2, 5, 7\} \text{ and } S_u^- = \{s_-^{(0)}, s_-^{(1)}, s_-^{(2)}\} = \{0, 3, 6\}.$$

It is clear from the definitions (and it can be checked in our example) that the value $s_+^{(k)}$, defining the positive boundary configuration $u[s_+^{(k)}]$, is the label of the cell left to the north green step crossing the row just below $i_+^{(k)}$. We define $f_+(u[s_+^{(k)}])$ to be the pair of binomial paths read in the $m \times n$ grid whose southwest corner is $i_+^{(k)}$ (the blue dashed rectangle in our picture).

We claim that $f_+(u[s_+^{(k)}])$ is indeed a positive boundary pair. Indeed, the conditions on the initial red and green steps follow immediately from condition (1) of the definition of $i_+^{(k)}$. Notice that the other condition to be a positive boundary pair, i.e., that the length of the

east suffix of the green path should be longer than or equal to the length of the east suffix of the red path, can be translated to saying that in the top row of the diagram of the pair the red north step should be at most a unit step to the left of the green north step in the same row. Using the periodicities of the green and the red paths, this is equivalent to condition (2) of the definition of $i_+^{(k)}$ (cf. the diagram of Figure 22), and this proves the claim.

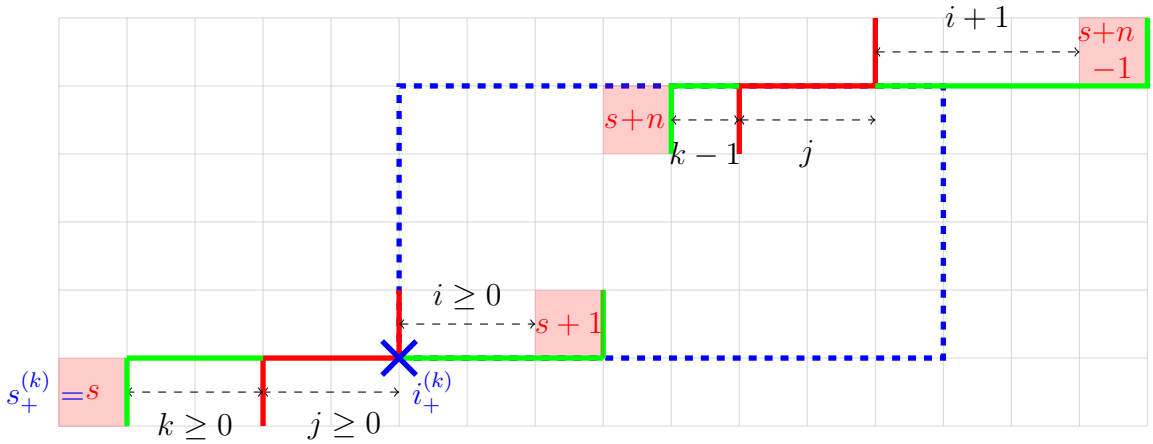

FIGURE 22. Diagram of $f_+(u[s_+^{(k)}])$.

Similarly, the value $s_-^{(k)}$, defining the negative boundary configuration $u[s_-^{(k)}]$, is the label of the cell left to the north green step that crosses the n-th row above $i_-^{(k)}$ (the first one being in the row just above $i_-^{(k)}$). We define $f_-(u[s_-^{(k)}])$ to be the pair of binomial paths read in the $m \times n$ grid whose southwest corner is $i_-^{(k)}$ (the dashed orange rectangle in our picture).

We claim that $f_-(u[s_-^{(k)}])$ is indeed a negative boundary pair: the proof is similar to the one for f_+ and it is left to the reader (cf. the diagram of Figure 23).

