

Highlights

Combinatorial and Asymptotic Results on the Neighborhood Grid

Alex McDonough, Ulrich Reitebuch, Martin Skrodzki

- Discussion of the *Neighborhood Grid* spatial data structure and its *stable states* from a combinatorial viewpoint.
- Proof that any d -dimensional point set has a *stable state* even for grids with varying side lengths. Additionally, these states can be constructed in polynomial time (Theorem 3.1).
- Proof of a lower bound on the run time of this construction (Theorem 4.3) and prove time optimality of the construction algorithm for the case where all side lengths of the grid are equal (Corollary 4.4).
- Disproving the implicit assumption of previous work by showing that all point sets placed in grids of size 4×4 or larger admit to at least two stable states (Propositions 5.5 and 5.6).
- Bounding the number of stable states that a point set admits to (Proposition 6.8).
- Conjecture a classification for the point sets with maximally many stable states for all 2-dimensional grids (Conjecture 6.15).

Combinatorial and Asymptotic Results on the Neighborhood Grid

Alex McDonough^a, Ulrich Reitebuch^b and Martin Skrodzki^{c,*}

^aUniversity Of Oregon, Eugene, OR, USA

^bFU Berlin, Mathematics and Computer Science, Berlin, Germany

^cComputer Graphics and Visualization, TU Delft, Delft, Netherlands

ARTICLE INFO

Keywords:

Neighborhood Grid

Uniqueness

Asymptotic

Young Tableaux

Spatial Data Structure

ABSTRACT

In various application fields, such as fluid-, cell-, or crowd-simulations, spatial data structures are very important. They answer nearest neighbor queries which are instrumental in performing necessary computations for, e.g., taking the next time step in the simulation. Correspondingly, various such data structures have been developed, one being the *neighborhood grid*.

In this paper, we consider combinatorial aspects of this data structure. Particularly, we show that an assumption on uniqueness, made in previous works, is not actually satisfied. We extend the notions of the neighborhood grid to arbitrary grid sizes and dimensions and provide two alternative, correct versions of the proof that was broken by the dissatisfied assumption.

Furthermore, we explore both the uniqueness of certain states of the data structure as well as when the number of these states is maximized. We provide a partial classification by using the hook-length formula for rectangular Young tableaux. Finally, we conjecture how to extend this to all 2-dimensional cases.

1. Introduction

One of the most fundamental tasks in data processing, modeling, or simulations is finding answers to nearest neighbor queries. These queries are to identify those points from a given data set that are closest or close enough to a certain query point. Application examples include particle-, cell-, or crowd-simulations, where the movement of the particles, cells, or humans in the next time step is determined by the current placement and movement of their respective nearest neighbors. As these queries have to be answered with respect to some embedding space, so-called *spatial data structures* are designed to answer them. The main metric to be optimized for these answers is time, as quick answers allow more simulation steps.

The *Monotonic Logical Grid* (MLG)¹ is one such data structure that was developed specifically for large-scale simulations of molecular and fluid dynamics systems Boris (1986). As opposed to, e.g., *k-d trees* Skrodzki (2019a), the MLG does not answer neighborhood queries exactly. That is, the MLG only provides approximate answers of neighbors that are somewhat close to the query point while not necessarily being the closest. For the MLG, no guarantee can be given on the quality of the provided neighborhoods. However, it can be implemented easily on highly-parallelized hardware and therefore computes its answers faster than, e.g., *k-d trees*, making it a suitable alternative for applications where speed trumps exactness. See the article of Uhlmann Uhlmann (2020) for an overview of the history of the MLG.

More recently, the MLG was rediscovered by researchers who were investigating crowd simulations Joselli, Passos, Zamith, Clua, Montenegro and Feijó (2009) and fluid animations Joselli, Junior, Clua, Montenegro, Lage and Pagliosa (2015). They coined the data structure *Neighborhood Grid*, a terminology which we will use for the remainder of this paper. Their interest was sparked by the need for fast answers in their simulations. Hence, a highly-parallelized *CUDA* implementation on a graphics card that provided fast, approximate answers was exactly what was necessary for their applications.

To illustrate the neighborhood grid and how it is used in an application scenario, we will give a brief example. A more formalized description is given in Section 2.

*Corresponding author

✉ alexmcd@uoregon.edu (A. McDonough); ulrich.reitebuch@fu-berlin.de (U. Reitebuch);
mail@ms-math-computer.science (M. Skrodzki)

🌐 <https://ms-math-computer.science> (M. Skrodzki)

ORCID(s): 0000-0002-8126-0511 (M. Skrodzki)

🐦 <https://twitter.com/msmathcomp> (M. Skrodzki)

🌐 <https://www.linkedin.com/profile/view?id=dr-martin-skrodzki-7a80b5189> (M. Skrodzki)

¹Some authors refer to it as the *Monotonic Lagrangian Grid*.

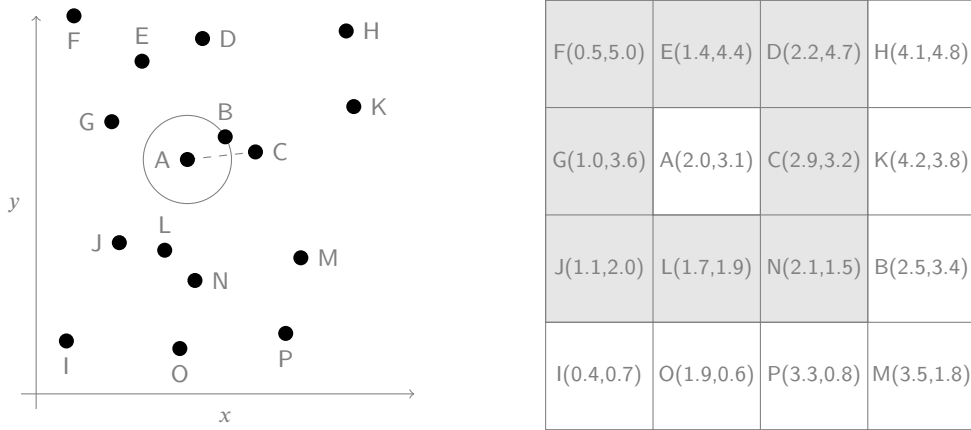


Figure 1: Sixteen points A to P in two-dimensional Euclidean space, note how the closest point to query point A is point B . The points are moved into a 4×4 grid in such a way that the rows of the grid are increasing in x -value from left to right while the columns of the grid are increasing in y -value from bottom to top, which forms a *stable state* (see Definition 2.1). Now, the closest point in the one-ring around a query point (a gray one-ring around query point A is shown) is taken as approximate nearest neighbor to the query point. Here, the closest point in the one-ring is C , which is not the actual nearest neighbor B , but a good approximation.

Example 1.1. Consider the set of 16 two-dimensional points in Euclidean space

$$A(2.0, 3.1), B(2.5, 3.4), C(2.9, 3.2), D(2.2, 4.7), E(1.4, 4.4), F(0.5, 5.0), G(1.0, 3.6), H(4.1, 4.8), \\ I(0.4, 0.7), J(1.1, 2.0), K(4.2, 3.8), L(1.7, 1.9), M(3.5, 1.8), N(2.1, 1.5), O(1.9, 0.6), P(3.3, 0.8).$$

This point set is shown in the left of Figure 1.

To estimate neighborhoods of points in this point set, the neighborhood grid for these points is built by moving them into a 4×4 grid, with each grid cell holding exactly one point. Furthermore, the points are arranged in a way such that the x -coordinate in the rows increases from left to right and the y -coordinates in the columns increases from bottom to top. A neighborhood grid like this is said to be in a *stable state* (see Definition 2.1). In Theorem 3.1, we will establish that such a stable state always exists. See the right image of Figure 1 for an illustration of a stable state for the point set given above.

To estimate the neighborhood of a query point from the point set (say, point A), the one-ring around the query point in the neighborhood grid is considered. As an estimate, the nearest neighbor from this one-ring is output as the nearest neighbor of the query point. In the right of Figure 1, the one-ring around query point A is shown in gray. The closest point to A in this one-ring is point C . As can be seen in the left of Figure 1, the actual closest point to A within the set is point B . However, point C is a reasonable estimate for a nearest neighbor to A , indicated by the dashed line.

Note that if the points are distinct when reducing to each dimension, the actual geometric coordinates of the points are not necessary to provide the configuration in the grid. That is because we can order the points along a specific coordinate axis and replace this coordinate entry by the point's index in this order (which is unique if and only if the coordinates along this axis are distinct), starting from 1. This operation reduces all coordinate entries to natural numbers, but does not change any relation of the points with respect to the grid representation. This observation allows us to reduce a set of N geometric points to permutations of the first $[N]$ natural numbers in each coordinate. Hence, we can study all possibly arising neighborhood grids by reducing to their combinatorics.

We are not the first to start from a data structure and investigate it via its underlying combinatorics. In fact, this approach has been applied almost throughout the entire history of data structure research, see, e.g., DeMillo, Eisenstat and Lipton (1978); Woo (1985); Wolter (1991); Malashina (2021). It is a fruitful endeavor to consider computer science from the special standpoint of combinatorics, to the point that whole research projects can solely be devoted to this². Thereby, we are motivated to apply the combinatorial reduction outlined above to the neighborhood grid data structure. We will illustrate this reduction procedure on the previous example.