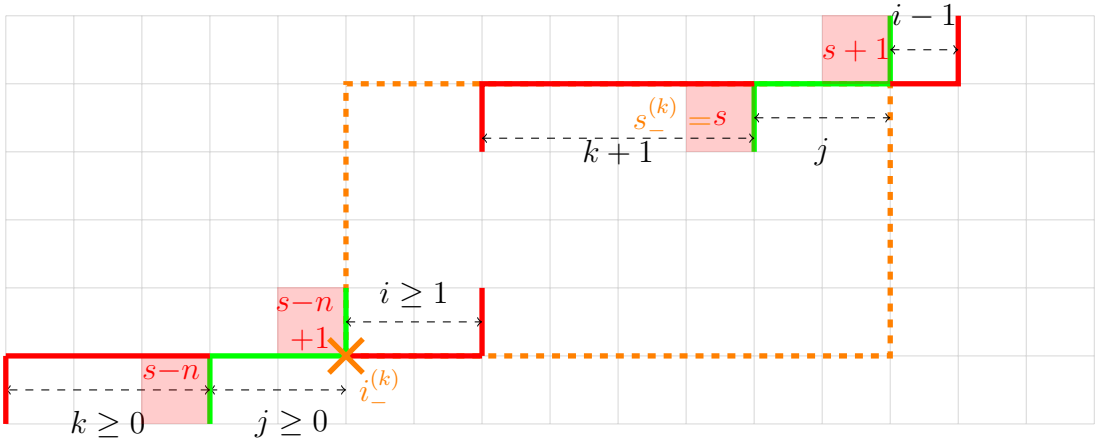

FIGURE 23. Diagram of $f_-(u[s_-^{(k)}])$.

Now that we defined our maps f_+ and f_- we can turn to the proof that they have all the claimed properties.

Proof of Lemma 15.4. We first discuss the bijectivity of f_+ and f_-. For both, we rely on some results from [ADDL]. Indeed, in [ADDL, Proposition 5.12] it was shown that, given a pair of binomial paths in the $m \times n$ grid, if we look at the corresponding periodic diagram, and we move an $m \times n$ grid with its southwest corner anchored to the green path, we always pass by the diagrams of m distinct stable sorted configurations; moreover all the stable sorted configurations occur in precisely one of these periodic diagrams (in fact, this is the essence of the Cyclic Lemma in [ADDL, Lemma 3.1]).

We already observed that $f_+(u[s_+^{(k)}])$ is a positive boundary pair, and that such pairs correspond to stable sorted configurations (cf. Remark 15.3). By construction, $f_+(u[s_+^{(k)}])$ is obtained by looking at the periodic diagram of the parking sorted configuration u, and then by moving along the diagram according to $s_+^{(k)}$ in order to get the corresponding pair. By the results mentioned above, starting with distinct parking sorted partial configurations $u[*]$ and $u'[*]$, we will necessarily eventually arrive at distinct periodic diagrams, hence at distinct pairs, while for $k \neq j$ the configurations $f_+(u[s_+^{(k)}])$ and $f_+(u[s_+^{(j)}])$ will be two of the stable configurations of the periodic diagram of the same $u[*]$, hence once again they will be distinct. This proves the injectivity of f_+. For the surjectivity, we can again use the same results: given a positive boundary pair, this corresponds to a stable sorted configuration, hence looking at the corresponding periodic diagram we can deduce the corresponding parking sorted configuration $u[*]$, and finally the correct value s_+ for the sink. This finishes the proof of the bijectivity of f_+.

The proof of the bijectivity of f_- is analogous: we need only to observe that a negative boundary pair of binomials paths contains necessarily a positive boundary intersection (cf. the diagram of Figure 23). So, in the periodic diagram of our pair, we can move the southwest corner of the $m \times n$ grid on the lowest of such positive boundary intersections, and this by definition gives us a positive boundary pair. By the periodicity of the red and the green paths, such a pair is uniquely determined by our original pair. Now the same argument that we used to prove the bijectivity of f_+ gives us also the bijectivity of f_-.

It remains to show the coincidence of the bistatistics $(\mathsf{xpara}, \mathsf{ypara})$ for $u[s_\pm^{(k)}]$ and $(\mathsf{xpara}_\pm, \mathsf{ypara}_\pm)$ for $f_\pm(u[s_\pm^{(k)}])$.