²Confer to the results of European Project 678765, <https://cordis.europa.eu/project/id/678765>.

F(0.5,5.0)	E(1.4,4.4)	D(2.2,4.7)	H(4.1,4.8)
G(1.0,3.6)	A(2.0,3.1)	C(2.9,3.2)	K(4.2,3.8)
J(1.1,2.0)	L(1.7,1.9)	M(2.1,1.5)	B(2.5,3.4)
I(0.4,0.7)	O(1.9,0.6)	P(3.3,0.8)	M(3.5,1.8)

F(2,16)	E(5,13)	D(10,14)	H(15,15)
G(3,11)	A(8,8)	C(12,9)	K(16,12)
J(4,7)	L(6,6)	N(9,4)	B(11,10)
I(1,2)	O(7,1)	P(13,3)	M(14,5)

Figure 2: Replacing the actual point coordinate entries by their indices of a sorting in this coordinate keeps the property of a stable state in the neighborhood grid.

Example 1.2. Consider the point set as given in Example 1.1. Ordering the points by their first coordinate yields the order

$(I, F, G, J, E, L, O, A, N, D, B, C, P, M, H, K),$

while sorting along the second coordinate yields the order

$(O, I, P, N, M, L, J, A, C, B, G, K, E, D, H, F).$

Replacing the actual coordinates by their indices in these sortings then gives the following alternate point set

$A(8, 8), B(11, 10), C(12, 9), D(10, 4), E(5, 13), F(2, 16), G(3, 11), H(15, 15), I(1, 2), J(4, 7),$
 $K(16, 12), L(6, 6), M(14, 5), N(9, 4), O(7, 1), P(13, 3).$

As shown in Figure 2, replacing the original points in the neighborhood grid by this alternate point set keeps the neighborhood grid in a stable state.

Regarding the existence of stable states, Malheiros and Walter provide an algorithm to build these for arbitrary point sets de Gomensoro Malheiros and Walter (2015). However, their proof is given solely for two-dimensional point sets. Furthermore, it relies on the implicit assumption that there exists a point set with an unique stable state. Building on the combinatorial viewpoint on the neighborhood grid and on this previous work, in this paper, we will:

- ... show that any d -dimensional point set has a stable state, even for grids with varying side lengths. These stable states can be constructed in polynomial time (Theorem 3.1).
- ... prove a lower bound on the run time of this construction (Theorem 4.3) and prove time optimality of the construction algorithm for the case where all side lengths of the grid are equal (Corollary 4.4).
- ... disprove the implicit assumption of Malheiros and Walter by showing that all point sets placed in grids of size 4×4 or larger admit to at least two stable states (Propositions 5.5 and 5.6).
- ... bound the number of stable states that a point set admits to (Proposition 6.8).
- ... conjecture a classification for the point sets with maximally many stable states for all 2-dimensional grids (Conjecture 6.15).

Preliminary results of this paper were published in a PhD thesis Skrodzki (2019b) and presented at the EuroCG2018 conference Skrodzki, Reitebuch, Polthier and Das (2018).

Stable State	Stable State	Non-Stable State
(6,8)	(7,7)	(9,9)
(3,4)	(4,6)	(5,5)
(1,3)	(2,2)	(8,1)
	(3,4)	(7,7)
	(1,3)	(4,6)
	(2,2)	(5,5)
	(3,4)	(6,8)
	(1,3)	(7,7)
	(2,2)	(8,1)
	(3,4)	(9,9)
	(6,8)	(8,1)
	(5,5)	(4,6)

Figure 3: For $d = 2$ and $N = 9$, consider the collection of two permutations $\sigma = (\mathcal{I}, (3, 2, 4, 6, 5, 8, 7, 1, 9))$. Above are three states for P_σ on the 3×3 grid. The first two states are stable while the third is not as, e.g., the lowest row is not increasing in the first coordinate. As $\sigma_1 = \mathcal{I}$, we can easily read off the placement maps from the states, as the first coordinate defines the placement. For instance, in the leftmost state, we have: $\pi(1, 1) = 1$, $\pi(2, 1) = 2$, $\pi(2, 2) = 4$, etc.

2. Defining Point Sets and Stable States

For natural numbers d and N , let $\sigma = (\sigma_1, \dots, \sigma_d)$ be a collection of permutations of the set $[N]$ (i.e., each $\sigma_i \in \mathfrak{S}_N$). For each $i \in [N]$, we write $\sigma(i) = (\sigma_1(i), \sigma_2(i), \dots, \sigma_d(i))$. We call the collection $P_\sigma := \bigcup_{i=1}^N \sigma(i)$ a *point set*, which we think of as the relative positions of a collection of points in d -space. We will often set $\sigma_1 = \mathcal{I}$, where \mathcal{I} is the identity permutation, so that every point set is associated with a unique choice of permutations.

Let $(n_1, \dots, n_d) \in \mathbb{N}^d$ such that $\prod_{i=1}^d n_i = N$. Consider the tuple (a_1, \dots, a_d) such that $1 \leq a_i \leq n_i$ for all $i \in [1, d]$. We say that these tuples make up the *positions* on the $(n_1 \times \dots \times n_d)$ grid. For $(k_1, \dots, k_d) \in \mathbb{N}^d$ such that $k_i \leq n_i$ for all $i \in [d]$ and for some position $\mathbf{c} = (c_1, \dots, c_d)$ such that $c_i + k_i \leq n_i$, we call the set of positions

$$\{(b_1, \dots, b_d) \mid c_i \leq b_i \leq c_i + k_i, i \in [d]\}$$

a $k_1 \times \dots \times k_d$ connected subgrid anchored at \mathbf{c} .

Let π be a bijective map $\pi : [n_1] \times \dots \times [n_d] \rightarrow N$ which we call a *placement map*. Given collection $\sigma = (\sigma_1, \dots, \sigma_d)$, let $M_\pi(P_\sigma)$ be the $(n_1 \times \dots \times n_d)$ grid where the position $\mathbf{a} = (a_1, \dots, a_d)$ is labeled with the *point*

$$\sigma(\pi(\mathbf{a})) := (\sigma_1(\pi(a_1, \dots, a_d)), \sigma_2(\pi(a_1, \dots, a_d)), \dots, \sigma_d(\pi(a_1, \dots, a_d))).$$

We call $M_\pi(P_\sigma)$ a *state*. See Figure 3 for three examples of states for a given point set.

We are interested in finding a particular class of states called *stable states*. Let $\mathbf{a} = (a_1, \dots, a_d)$ and $\mathbf{b} = (b_1, \dots, b_d)$ be two distinct grid positions in $M_\pi(P_\sigma)$. They address both the position in the grid underlying $M_\pi(P_\sigma)$ as well as the labeling induced at said positions by π and σ . We say that \mathbf{a} is *compatible* with \mathbf{b} with respect to π if any of the following hold:

- For some $1 \leq i < j \leq d$, we have $a_i \neq b_i$ and $a_j \neq b_j$.
- For some $1 \leq i \leq d$, we have $a_i < b_i$ and $\sigma_i(\pi(a_1, \dots, a_d)) < \sigma_i(\pi(b_1, \dots, b_d))$.
- For some $1 \leq i \leq d$, we have $a_i > b_i$ and $\sigma_i(\pi(a_1, \dots, a_d)) > \sigma_i(\pi(b_1, \dots, b_d))$.

Definition 2.1. A state $M_\pi(P_\sigma)$ is a *stable state* if every pair of grid positions are compatible.

For each $i \in [d]$, an *i -strip* is a maximal collection of positions in $[n_1] \times \dots \times [n_d]$ which differ only in their i^{th} entry. In particular, for any fixed

$$(a_1, \dots, a_{i-1}, a_{i+1}, \dots, a_d) \in [n_1] \times \dots \times [n_{i-1}] \times [n_{i+1}] \times \dots \times [n_d],$$

we get an i -strip of the form $\{(a_1, \dots, a_{i-1}, k, a_{i+1}, \dots, a_d) \mid 1 \leq k \leq n_i\}$. Notice that $M_\pi(P_\sigma)$ is stable if and only if for every $1 \leq i \leq d$, the labeling along each i -strip is increasing in its i^{th} coordinate. We will also call 1-strips *rows* and 2-strips *columns*. Conversely, for each $i \in [d]$, an *i -slice* is a maximal collection of positions in $[n_1] \times \dots \times [n_d]$ which can differ everywhere, but in their i^{th} entry.

Example 2.2. If $d = 2$, then $M_\pi(P_\sigma)$ is a matrix whose entries are ordered pairs of integers. In this case, $M_\pi(P_\sigma)$ is a stable state if the rows are ordered by first coordinate and the columns are ordered by second coordinate. See Figure 3 for an example of stable states and a non-stable state.

Lemma 2.3. If σ and σ' are collections of permutations such that $P_\sigma = P_{\sigma'}$, then the stable states of σ and σ' coincide.

PROOF. Let $\sigma = (\sigma_1, \dots, \sigma_d)$ and $\sigma' = (\sigma'_1, \dots, \sigma'_d)$. Then, the condition that $P_\sigma = P_{\sigma'}$ is equivalent to the condition that there is a $\sigma^* \in \mathfrak{S}_N$ such that for every $i \in [d]$, we have $\sigma'_i = \sigma_i \circ \sigma^*$. It is immediate that $M_\pi(P_{\sigma'})$ corresponds to $M_{\sigma^* \circ \pi}(P_\sigma)$. The lemma follows.

Definition 2.4. $S(P_\sigma)$ is the set of all stable states for a point set P_σ .

Notice that this set is well defined by Lemma 2.3.

3. A Polynomial-Time Building Algorithm

Starting from the definitions as presented in the previous section, we will now investigate the following question: Given any point set P_σ as specified above, does it always hold that $|S(P_\sigma)| > 0$, i.e., is there always at least one stable state $M_\pi(P_\sigma)$? Malheiros and Walter have answered this questions for the case of $d = 2$ and $n_1 = n_2$, see (de Gomensoro Malheiros and Walter, 2015, Sec. III, pp. 181–182). They did so by introducing a sorting algorithm that for any P_σ produces a placement π such that $M_\pi(P_\sigma)$ is a stable state. In the following theorem, we will generalize this statement and their algorithm to the case of arbitrary grid lengths n_i and arbitrary dimension d .

Theorem 3.1 (generalizing de Gomensoro Malheiros and Walter (2015)). For every natural number $N = \prod_{i=1}^d n_i$, $(n_1, \dots, n_d) \in \mathbb{N}^d$, $d \in \mathbb{N}$, and for every collection of permutations $\sigma = (\sigma_1, \dots, \sigma_d)$, $\sigma_i \in \mathfrak{S}_N$, there is a placement map π such that $M_\pi(P_\sigma)$ is a stable state. Furthermore, this map π can be constructed in $\mathcal{O}\left(N \sum_{j=0}^{d-1} \log\left(\prod_{i=1}^{d-j} n_i\right)\right)$.

PROOF. Start with the points p_1, \dots, p_N of P_σ , $N = \prod_{i=1}^d n_i$, $d \geq 1$ interpreted as a first sequence. Use it as input for Algorithm 1. The algorithms recursively sorts the input along its highest dimension and splits it into equally large subsequences, stripping them of their highest dimension entry, to be processed in the next recursion step. From the returned, ordered subsequences, a grid M is assembled.

Algorithm 1 Recursive computation of a stable placement π

```

1: procedure RECURSIVESPLIT(Sequence  $P'$ , lengths  $(n_1, \dots, n_\ell)$ )
2:   Sort  $P'$  by the entries in the  $\ell$ th dimension
3:   Denote this ordering by  $q_1, \dots, q_{N'}$ 
4:   if  $\ell = 1$  then
5:     return the sequence  $(q_1, \dots, q_{N'})$ 
6:   else
7:      $M \leftarrow$  an empty  $n_1 \times \dots \times n_\ell$  grid
8:      $N'' \leftarrow \prod_{i=1}^{\ell-1} n_i$ 
9:     for  $i = 1, \dots, n_\ell$  do
10:        $Q_i \leftarrow \{q_{(i-1)N''+1}, \dots, q_{iN''}\}$   $\triangleright$  Split the sorted sequence into  $n_\ell$  subsequences  $Q_i$  of length  $N''$  each
11:        $M(\dots, i) \leftarrow$  RECURSIVESPLIT( $Q_i, (n_1, \dots, n_{\ell-1})$ )  $\triangleright$  Fill each of the  $n_\ell$   $i$ -slices of the grid  $M$  with a
           recursive solution
12:     end for
13:     return  $M$ 
14:   end if
15: end procedure

```

The resulting grid M is then a stable state $M_\pi(P_\sigma)$ by induction: If $d = 1$, the grid is just a sorted sequence of numbers, which is trivially stable. For $d > 1$, each of the $(d - 1)$ -dimensional slices of the resulting grid M is stable

by the induction hypothesis. Furthermore, it is stable along all d -strips as the glued sub-sequences have been ordered along this dimension in the first step of the procedure. Thus, the grid represents indeed a stable state.

The runtime of the algorithm consists of the respective sorting steps, multiplied by the recursion level. It can be computed as

$$\begin{aligned} N \log(N) + (n_d) \binom{d-1}{\prod_{i=1}^{d-1} n_i} \log \binom{d-1}{\prod_{i=1}^{d-1} n_i} + \binom{d}{\prod_{j=d-1}^d n_j} \binom{d-2}{\prod_{i=1}^{d-2} n_i} \log \binom{d-2}{\prod_{i=1}^{d-2} n_i} + \dots + \binom{d}{\prod_{j=2}^d n_j} (n_1) \log(n_1) \\ = N \sum_{j=0}^{d-1} \log \binom{d-j}{\prod_{i=1}^{d-j} n_i} \end{aligned}$$

Note that the implicit decision in the first step of the algorithm, to sort along the last coordinate, is arbitrary. The theorem holds for sorting along any dimension, when corresponding splitting and filling is applied. We will explore this observation in Section 5.

Note further that in the specific case of $n_i = n \in \mathbb{N}$ for all i , we have

$$\begin{aligned} n^0(n^d \log(n^d)) + n^1(n^{d-1} \log(n^{d-1})) + \dots + n^{d-1}(n^1 \log(n)) \\ = d(n^d \log(n)) + (d-1)(n^d \log(n)) + \dots + (d-(d-1))(n^d \log(n)) \\ = (n^d \log(n)) \sum_{i=1}^d i \\ = (n^d \log(n)) \frac{d(d+1)}{2} \end{aligned}$$

When treating d as a constant, the stable state for this arrangement can therefore be found in time $\mathcal{O}(n^d \log(n))$.

Example 3.2. For the case $d = 2$ and $n = 3$, an illustration of the procedure presented by this theorem is given in Figure 4.

Theorem 3.1 imposes an upper bound on the runtime of any time-optimal, comparison-based algorithm that creates a stable state $M_\pi(P_\sigma)$ for a point set P_σ . A next question to tackle is: What is a lower bound? A trivial lower bound is given by the number N of points in the point set. However, to sharpen this lower bound, we need further results on the cardinality of the set $S(P_\sigma)$, i.e., on the number of stable states for a given point set P_σ .

4. Counting Stable States and a Lower Bound

In order to better understand the search space for any sorting algorithm that is to build a stable state $M_\pi(P_\sigma)$, we start by counting the number of possible stable states through straightforward combinatorial means.

Proposition 4.1. Given some $n_i \in \mathbb{N}$ with $N = \prod_{i=1}^d n_i$, there are:

1. $(N!)^{d-1}$ point sets P_σ and $(N!)^d$ different states $M_\pi(P_\sigma)$,
2. $(N!)^d / \prod_{i=1}^d (n_i!)^{N/n_i}$ ways to fill the grid with a stable state,
3. $N! / (n_1!)^{N/n_1}$ placements that are stable with respect to the first coordinate and each such placement is stable for $(N!)^{d-1} / \prod_{i=2}^d (n_i!)^{N/n_i}$ point sets P_σ .

Furthermore,

- (4) $1 / \prod_{i=1}^d (n_i!)^{N/n_i}$ of all fillings of the grid are stable.
- (5) The expected number of stable states for a point set chosen uniformly at random is $N! / \prod_{i=1}^d (n_i!)^{N/n_i}$.

PROOF. For the first statement, recall that each point set P_σ is created by d permutations $\sigma_i \in \mathfrak{S}$, where σ_1 is the identity permutation \mathcal{I} . Thus, are $N!$ choices for $2 \leq i \leq d$, yielding a total of $(N!)^{d-1}$ possible point sets. Furthermore, there are $N!$ ways to pick the map π which places the point set in the grid. Therefore, we obtain $(N!)^d$ possible states $M_\pi(P_\sigma)$.