In order to read $(\mathsf{xpara}(u[s_+^{(k)}]), \mathsf{ypara}(u[s_+^{(k)}]))$ from the positive boundary pair $f_+(u[s_+^{(k)}])$, we will use a local cylindric labelling of the diagram of the pair. The construction is better understood in an example: consider the diagram to the left of Figure 24. This is the diagram of a positive boundary pair. We labelled the cells of this diagram in the usual cylindric way, following the green path, starting with 0 in the cell to the left of the lowest north green step. We already observed that this cell is the one labelled by $s_+^{(k)} + 1$ in the periodic diagram of the corresponding parking sorted configuration (cf. the diagram of Figure 22). Again, we write $\overline{n}$ for $-n$ for all $n \in \mathbb{N} \setminus \{0\}$.

The general remark is that this new cylindric diagram still allows us to compute xpara and ypara of the original parking sorted configuration $u[s_+^{(k)}]$: indeed, it turns out that in this way we labelled the cells visited by the algorithm computing the rank of this configuration

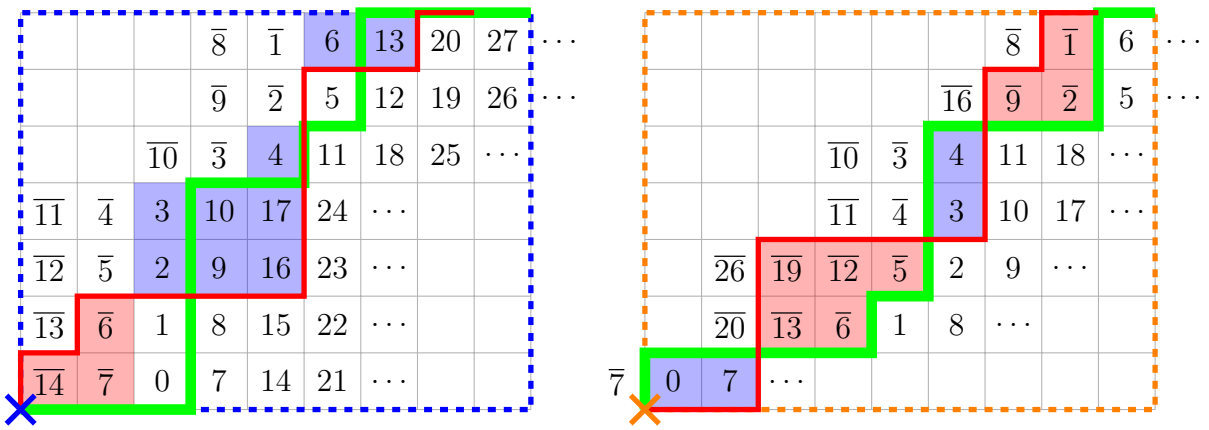

FIGURE 24. Local cyclindric labelling of (positive then negative) boundary pairs

by negative values in cylindric fashion, and the unvisited cells by non-negative values. The periodicity of the diagram implies now that $\mathsf{xpara}(u[s_+^{(k)}])$ is equal to the number of cells to the left of the red path that we now labelled by non-negative numbers (blue in our picture), while $\mathsf{ypara}(u[s_+^{(k)}])$ is the number of cells to the right of the red path that we labelled by negative numbers (pink in our picture).

It is now clear (and it can be checked in our example) that the $\mathsf{xpara}(u[s_+^{(k)}])$ unvisited cells (hence with non-negative labels) to the left of the red path are also counted by $\mathsf{xarea}(p_+) + \mathsf{xrow}(p_+) = \mathsf{xpara}_+(p_+)$, and that the $\mathsf{ypara}(u[s_+^{(k)}])$ visited cells (hence with negative labels) to the right of the red path are also counted by $\mathsf{yarea}(p_+) - \mathsf{yrow}(p_+) = \mathsf{ypara}_+(p_+)$.

In order to read $(\mathsf{xpara}(u[s_-^{(k)}]), \mathsf{ypara}(u[s_-^{(k)}]))$ from the negative boundary pair $f_-(u[s_-^{(k)}])$, we can use a similar construction, but with a different convention. Consider the diagram to the right of Figure 24. This is the diagram of a negative boundary pair. We labelled the cells of this diagram in the usual cylindric way, following the green path, starting with $\overline{1} = -1$ in the cell to the left of the highest green north step. We already observed that this cell is the one labelled by $s_-^{(k)}$ in the periodic diagram of the corresponding parking sorted configuration (cf. the diagram of Figure 23). Again, we write $\overline{n}$ for $-n$ for all $n \in \mathbb{N} \setminus \{0\}$.

As before, also in this case the negative labels correspond to cells visited by the algorithm that computes the rank of $u[s_-^{(k)}]$, while the non-negative labels correspond to the unvisited cells.

It is again clear (and it can be checked in our example) that the $\mathsf{xpara}(u[s_-^{(k)}])$ unvisited cells to the left of the red path (blue in our picture) are also counted by $\mathsf{xarea}(p_-) = \mathsf{xpara}_-(p_-)$, and the $\mathsf{ypara}(u[s_-^{(k)}])$ visited cells to the right of the red path (pink in our picture) are also counted by $\mathsf{yarea}(p_-) = \mathsf{ypara}_-(p_-)$.

This completes the proof of Lemma 15.4. □

Now that we identified the bistatistics on boundary configurations with the ones on boundary pairs, we need only one last ingredient to get our formula for $\mathcal{F}(x, y, w, h)$: parallelogram polyominoes.

A *parallelogram* (or *staircase*) *polyomino* can be defined as the *non-empty* union of unit squares enclosed by a pair of binomial paths in a grid that cross only in the southwest corner and in the northeast corner of the grid. Naturally, the area of such a polyomino is defined as the number of these unit squares, its width is the the width of the bounding box, and its height is the height of the bounding box.

Remark 15.6. Notice that Proposition 6.4 says that the recurrent sorted configurations on $K_{m,n}$ are the stable sorted configurations on $K_{m,n}$ whose diagram is a polyomino of width m and height n.

In [BV], the generating function of parallelogram polyominoes according to their area (counted by q), their width (counted by w), and their height (counted by h)

$$P(q; w, h) := \sum_P q^{\mathsf{area}(P)} w^{\mathsf{width}(P)} h^{\mathsf{height}(P)}$$

is shown to be equal to

$$P(q; w, h) = qwh \frac{L(qw, qh)}{L(w, h)},$$

where

$$L(w, h) := \sum_{n \geq 0, m \geq 0} \frac{(-1)^{m+n} h^n w^m q^{\binom{m+n+1}{2}}}{(q)_n (q)_m},$$

and $(a)_n := \prod_{i=0}^{n-1} (1 - q^i a)$.

We want to prove the following formula (compare it with [CL, Theorem 21]).

Theorem 15.7. *We have*

$$\mathcal{F}(x, y, w, h) = \frac{(1 - xy)(hw - P(x; w, h)P(y; w, h))}{(1 - x)(1 - y)(1 - h - w - P(x; w, h) - P(y; w, h))}.$$

Remark 15.8. Notice that from the formula of Theorem 15.7 the symmetries of $\mathcal{F}(x, y, w, h)$ in x and y (proved combinatorially in Theorem 14.1) as well as in w and h (immediate from (13.1) and the obvious symmetry $\widetilde{K}_{m,n}(d, r) = \widetilde{K}_{n,m}(d, r)$), are both apparent.

We set

$$K^+_{m,n}(x, y) := \sum_u x^{\mathsf{xpara}(u)} y^{\mathsf{ypara}(u)},$$

where the sum is taken over all positive boundary configurations u on $K_{m,n}$, and similarly

$$K^-_{m,n}(x, y) := \sum_u x^{\mathsf{xpara}(u)} y^{\mathsf{ypara}(u)}$$

where the sum is taken over all negative boundary configurations u on $K_{m,n}$.

Lemma 15.9. *We have*

$$\sum_{n\geq 1,m\geq 1} K^{+}_{m,n}(x,y)w^m h^n = \frac{(1-hx-w)hw+(w-h)P(x;w,xh)P(y;w,h/y)}{(1-w)(1-w-xh-P(y;w,h/y)-P(x;w,xh))} + \frac{(1-hx-w+xw)hP(y;w,h/y)-hwP(x;w,xh)}{(1-w)(1-w-xh-P(y;w,h/y)-P(x;w,xh))}$$

and

$$\sum_{n\geq 1,m\geq 1} K^{-}_{m,n}(x,y)w^m h^n = \frac{P(x;w,h)P(y;w,h)}{(1-w)(1-h-w-P(x;w,h)-P(y;w,h))},$$

where $P(q;w,h)$ is the previously defined generating function for parallelogram polyominoes according to area*,* width*, and* height*.*