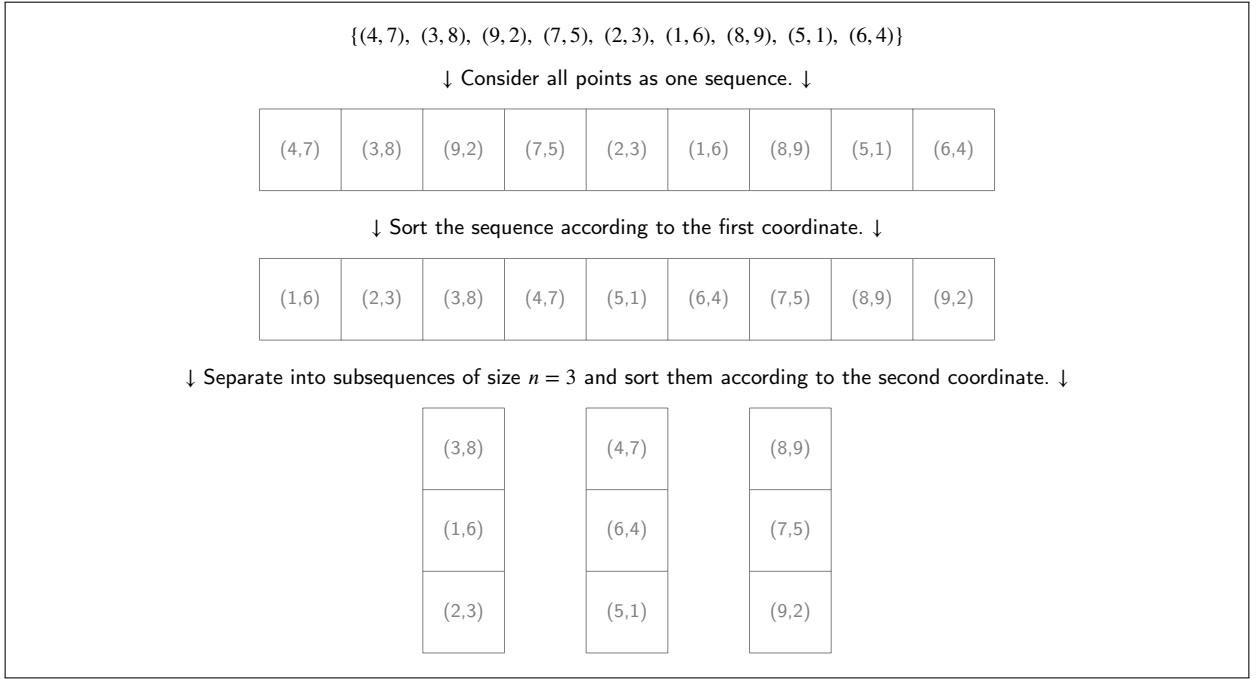


Figure 4: An illustration of Algorithm 1 used in Theorem 3.1. The last step gives the columns of the final matrix which is then in a stable state.

For the second statement, recall that when a dimension $i \in [d]$ is fixed, the i^{th} coordinate has to be ascending along all i -strips. For a fixed $i \in [d]$, the number of possible partitions, i.e., the number of ways to fill the i -strips with coordinates, is given by the multinomial coefficient

$$\binom{N}{n_i, \dots, n_i} = \binom{N}{n_i} \binom{N - n_i}{n_i, \dots, n_i} = \dots = N! / (n_i!)^{N/n_i},$$

because once the coordinates for a specific i -strip are chosen, they have to be ordered increasingly, for which there is only one way. However, the choices for how to fill the i -strips have to be made for each of the $i \in d$ dimensions. Therefore, the total number of stable states is given by

$$\prod_{i=1}^d \frac{N!}{(n_i!)^{N/n_i}} = (N!)^d / \prod_{i=1}^d (n_i!)^{N/n_i}.$$

The third statement follows from an analogous argument to the second statement.

For the fourth statement, we divide the number of stable states (given by statement 2) by the total number of states (given by statement 1).

Finally, the fifth statement follows from dividing the number of point sets by the number of stable states.

For the special case of $n_i = n_j$ for all $i \neq j$, we obtain the following corollary:

Corollary 4.2. *Given some $N, n \in \mathbb{N}$ with $N = n^d$, there are:*

1. $(N!)^{d-1}$ point sets P_σ and therefore $(N!)^d$ different states $M_\pi(P_\sigma)$,
2. $\left(N! / (n!)^{n^{d-1}} \right)^d$ ways to fill the grid with a stable state,
3. $\frac{N!}{(n!)^{d-1}}$ placements π that are stable with respect to the first coordinate and each such placement is stable for $\left(N! / (n!)^{n^{d-1}} \right)^{d-1}$ point sets P_σ .

Furthermore,

4. $1/(n!)^{n^{d(d-1)}}$ of all fillings of the grid are stable
5. and each point set P_σ has $N!/(n!)^{dn^{d-1}}$ stable states on average.

Having the necessary results at hand, we can now prove a lower bound on the building time of the neighborhood grid.

Theorem 4.3. *Any comparison-based algorithm has to perform at least $\omega\left(\log\left(\prod_{i=2}^d (n_i!)^{N/n_i}\right)\right)$ operations.*

PROOF. Consider any comparison-based algorithm \mathcal{A} that constructs a placement π such that $M_\pi(P_\sigma)$ is a stable state for some given point set P_σ with respect to the first coordinate. By Proposition 4.1, there are $\frac{N!}{(n_1!)^{N/n_1}}$ many of these, where the algorithm has to identify the correct one. Each subsequent query of \mathcal{A} can be considered as a node of a decision-tree, where the leaves correspond to placements of which some are stable for P_σ . For an optimal algorithm, this tree is balanced and has depth $\log(N!)$. Recall the result of Proposition 4.1 that any such placement π is stable for $\prod_{i=2}^d \frac{N!}{(n_i!)^{N/n_i}}$ point sets. Thus, when building the tree, the algorithm \mathcal{A} cannot stop at a subtree with more than

$$\frac{N!}{(n_1!)^{N/n_1}} \prod_{i=2}^d \frac{N!}{(n_i!)^{N/n_i}} = \prod_{i=1}^d \frac{N!}{(n_i!)^{N/n_i}}$$

leaves, as either the placement will not be stable with respect to the first coordinate or one of the leaves will surely not be stable under the currently considered placement. Therefore, the algorithm has to perform at least

$$\log(N!) - \log\left(\prod_{i=1}^d \frac{N!}{(n_i!)^{N/n_i}}\right) = \log(N!) - \left(\log(N!) + \log\left(\prod_{i=1}^d \frac{1}{(n_i!)^{N/n_i}}\right)\right) = \log\left(\prod_{i=1}^d (n_i!)^{N/n_i}\right)$$

many comparisons before it can possibly terminate.

Overall, this is reminiscent of proving a lower bound on the one-dimensional sorting problem. The main difference to sorting is that a placement π can yield a stable state for several point sets, where in one-dimensional sorting, the solution is unique for each point set and thus each placement also only admits to exactly one point set. From the proof of Theorem 4.3, for non-uniform configurations with $n_i \neq n_j$ for some $i \neq j$, it remains unclear whether a faster algorithm than that from Theorem 3.1 can be found. However, Theorem 4.3 yields the following corollary for d -dimensional point sets of uniform shape $n_i = n$ for all $i \in [d]$ and some $n \in \mathbb{N}$.

Corollary 4.4. *In the case of $n_i = n \in \mathbb{N}$ for all $i \in [d]$, Algorithm 1 is a time-optimal building algorithm for the neighborhood grid among all comparison-based algorithms and takes $\Theta(n^d \log(n))$ time to build a grid for a point set P_σ of $N = n^d$ points.*

PROOF. In the case of $n_i = n$, the tree has to be traversed to depth at least

$$\log\left(\prod_{i=1}^d (n!)^{n^{d-1}}\right) = \log\left(\left((n!)^{n^{d-1}}\right)^d\right) = d \log\left((n!)^{n^{d-1}}\right) = dn^{d-1} \log(n!) = \mathcal{O}(n^d \log(n)).$$

Therefore, each comparison-based algorithm building a stable state needs to perform at least $\Omega(n^d \log(n))$ operations. The observation that the property of a point set being restricted does not interfere with the run time completes the proof.

Finally, we state the result in the form that it is historically most used in, cf. de Gomensoro Malheiros and Walter (2015), namely the case of uniform grid side lengths in dimension $d = 2$.

Corollary 4.5. *In the case of $d = 2$, Algorithm 1 is a time-optimal building algorithm for the neighborhood grid among all comparison-based algorithm and takes $\Theta(n^2 \log(n))$ time to build a grid for a point set P_σ of $N = n^2$ points.*

5. Point Sets with Unique Stable States

In the previous sections, we have constructively proven the existence of stable states for a given point set P_σ on an $n_1 \times \dots \times n_d$ grid. However, in many cases, this stable state is not unique (as shown in Figure 3). In this section, we will explore properties of point sets with unique stable states and partially classify the grids which allow for such point sets. We begin with two necessary conditions for a point set to have a unique stable state.

Lemma 5.1 (First necessary condition of a unique stable state). *Consider a stable state $M_\pi(P_\sigma)$ on a d -dimensional point set P_σ . Suppose that $M_\pi(P_\sigma)$ is the unique stable state of P_σ and consider any $i \in [d]$. For any grid positions $\mathbf{a} = (a_1, \dots, a_d)$ and $\mathbf{b} = (b_1, \dots, b_d)$ such that $a_i < b_i$, we must have $\sigma_i(\pi(\mathbf{a})) < \sigma_i(\pi(\mathbf{b}))$.*

PROOF. First, suppose that the condition is not satisfied for $i = 1$. Then, if we apply Theorem 3.1 to P_σ , we obtain a stable state which does satisfy the condition, because the algorithm begins by sorting by the first coordinate. This contradicts the assumption that $M_\pi(P_\sigma)$ is unique.

For general i , we can consider an alternate version of the algorithm from Theorem 3.1 that switches the role of the first and i -th coordinate. Because the order of the coordinates is arbitrary, this will also produce a stable state, and our proof works equivalently to the $i = 1$ case.

This provides a first necessary condition on the uniqueness of a stable state $M_\pi(P_\sigma)$. We can equivalently reformulate this condition on σ in terms of a condition on π as follows.

Corollary 5.2. *Given a point set P_σ with $N = \prod_{i=1}^d n_i$ points and a stable state $M_\pi(P_\sigma)$. In order for $M_\pi(P_\sigma)$ to be the unique stable state of P_σ , it has to satisfy the following condition: For any given coordinate $i \in [d]$, the $\prod_{j \in [d] \setminus i} n_j$ smallest i -coordinates have to be placed in the first positions of all i -strips, the next $\prod_{j \in [d] \setminus i} n_j$ smallest i -coordinates have to be placed in the second positions of all i -strips, etc., until the $\prod_{j \in [d] \setminus i} n_j$ largest i -coordinates are placed in the n_i -th position of all i -strips.*

This first necessary condition provides candidates for point sets P_σ that could have a unique stable placement. We can count the number of point sets P_σ that satisfy the condition of Lemma 5.1 by considering the reformulation according to Corollary 5.2. For each of the $i \in [d]$ coordinates, consider each of their n_i many i -slices. Each of these slices has $\frac{N}{n_i}$ many points of which the i -th coordinates can be permuted without violating the condition. Therefore, there are

$$\prod_{i \in [d]} \left(\left(\frac{N}{n_i} \right)! \right)^{n_i}$$

many point sets that satisfy the condition of Lemma 5.1. Throughout this section, we will have the case of $d = 2$, $n_1 = n_2 = 4$ as a running example. In that case, there are $16! \approx 2.1 \cdot 10^{13}$ point sets of which $((4!)^4)^2 \approx 1.1 \cdot 10^{11}$ satisfy the condition of Lemma 5.1.

To further simplify the challenge of checking which point sets have a unique stable state, we introduce a second necessary condition. This condition works recursively and allows results on smaller grids to be applied to larger grids.

Lemma 5.3 (Second necessary condition of a unique stable state). *Given a point set P_σ with $N = \prod_{i=1}^d n_i$ points and a unique stable state $M_\pi(P_\sigma)$. Then, any $k_1 \times \dots \times k_d$ connected subgrid of $M_\pi(P_\sigma)$ with $k_i \in [d]$ is in a unique stable state.*

PROOF. Given some unique stable state $M_\pi(P_\sigma)$. Assume there exists some $k_1 \times \dots \times k_d$ connected subgrid \widetilde{M} of M , anchored at $\mathbf{c} = (c_1, \dots, c_d)$, such that \widetilde{M} is not unique, but has a different stable state \widetilde{M} .

By Corollary 5.2, we know that the i -th values of the first $c_i - 1$ elements in each i -strip are smaller than the i -th value of the c_i -th element. Furthermore, the i -th values of the last $n_i - (c_i + k_i)$ elements in each i -strip are larger than the i -th value of the $c_i + k_i$ element. This fact remains true independent of any stable re-ordering of the $k_1 \times \dots \times k_d$ connected subgrid of M , anchored at \mathbf{c} . In particular, this remains true for the reordering induced by \widetilde{M} . Therefore, replacing \widetilde{M} by \widetilde{M} in M gives another stable state, which violates the uniqueness of M .

(2,3)	(4,4)
(1,1)	(3,2)

(3,2)	(4,4)
(1,1)	(2,3)

(1,9)	(4,8)	(7,7)
(2,4)	(5,5)	(8,6)
(3,3)	(6,2)	(9,1)

(1,9)	(7,7)	(8,6)
(4,8)	(5,5)	(6,2)
(2,4)	(3,3)	(9,1)

Figure 5: The left 2×2 and 3×3 stable states respectively satisfy the conditions of Lemmas 5.1 and 5.3. While the condition of Lemma 5.3 is trivially satisfied for the 2×2 example, it has to be checked for the 3×3 example. The right 2×2 and 3×3 stable states show that the conditions of Lemmas 5.1 and 5.3 are only necessary conditions, but do not imply uniqueness of the stable states.

For this statement, it is not as easy as it was for Lemma 5.1 to give a count of those point sets that satisfy the condition. This is mostly because of the recursive nature of the condition: we would not only have to take into account all connected subgrids of some $M_\pi(P_\sigma)$, but also ensure that they interlock correctly. However, we can turn to the case of $d = 2$, $n_1 = n_2 = 4$ again to provide a feeling of how Lemma 5.3 reduces the number of point sets that can have a unique stable state. Recall that there are approximately $2.1 \cdot 10^{13}$ point sets for this case, out of which approximately $1.1 \cdot 10^{11}$ satisfy Lemma 5.1. We can put an upper bound on the number of point sets satisfying Lemma 5.3 by considering all nine connected 2×2 subgrids, for each of which there are 12 unique stable states. Hence, there are at most $12^9 \approx 5.2 \cdot 10^9$ point sets that have a unique stable state. While this brings the previous estimate down by two orders of magnitude, the actual number is far lower, as we will establish within the proof of Proposition 5.5.

Note that the converses of Lemmas 5.1 and 5.3 do not hold: a stable state can satisfy both properties without being unique. For example, consider the point sets and their placements as shown in Figure 5. The respective left stable states satisfy the conditions of Lemmas 5.1 and 5.3, but the right state shows that these conditions do not imply uniqueness.

Definition 5.4. Let $(n_1, \dots, n_d) \in \mathbb{N}^d$, then we write $\mathcal{U}(n_1, \dots, n_d)$ for the number of point sets which have a unique stable state on the $(n_1 \times \dots \times n_d)$ grid.

To give some examples, it is easy to check that of the 24 different point sets on four points, it holds that $\mathcal{U}(2, 2) = 12$. The other twelve point sets allow for two stable states each. When placing nine points in a (3×3) grid, checking the candidates that satisfy Lemmas 5.1 and 5.3 reveals that $\mathcal{U}(3, 3) = 966$, out of 362,880 point sets. Considering these numbers, one natural question to ask is whether there exist grids for which $\mathcal{U}(n_1, \dots, n_d) = 0$. In fact, we prove that there are infinitely many such grids in the next two propositions.

Proposition 5.5. $\mathcal{U}(4, 4) = 0$

PROOF. Making use of the reductions provided by Lemmas 5.1 and 5.3, we proved the statement using a brute force computational approach. For this, we assume that we have a complete list of unique stable states that fit all possible subgrids of the 4×4 grid. These are 12, 106, and 966 for the 2×2 , 2×3 , and 3×3 case, respectively. As these instances are small, they are easily computed and the time to do so is negligible within the overall time used to run the program. This brings the upper bound down to $966 \cdot 106^2 \cdot 12 \approx 1.3 \cdot 10^8$.

Given this data for the subgrids, the program iteratively constructs all point sets that satisfy the condition of Lemma 5.1. For each point set, first, the condition of Lemma 5.3 is checked and those that do not satisfy it are discarded. This leaves a mere 37,536 candidate point sets for the 4×4 case. In order to optimally utilize Lemma 5.3, the construction of the point sets is done in a lazy way: Only part of the point set is constructed and if this already violates Lemma 5.3, all point sets containing this part can be neglected. This shows the importance of Lemma 5.3, as iterating over all approximately 10^8 point sets satisfying the condition of Lemma 5.1 would cause a significantly longer computation time. Thus, by combining Lemmas 5.1 and 5.3, we reduce the order of magnitude of point sets to check from the trivial bound of 10^{11} to 10^4 for our example.

Finally, for the remaining candidate point sets, all $16!$ different placements of the point set in the (4×4) grid are checked. Again, this is only done until two stable placements are found, which contradicts uniqueness for this point set and the iteration over all other placements can be halted.

After running this program, we have found two stable states for all point sets that satisfy the conditions of Lemma 5.3. Hence, we can conclude that there is not a single point set with a unique placement in the (4×4) grid.

Hence, the (4×4) grid does not allow for unique stable states. We can extend this to other grids by the following proposition.

Proposition 5.6. *For two natural numbers $d \geq d'$, let $(n_1, \dots, n_d) \in \mathbb{N}^d$, $(n'_1, \dots, n'_{d'}) \in \mathbb{N}^{d'}$ such that for all $i \in [d']$, we have $n_i \geq n'_i$. If $\mathcal{U}(n'_1, \dots, n'_{d'}) = 0$, then $\mathcal{U}(n_1, \dots, n_d) = 0$.*

PROOF. Assume there is a stable state on an $(n_1 \times \dots \times n_d)$ grid that satisfies the condition of Lemma 5.1. As $n_i \geq n'_i$ for all $i \in [d']$, there exists an $n'_1 \times \dots \times n'_{d'} \times 1 \times \dots \times 1$ connected subgrid (with $d - d'$ trailing 1s) which has no unique stable state. This violates the condition of Lemma 5.3, which implies that the $(n_1 \times \dots \times n_d)$ grid also does not have any stable states.

By Proposition 5.6, a grid that has any subgrid without unique stable states does not have unique stable states itself. Combining this with Proposition 5.5, we arrive at the following corollary.

Corollary 5.7. *Let $(n_1, \dots, n_d) \in \mathbb{N}^d$ that for some $i \neq j$, we have $n_i, n_j \geq 4$. Then, we must have $\mathcal{U}(n_1, \dots, n_d) = 0$.*

This proves that there are infinitely many grids that do not admit a single unique stable state. Going back to the paper of de Gomensoro Malheiros and Walter (2015), as stated above, they provided a proof for an upper bound on the time needed to build a stable state, see the discussion before Theorem 4.3. In their paper, they continue to say that “the problem of sorting s unrelated lists of s real values has $\mathcal{O}(n \log n)$ as its established lower bound” (de Gomensoro Malheiros and Walter, 2015, p. 182), where they assume $d = 2$; their s would be n_1, n_2 and their n is N in our notation. This serves as their argument to establish a lower bound that coincides with the upper one.