Proof. We use the notation of regular (formal) languages: given a *language* L, i.e., a set of words in some *alphabet*, we denote by L^* the language of all possible concatenations of a finite (possibly empty) sequence of words each coming from the language L. In particular, we write a language L as the formal sum of its words: so, for example, if $L = \{010, 101, 11011\}$ is our language in the alphabet $\{0,1\}$, then we identify L with $010 + 101 + 11011$, and we write $(010 + 101 + 11011)^*$ for L^*. Finally, given two languages L and K, we write LK for the language of all possible concatenations of a word v from L with a word w from K.

Consider the diagram of a boundary pair, with the red path incremented by a final east step. Such a pair can be decomposed into factors made up of superimposed red and green north (▮) or east (▬) steps, or parallelogram polyominoes whose upper path is either red (□), which we call *red polyominoes*, or green (□), which we call *green polyominoes*. The constraints on prefixes or suffixes of these paths coming from the definition of positive or negative boundary pairs may be translated into this decomposition. We want to express the languages of positive and negative boundary pairs in terms of these languages.

We denote by □ the language consisting of (i.e., the formal sum of) all the red polyominoes. Similarly, we denote by □ the language consisting of all the green polyominoes. Here ▮ and ▬ denote the languages consisting just of a superimposed red and green north step and a superposed red and green east step, respectively.

We discuss first the case of positive boundary pairs.

The first red step is north and the first green step is east, hence a positive boundary pair always starts with a red polyomino. The constraint on the lengths of east suffixes is taken into account by a maximal suffix of superimposed red and green east steps. In addition, we have to discuss if the factor just before the maximal suffix in (▬)* is a red polyomino (□) or something else (▮ or □).

◦ If the factor just before the maximal suffix in (▬)* is a red polyomino, then its topmost row contains exactly one cell: we denote the language consisting of such polyominoes by □. We then have to discuss if the first and last red polyominoes are the same or if they are distinct.

If they are the same, we are led to

□(▬)*,

otherwise to

□(▬ + ▮ + □ + □)*□(▬)*.

◦ If the last factor before the maximal suffix in $(\_)^*$ is either | or □, then, by the periodicity of the red path, the maximal suffix in $(\_)^*$ must have length at least 1. There are no further constraints, hence the language of these pairs can be described as

$$□(\_ + | + □ + □)^*(| + □)\_(\_)^*.$$

In conclusion, the language of positive boundary pairs can be described as

$$□(\_)^* + □(\_ + | + □ + □)^*□(\_)^* + □(\_ + | + □ + □)^*(| + □)\_(\_)^*. \tag{15.1}$$

We discuss now the case of negative boundary pairs.

Its first red step is east and the first green step is north, hence a negative boundary pair starts by a green polyomino. The constraint on the lengths of east suffixes is taken into account by the maximal suffix in $(\_)^*$ and the fact that there is a final red polyomino just before this suffix. Therefore all the negative pairs are described by

$$□(\_ + | + □ + □)^*□(\_)^*. \tag{15.2}$$

Now that the positive and negative pairs are described by such non-ambiguous regular expressions, it remains to compute the generating functions that take into account the parameters $(\mathsf{xpara}_+, \mathsf{ypara}_+)$, respectively $(\mathsf{xpara}_-, \mathsf{ypara}_-)$, and of course the width and the height of the grids.

The case of negative boundary pairs p_- is easier: according to the end of the proof of Lemma 15.4, the area of red polyominoes counts exactly all the contributions to the statistic $\mathsf{ypara}_-(p_-)$, while the area of green polyominoes describes exactly all the contributions to the statistic $\mathsf{xpara}_-(p_-)$.