Note that this argument assumes that in the two-dimensional case, there is a unique separation of the points into the rows (or columns) of the grid. Given this assumption, the rows (or columns) have a respective unique sorting. However, our Corollary 5.7 shows that each point set has at least two stable states. Each of these stable states provides a different partition of the points into the rows (or columns). Thereby, the assumption of unique sets of rows (or columns) is incorrect. Thus, the argument for the existence of a lower bound as provided by de Gomensoro Malheiros and Walter (2015) does not hold.

Note further that this is not a problem in the order of the points from the input point set. One might fix an arbitrary order of the points, for instance by going through their coordinates in some order and imposing a corresponding lexicographical sorting of the points. As soon as it is determined which points of the input point set go into which row of the grid, this order would indeed imply uniqueness within each row, respectively. However, Corollary 5.7 states that the partition of the input elements into the rows of the grid is not unique. In fact, we will see in Section 6 that the number of partitions into the different rows can grow exponentially large with respect to the size of the input point set.

Nevertheless, the proof as given by us for Corollary 4.4 does not depend on any assumption of uniqueness. Thus, this proof of time-optimality – for the case of $n_i = n$ – repairs the statement of Malheiros and Walter and even extends it to arbitrary grid shapes.

To continue, note that Corollary 5.7 proves that there are infinitely many point sets without a unique stable placement. Now, we turn to the converse and ask: How many grids do allow for unique stable states? In fact, we also give an infinite family of grids such that $\mathcal{U}(n_1, \dots, n_d) > 0$.

Proposition 5.8. *If $n_i = 2$ for all $i < d$ (with $n_d \in \mathbb{N}$ arbitrary), then $\mathcal{U}(n_1, \dots, n_d) > 0$.*

PROOF. We can explicitly construct a point set with a unique stable state. Recall that

$$N = \prod_{i=1}^d n_i = 2^{d-1} n_d.$$

Let $\sigma_1 = \mathcal{I}$ and for $i > 1$, define:

$$\sigma_i(j) = \begin{cases} \min k \geq 1 \text{ such that } k \notin \{\sigma_i(1), \dots, \sigma_i(j-1)\} & \text{if } \lfloor j/2^{i-1} \rfloor \text{ is even,} \\ \max k \leq N \text{ such that } k \notin \{\sigma_i(1), \dots, \sigma_i(j-1)\} & \text{if } \lfloor j/2^{i-1} \rfloor \text{ is odd.} \end{cases}$$

(2,10)	(4,9)	(6,8)	(8,7)	(10,6)
(1,1)	(3,2)	(5,3)	(7,4)	(9,5)

Figure 6: For $d = 2$, $n_1 = 2$, and $n_2 = 5$, Proposition 5.8 says that there exists a point set with a unique stable state. Above is the the unique stable state for the point set $P_{(\sigma_1, \sigma_2)}$ as constructed in the proof of the proposition.

We then place the points in the grid sequentially from p_1 up to p_N . Notice that for each $k \in [N]$, once we have placed the points p_1, \dots, p_k into the grid, there is only one place where p_{k+1} can go without forcing the state to be non-stable.

This can be seen by inductively considering the d -slices of the grid to be filled. Arriving at an empty d -slice, there are 2^{d-1} points to be placed. The next 2^{d-1} points given by σ have the property that each of their coordinates is either minimal or maximal with respect to the currently available numbers. Furthermore, among these next 2^{d-1} points, each of the 2^{d-1} many choices of minimum and maximum for each coordinate occurs exactly once. This places the points uniquely within the currently considered d -slice.

Furthermore, because the minimum or maximum is chosen, the first 2^{d-1} points of σ have to go into the first d -slice, the next 2^{d-1} points have to go into the second slice, and so on, as otherwise there would be a pair of points along an i -strip, $1 \leq i \leq d - 1$ that is not stably placed. Hence, there is only one stable placement for this point set and $\mathcal{U}(2, \dots, 2, n_d) \geq 1$.

Example 5.9. For $1 \leq i \leq 5$, the construction of Proposition 5.8 gives the following permutations:

$$\begin{aligned} \sigma_1 &= (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, \dots, N), \\ \sigma_2 &= (1, N, 2, N - 1, 3, N - 2, 4, N - 3, 5, N - 4, \dots), \\ \sigma_3 &= (1, 2, N, N - 1, 3, 4, N - 2, N - 3, 5, 6, \dots), \\ \sigma_4 &= (1, 2, 3, 4, N, N - 1, N - 2, N - 3, 5, 6, \dots), \\ \sigma_5 &= (1, 2, 3, 4, 5, 6, 7, 8, N, N - 1, \dots). \end{aligned}$$

See Figure 6 for the case where $d = 2$, $n_1 = 2$, and $n_2 = 5$. Note that all points with odd first coordinates have to go into the bottom row because of their small second coordinate. Similarly, all points with even first coordinates have to go into the upper row because of their large second coordinate. Thus, there is a unique way to place each point while maintaining stability.

For the remainder of this section, we will restrict to the case of $d = 2$. From Proposition 5.5 and Corollary 5.7, we can deduce that $\mathcal{U}(n_1, n_2) = 0$ for all $n_1, n_2 \geq 4$. Thus, it suffices to restrict to the case where $n_1 \in [3]$.

When $n_1 = 1$, the number of unique stable states is just the number of point sets, as these are a simple sequence and are therefore uniquely ordered. Hence, we have $\mathcal{U}(1, n_2) = n_2!$. When $n_1 = 2$, Proposition 5.8 implies that $\mathcal{U}(2, n_2) > 0$. When $n_1 = 3$, we conjecture that $\mathcal{U}(3, n_2) > 0$ (see Conjecture 5.11). However, general formulas for $\mathcal{U}(2, n_2)$ and $\mathcal{U}(3, n_2)$ remain to be derived (see Open 5.10). Finally, using a brute force computation, we calculated the number of unique stable states for $n_1 = 2$ when $n_2 \in [8]$ and $n_1 = 3$ when $n_2 \in [7]$. The results are given in Table 1.

Despite calculating $\mathcal{U}(2, n_2)$ and $\mathcal{U}(3, n_2)$ for small values of n_2 , the sequences $\mathcal{U}(2, n_2)$ and $\mathcal{U}(3, n_2)$ remain mysterious, and do not match any existing OEIS entries.

Open 5.10. What are the sequences $\mathcal{U}(2, n_2)$ and $\mathcal{U}(3, n_2)$? Can these be expressed in a closed form or at least asymptotically?

While we do not know much about the sequence $\mathcal{U}(3, n_2)$, we are fairly confident that it is always positive. In particular, we conjecture the following.

Conjecture 5.11. For any $n_2 \in \mathbb{Z}_{\geq 1}$, we have $\mathcal{U}(3, n_2) > 0$.

$n_1 \backslash n_2$	1	2	3	4	5	6	7	8	...	n_2
1	1	2	6	24	120	720	5,040	40,320	...	$n_2!$
2		12	106	1,108	12,826	163,276	2,274,592	34,318,068	...	> 0
3			966	1,484	3,528	8,176	18,592	???	...	???
4				0	0	0	0	0	...	0
≥ 5					0	0	0	0	...	0

Table 1

A partially filled chart giving $\mathcal{U}(n_1, n_2)$. for small n_1 and n_2 . If Conjecture 5.11 holds, then the question marks in row 3 must be strictly positive.

One way to prove Conjecture 5.11 is to construct an explicit point set with a unique stable state for each n_2 . We have a candidate construction. Let $\sigma_1 = \mathcal{I}$, and define σ_2 in the following way:

$$\sigma_2(j) = \begin{cases} 3j - 2 & \text{for } 1 \leq j \leq n_2 \\ 3(2n_2 + 1 - j) & \text{for } n_2 < j \leq 2n_2 \\ 3(j - 2n_2) - 1 & \text{for } 2n_2 < j \leq 3n_2 \end{cases} .$$

The construction above gives a unique stable state for $n_2 \leq 7$, and we conjecture that this generalizes to all n_2 (which would prove Conjecture 5.11).

Note that Table 1 gives an almost complete picture of unique stable states for two-dimensional grids. While Lemma 5.1 and 5.3 as well as Proposition 5.6 hold in arbitrary dimension d , it is a priori unclear how many unique stable states there are for grids of higher dimension $d \geq 3$ with $n_i \leq 3$ for all but one index i . Using the aforementioned computer searches, we found 1,823,944 point sets that admit a unique stable state for three-dimensional grids of size $2 \times 2 \times 2$. However, the computational search quickly becomes extremely expensive. It is therefore left as future work to expand this investigation to the higher-dimensional setting.

As a final remark in this section, note that in the proof of Proposition 5.5, we found that each point set of 16 points allows for at least two stable placements. However, this lower bound on the minimum number of stable placements is not necessarily sharp. Using a computer search, we checked about 7% of the point sets on the 4×4 grid. The minimum number of stable states that one point set had, was seven. This minimum number came up several times, for different point sets. However, as we did not pick the point sets uniformly random, we cannot extrapolate how many point sets have exactly seven stable states. Also, there might be other point sets with even lower numbers of stable states that we have not found yet. Hence, we ask the following, open question.

Open 5.12. *For any specific grid with no point sets that have a unique stable state (e.g., the 4×4 grid), what is the minimum number of stable states a point set has? And is it possible to classify the point sets for which this minimum is achieved?*

Having phrased this open question, we turn to its opposite: What are the point sets with the highest number of stable states? We will investigate this question in the following section.

6. Point Sets with Maximally Many Stable States

Fix d and (n_1, \dots, n_d) with $N = \prod n_i$. Recall that \mathcal{I} denotes the identity permutation. We will write $P_{\mathcal{I}}$ as shorthand for $P_{(\mathcal{I}, \dots, \mathcal{I})}$ (the point set associated with d copies of the same permutation). We can think of a state of $P_{\mathcal{I}}$ as an assignment of a single number to each position on the grid by identifying each point (a, \dots, a) , $a \in [N]$ with the number a . The state is stable if and only if these numbers are ordered in every Cartesian direction when traversing the grid.

Because of the reduction to numbers, $P_{\mathcal{I}}$ is one-dimensional. Hence, one might suspect that it would have a large number of stable states. In fact, we will show in Corollary 6.3 that when $d = 2$, the number of stable states of $P_{\mathcal{I}}$ is maximal among all point sets. We conjecture that this is true for all d (see Conjecture 6.4).

Algorithm 2 Stable State to Identity Stable State for $d = 2$

- 1: **procedure** $\varphi_\sigma : S(P_\sigma) \rightarrow S(P_I)$
- 2: We begin with a stable state $M_\pi(P_\sigma)$, and can assume that $\sigma_1 = I$.
- 3: Sort the values in each column by their first coordinate.
- 4: Replace σ_2 with I .
- 5: **end procedure**

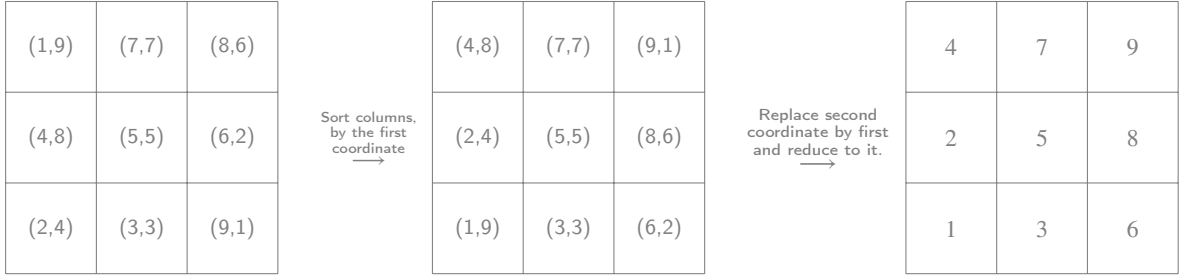


Figure 7: Illustrating the procedure given by Algorithm 2 when applied to a stable state from Figure 5.

In order to show that the set of stable states of P_I is maximally large among all sets of stable states of other point sets P_σ for any collection σ , consider the following algorithm. It defines a map $\varphi_\sigma : S(P_\sigma) \rightarrow S(P_I)$ that transforms every stable state of P_σ into a stable state of P_I .

Example 6.1. To illustrate Algorithm 2, consider, e.g., the 3×3 stable state of P_σ with $\sigma_2 = \langle 9, 4, 3, 8, 5, 2, 7, 6, 1 \rangle$ given in Figure 5. Sorting the columns by the first coordinate brings the top element of the first and third column to the bottom while leaving the second column untouched. Then, replacing the second coordinate by the first, i.e., reducing to one coordinate, yields a stable state of P_I . See Figure 7 for an illustration of this. Note how both columns and rows are ordered increasingly from bottom to top and left to right, respectively.

The function φ_σ from Algorithm 2 provides a way to relate the stable states of any point set P_σ to those of P_I . The following theorem establishes that φ_σ is injective and well-defined, i.e., that it indeed maps to $S(P_I)$.

Theorem 6.2. The function φ_σ defined in Algorithm 2 is injective and always maps to a stable state in $S(P_I)$.

PROOF. We first show that φ_σ is injective. Notice that φ_σ preserves the first coordinate of the points in each column. This means that for placement maps π and π' , if $\varphi_\sigma(M_\pi(P_\sigma)) = \varphi_\sigma(M_{\pi'}(P_\sigma))$, then the states $M_\pi(P_\sigma)$ and $M_{\pi'}(P_\sigma)$ must have the same points in each column. However, there is at most one way to arrange the points within each column to give a stable state. Thus, we must have $\pi = \pi'$.

What is left to show is that φ_σ maps stable states to stable states. Let $M_{\pi'}(P_I) = \varphi_\sigma(M_\pi(P_\sigma))$. By construction, the columns of $M_{\pi'}(P_I)$ must be sorted, so we just have to check the rows. Recall that we can express $M_{\pi'}(P_I)$ by writing a single number in each box of our grid. Assume for the sake of contradiction that there are two columns of $M_{\pi'}(P_I)$ with their respective entries as

$$M_{\pi'}(P_I) = \begin{array}{c|c|c|c|c} \dots & p_{n_2} & \dots & q_{n_2} & \dots \\ \vdots & \vdots & \dots & \vdots & \vdots \\ \dots & p_i & \dots & q_i & \dots \\ \vdots & \vdots & \dots & \vdots & \vdots \\ \dots & p_1 & \dots & q_1 & \dots \end{array}$$

such that there is some $i \in [n_2]$ with $p_i > q_i$. Since the columns are sorted increasingly, we have $p_i < p_{i+1} < \dots < p_{n_2}$. Because φ_σ did not change the position of the first coordinate between columns, for each p_j , there is some q_k such that p_j and q_k were placed in the same row in $M_\pi(P_\sigma)$. Since $M_\pi(P_\sigma)$ is stable, it follows that $p_j < q_k$. Thus, there



Figure 8: Two Young tableaux corresponding to the partition $\lambda = (3, 1)$ of $N = 4$. The left tableau is standard, while the right tableau is not.

are $n_2 - i + 1$ values q_k larger than p_i . However, if $q_i < p_i$, then all larger values have to be above q_i in the column as the column is sorted increasingly. There are only $n_2 - i$ places in the column left to store the $n_2 - i + 1$ larger values, which gives the desired contradiction.

By this theorem, we have an injective map from the set of stable states of any point set P_σ to the set of stable states of P_I . As these sets are finite, the following corollary follows immediately.

Corollary 6.3. *For $d = 2$, the size of the set $S(P_\sigma)$ is maximized when $\sigma = (I, \dots, I)$.*

There are several natural ways to extend the definition of φ_σ to all $d \in \mathbb{N}$, but we were not able to find a map satisfying both of the properties of Theorem 6.2. Still, we take it as a basis to formulate the following conjecture.

Conjecture 6.4. *Corollary 6.3 holds for all $d \in \mathbb{N}$.*

For the case of $d = 2$, stable states for the identity permutation have been extensively studied in the the context of *standard Young tableaux*, which we will briefly discuss below.

Definition 6.5. *Let $\lambda_1, \lambda_2, \dots, \lambda_k$ be a non-increasing collection of positive integers. The Young diagram $\lambda = (\lambda_1, \lambda_2, \dots, \lambda_k)$ is the diagram formed by k rows of boxes where the i -th row contains λ_i boxes.*

To match our grid conventions, will write these diagrams using *French notation*, where the rows are left-justified and are written upward from widest to narrowest (see Figure 8). We will be interested in the Young diagram formed by n_1 rows each with n_2 boxes each (i.e., the Young diagram $\lambda = (n_2^{n_1})$, where this notation is short for $\lambda = \underbrace{(n_2, \dots, n_2)}_{n_1 \text{ times}}$).

Definition 6.6. *Given a Young diagram with N boxes, a standard Young tableau (on the alphabet $\{1, \dots, N\}$) is an assignment of the integers 1 to N to the boxes of the Young diagram such that the entries increase along rows and up columns.*

Example 6.7. *In Figure 8, we consider the Young diagram for $N = 4$ and the partition $\lambda = (3, 1)$ with two different associated Young tableaux. The tableau on the left is standard while the tableau on the right is not. A straightforward combinatorial calculation shows that there are 24 fillings of this Young diagram, 3 of which are standard.*

By definition, an assignment of the numbers 1 through N to the Young diagram $\lambda = (n_2^{n_1})$ produces a standard Young tableau if and only if this assignment corresponds to a stable state $M_x(P_\sigma)$ for $\sigma = (I, I)$. The number of standard Young tableaux for a given partition of N is given by the *hook length formula*, which was first proven by Frame, Robinson, and Thrall Frame, Robinson and Thrall (1954). See (Sagan, 2001, Section 10) for more information and history about this formula. The following result is immediate from the hook length formula.

Proposition 6.8. *The number of standard Young tableaux associated with $\lambda = (n_2^{n_1})$ is precisely*

$$f^{n_1 \times n_2} := \frac{(n_1 n_2)!}{\prod_{i=1}^{n_1} \prod_{j=1}^{n_2} (n_1 + n_2 - i - j + 1)}.$$

Corollary 6.9. *When $d = 2$, the value of $|S(P_I)|$ is given by the formula from Proposition 6.8. This is also an upper bound for $|S(P_\sigma)|$ among all pairs of permutations σ .*

PROOF. The first claim is immediate from Proposition 6.8. The second claim is immediate from Corollary 6.3.