Hence our generating function for □ is $P(x; w, h)$, for □ is $P(y; w, h)$, and trivially for _ and | is simply w and h respectively. All this, together with (15.2), gives the claimed

$$\begin{aligned}\sum_{n\geq 1, m\geq 1} & K^-_{m,n}(x,y)w^m h^n \\ &= P(x;w,h)\frac{1}{1-(w+h+P(x;w,h)+P(y;w,h))}P(y;w,h)\frac{1}{1-w} \\ &= \frac{P(x;w,h)P(y;w,h)}{(1-w)(1-w-h-P(x;w,h)-P(y;w,h))}.\end{aligned}$$

The case of positive boundary pairs p_+ is slightly more complicated since, still according to the end of the proof of Lemma 15.4, the area of red polyominoes must be decremented by one on each row for the contribution of $\mathsf{ypara}_+(p_+)$, while the area of green polyominoes and vertical superimposed green and red north steps must be incremented by one on each row for the contribution of $\mathsf{xpara}_+(p_+)$. These modifications may be taken into account by suitable changes of variables, counting the height as follows: our generating function for □ is $P(x; w, xh)$, for □ is $P(y; w, h/y)$, for _ is just w, and for | is just xh. In order to compute the generating function of □, we observe that adding one extra top row of one cell to a parallelogram polyomino is unambiguous, hence its generating function is $h(w +$

$P(y;w,h/y)$). All this together with (15.1) gives the claimed

$$\begin{aligned}
&\sum_{n\geq 1,m\geq 1} K^{+}_{m,n}(x,y)w^m h^n \\
&\quad = h(w+P(y;w,h/y))\frac{1}{1-w} \\
&\qquad + P(y;w,h/y)\frac{1}{1-w-xh-P(y;w,h/y)-P(x;w;xh)}h(w+P(y;w,h/y))\frac{1}{1-w} \\
&\qquad + P(y;w,h/y)\frac{1}{1-w-xh-P(y;w,h/y)-P(x;w;xh)}(xh+P(x;w,xh))\frac{w}{1-w} \\
&\quad = \frac{(1-hx-w)hw+(w-h)P(x;w,xh)P(y;w,h/y)}{(1-w)(1-w-xh-P(y;w,h/y)-P(x;w,xh))} \\
&\qquad + \frac{(1-hx-w+xw)hP(y;w,h/y)-hwP(x;w,xh)}{(1-w)(1-w-xh-P(y;w,h/y)-P(x;w,xh))}.
\end{aligned}$$

□

We are now in the position to prove Theorem 15.7.

Proof of Theorem 15.7. In the formulae of Lemma 15.9, we can remove the substitutions in the variable h by using the following decomposition of parallelogram polyominoes.

We look at the lowest occurrence of two rows of unit cells of the parallelogram polyomino whose intersection consists of a single unit edge.

If there is no such pair of rows, then either this is the one cell polyomino, counted by whq, or we can remove the first cell of each row and still get a parallelogram polyomino: these polyominoes are counted by $wP(q;w,qh)$.

If there is such a pair of rows, then our polyomino decomposes into two parallelogram polyominoes: the first one consists of the rows below (and including) the lowest row of our pair, while the second one is an unrestricted parallelogram polyomino. Hence these polyominoes are counted by $(qwh+wP(q;w,qh))\frac{P(q;w,h)}{w}$.

This decomposition leads to the identity

$$P(q;w,h) = (qh+P(q;w,qh))(w+P(q;w,h)).$$

From this equation, we deduce that

$$P(q;w,h) = \frac{w(qh+P(q;w,qh))}{1-qh-P(q;w,qh)} \text{ and } P(q;w,qh) = \frac{(1-qh)P(q;w,h)-hqw}{w+P(q;w,h)}.$$

The substitution $q=x$ in the second equation gives an expression for $P(x;w,xh)$ in terms of $P(x;w,h)$, namely

$$P(x;w,xh) = \frac{(1-xh)P(x;w,h)-hxw}{w+P(x;w,h)},$$

while the substitutions $q = y$ and $h = h/y$ in the first equation give an expression for $P(y; w, h/y)$ in terms of $P(y; w, h)$, namely

$$P(y; w, h/y) = \frac{w(h + P(y; w, h))}{1 - h - P(y; w, h)}.$$

Using these formulae, Lemma 15.1, and Lemma 15.9, we get

$$\begin{aligned}
\mathcal{F}(x, y, w, h) &= \sum_{n\geq 1, m\geq 1} K_{m,n}(x, y) w^m h^n \\
&= \frac{1 - xy}{(1 - x)(1 - y)} \left(\sum_{n\geq 1, m\geq 1} K^+_{m,n}(x, y) w^m h^n - \sum_{n\geq 1, m\geq 1} K^-_{m,n}(x, y) w^m h^n \right) \\
&= \frac{1 - xy}{(1 - x)(1 - y)} \left(\frac{(1 - hx - w)hw + (w - h)P(x; w, xh)P(y; w, h/y)}{(1 - w)(1 - w - xh - P(y; w, h/y) - P(x; w, xh))} \right. \\
&+ \frac{(1 - hx - w + xw)hP(y; w, h/y) - hwP(x; w, xh)}{(1 - w)(1 - w - xh - P(y; w, h/y) - P(x; w, xh))} \\
&\left. - \frac{P(x; w, h)P(y; w, h)}{(1 - w)(1 - h - w - P(x; w, h) - P(y; w, h))} \right) \\
&= \frac{(1 - xy)(hw - P(x; w, h)P(y; w, h))}{(1 - x)(1 - y)(1 - h - w - P(x; w, h) - P(y; w, h))}.
\end{aligned}$$

This completes the proof of the theorem. □

References

[ADDL] Aval J.-C., D'Adderio M., Dukes M., Le Borgne Y., *Two operators on sandpile configurations, the sandpile model on the complete bipartite graph, and a cyclic lemma*, Adv. Appl. Math. **73** (2016), 59–98.

[BTW] Bak P., Tang C., Wiesenfelds K., *Self-organized criticality: an explanation of* $1/f$ *noise*, Physical Review Letters **59**(4) (1987), 381–384.

[BN] Baker M., Norine S., *Riemann–Roch and Abel–Jacobi theory on a finite graph*, Adv. Math. **215** (2007), 766–788.

[BLS] Björner A., Lovász L., Shor P. W., *Chip-firing games on graphs*, Europ. J. Combin. **12** (1991), 283–291.

[BV] Bousquet-Mélou M., Viennot X., *Empilements de segments et q-énumération de polyominos convexes dirigés*, J. Combin. Theory Ser. A **60** (1992), 196–224.

[C] Caporaso L., *Rank of divisors on graphs: an algebro-geometric analysis*, A celebration of algebraic geometry, Clay Math. Proc., vol. 18, Amer. Math. Soc., Providence, R.I, 2013, pp. 45–64.

[CL] Cori R., Le Borgne Y., *On computation of Baker and Norine's rank on complete graphs*, Electron. J. Combin. **23**(1) (2016), Paper 1.31, 47 pp.

[CR] Cori R., Rossin D., *On the sandpile group of dual graphs*, Europ. J. Combin. **21** (2000), 447–459.

[CLRS] Cormen T. H., Leiserson C. E., Rivest R. L., Stein C., *Introduction to Algorithms*, 2nd ed., MIT Press and McGraw–Hill, 2001.

[D1] Dhar D., *Self-organized critical state of the sandpile automaton models*, Phys. Rev. Lett. **64** (1990), 1613–1616.

[D2] Dhar D., *Theoretical studies of self-organized criticality*, Physica A **369** (2006) (1).

[DL] Dukes M., Le Borgne Y., *Parallelogram polyominoes, the sandpile model on a complete bipartite graph, and a* q, t*-Narayana polynomial*, J. Combin. Theory Ser. A **120** (2013), 816–842.

[HKN] Hladký J., Král' D., Norine S., *Rank of divisors on tropical curves*, J. Combin. Theory Ser. A **120** (2013) no. 7, 1521–1538.

[KT] Kiss V., Tóthmérész L., *Chip-firing games on Eulerian digraphs and NP-hardness of computing the rank of a divisor on a graph*, Discrete Appl. Math. **193** (2015), 48–56.

[M] Merino C., *The chip-firing game*, Discrete Math. **302** (2005), 188–210.

Université Libre de Bruxelles (ULB), Département de Mathématique, Boulevard du Triomphe, B-1050 Bruxelles, Belgium

E-mail address: mdadderi@ulb.ac.be

LaBRI, Université de Bordeaux, CNRS, 351 cours de la Libération, 33405 Talence, France

E-mail address: yvan.le-borgne@u-bordeaux.fr