Recall that with Corollary 5.7, we have shown that the assumption made by Malheiros and Walter de Gomensoro Malheiros and Walter (2015) to prove the equivalent of Corollary 4.4 was not justified. We gave a complete proof of Theorem 4.3 above and thereby proved Corollary 4.4. However, we obtain an alternative proof of Corollary 4.5 by using Corollary 6.9.

PROOF. Recall that the proof of Theorem 4.3 built a decision-tree and bounded the minimal depth that this tree needs to be traversed to by any comparison-based algorithm \mathcal{A} in order to construct a placement π such that $M_\pi(P_\sigma)$ is stable. From Corollary 6.9, we know that any such decision-tree has to be traversed at least until the subtree has at most $|S(P_I)|$ leaves. That is, assuming $n_1 = n_2 = n$ for some $n \in \mathbb{N}$, the traversal has to take at least this many steps:

$$\begin{aligned}
 & \log(n^2!) - \log(f^{n \times n}) \\
 = & \log\left(\frac{n^2!}{f^{n \times n}}\right) \\
 = & \log\left(\prod_{i=1}^n \prod_{j=1}^n (2n - i - j + 1)\right) \\
 = & \sum_{i=1}^n \sum_{j=1}^n \log(2n - i - j + 1) \\
 = & \log(2n - 1) + \log(2n - 2) + \dots + \log(n) \\
 & + \log(2n - 2) + \log(2n - 3) + \dots + \log(n - 1) \\
 & \vdots \\
 & + \log(n) + \log(n - 1) + \dots + \log(1) \\
 = & n \log(n) + \sum_{i=1}^{n-1} (i \log(i) + i \log(2n - i)) \\
 \geq & n \log(n) + \frac{1}{2} \sum_{i=n/2}^{n-1} \frac{n}{2} \log(n) \\
 = & \Omega(n^2 \log(n)).
 \end{aligned}$$

Thereby, we have obtained the statement as made by de Gomensoro Malheiros and Walter (2015) in two different ways, for the two-dimensional case.

In the setting of $d = 2$, the map φ_σ , as constructed by Algorithm 2, gives rise to an algorithm for the enumeration of all stable states of a given point set. This enumeration algorithm is significantly faster than the naive brute force approach. Consider the procedure given in Algorithm 3.

Algorithm 3 Enumeration of Stable States

- 1: **procedure** ENUMERATE STABLE STATES(P_σ)
 - 2: Enumerate all stable states $M_\pi(I)$ of the identity I
 - 3: **for** $M_\pi(I)$ stable **do**
 - 4: Compute $\varphi_\sigma^{-1}(M_\pi(I))$
 - 5: Store it, if stable
 - 6: **end for**
 - 7: **end procedure**
-

Given this algorithm, enumeration of all stable states $M_\pi(I)$ takes time $\mathcal{O}(f^{n_1 \times n_2})$. Execution of Algorithm 2 then takes time $\mathcal{O}(n_1 n_2 \log(n_2))$. Furthermore, if $n_2 > n_1$, then we can reverse the role of n_1 and n_2 without changing the result. Thus, the total time to execute Algorithm 3 amounts to

$$f^{n_1 \times n_2} n_1 n_2 \log(\min(n_1, n_2)). \quad (1)$$

For a concrete example of the benefits of this approach, consider the case where $n_1 = n_2 = 4$. Given a point set P_σ , a naive method for enumerating stable states is to check all $16! \approx 2.09 \cdot 10^{13}$ possible placement maps. Using Algorithm 3, it suffices to compute just 870,912,000, a reduction by five orders of magnitude. One question that we expect this reduction to simplify is Open 5.12.

We will now turn to the final contribution of this section. In Corollary 6.3, we proved that $\sigma = (\mathcal{I}, \mathcal{I})$ maximizes the number of stable states for $d = 2$. However, this gives a mere example for a collection of permutations σ that realizes this maximum. Below, we will characterize a family of point sets P_σ that maximize $|S(P_\sigma)|$. We conjecture that this is a complete characterization apart from the trivial case where $n_1 = 1$ or $n_2 = 1$, and one exceptional case where $n_1 = n_2 = 2$ (see Conjecture 6.15).

First, we note the following relationship about the cardinality of two related sets of stable states, which holds in arbitrary dimension d .

Lemma 6.10. *Let $\sigma = (\sigma_1, \dots, \sigma_d)$ be a collection of permutations defining the point set P_σ . For any k , if we replace σ_k with the reverse permutation $\bar{\sigma}_k$, we do not change the number of stable states.*

PROOF. We obtain a bijection between $S(P_{(\sigma_1, \dots, \sigma_k, \dots, \sigma_d)})$ and $S(P_{(\sigma_1, \dots, \bar{\sigma}_k, \dots, \sigma_d)})$ as follows. Consider a stable state $M_\pi(P_\sigma)$. By reverting the order of all points along all k -strips, we obtain a stable state $M_{\pi'}(P_{(\sigma_1, \dots, \bar{\sigma}_k, \dots, \sigma_d)})$. Note that this operation is bijective.

Now, we focus on the two-dimensional case. Here, we can prove the following relationship between the number of stable states of two related point sets.

Lemma 6.11. *Let $\sigma = (\mathcal{I}, \sigma_2)$. Suppose for some $i \in [N]$ that $|\sigma_2(i) - \sigma_2(i + 1)| = 1$. Define σ' to be equivalent to σ , except with $\sigma'_2(i) = \sigma_2(i + 1)$ and $\sigma'_2(i + 1) = \sigma_2(i)$. Then, the point set $P_{\sigma'}$ has the same number of stable states as P_σ .*

PROOF. We will define a function ψ which gives a permutation on the placement maps. Given a placement map π , let $\mathbf{a} = (a_1, a_2)$ be the position given by $\pi^{-1}(i)$ and let $\mathbf{b} = (b_1, b_2)$ be the position given by $\pi^{-1}(i + 1)$. For any position $\mathbf{c} \in [n_1] \times [n_2]$, let

$$\psi \circ \pi(\mathbf{c}) = \begin{cases} \mathbf{b} & \text{if } \mathbf{c} = \mathbf{a} \text{ and } a_1 = b_1, \\ \mathbf{a} & \text{if } \mathbf{c} = \mathbf{b} \text{ and } a_1 = b_1, \\ \mathbf{c} & \text{otherwise.} \end{cases}$$

In other words, ψ switches the position of $(i, \sigma_2(i))$ and $(i + 1, \sigma_2(i + 1))$ when these points are in the same column, and leaves everything the same otherwise.

We claim that $M_\pi(P_\sigma)$ is stable if and only if $M_{\psi \circ \pi}(P_{\sigma'})$ is stable. The result then follows from the fact that ψ is invertible.

To prove the claim, there are two possibilities to consider. First, if $a_1 = b_1$, then the only difference between $M_\pi(P_\sigma)$ and $M_{\psi \circ \pi}(P_{\sigma'})$ is that the first entry of $(i, \sigma(i))$ and $(i + 1, \sigma(i + 1))$ are swapped. By the assumption that $a_1 = b_1$, these points must be in the same column and thus must be in different rows. Combining this with the fact that there are no numbers between i and $i + 1$ implies that this change does not impact stability.

The other case to consider is where $a_1 \neq b_1$. This time, the only difference between $M_\pi(P_\sigma)$ and $M_{\psi \circ \pi}(P_{\sigma'})$ is that the second entry of $(i, \sigma(i))$ and $(i + 1, \sigma(i + 1))$ are swapped. By the condition that $|\sigma_2(i) - \sigma_2(i + 1)| = 1$, there can be no values between $\sigma_2(i)$ and $\sigma_2(i + 1)$. The claim follows from combining this fact with the fact that the swapped entries are not in the same column.

Lemma 6.12. *A point set $\sigma = (\mathcal{I}, \sigma_2)$ can be obtained from $(\mathcal{I}, \mathcal{I})$ by repeated applications of Lemma 6.11 if and only if $|\sigma_2(i) - i| \leq 1$ for all $i \in [N]$.*

PROOF. The forward direction is straightforward to prove by induction. If $\sigma_2(1) = 1$, then we can ignore the first entry. Alternatively, if $\sigma_2(1) = 2$, then we must have $\sigma_2(2) = 1$ and we can ignore the first two entries (after a single swap for $i = 1$).

For the reverse direction, it suffices to show that if $|\sigma_2(j) - j| \leq 1$ for all $j \in [N]$, then $|\sigma'_2(j) - j| \leq 1$ for all $j \in [N]$ (using the notation from Lemma 6.11). Let i be the index such that $\sigma'_2(i) = \sigma_2(i + 1)$ and $\sigma'_2(i + 1) = \sigma_2(i)$.

For $k \notin \{i, i+1\}$, it is immediate that $|\sigma'_2(k) - k| = |\sigma_2(k) - k| \leq 1$. Thus, we must prove that $|\sigma'_2(i) - i| \leq 1$ and $|\sigma'_2(i+1) - (i+1)| \leq 1$.

By the conditions of Lemma 6.11, the only way to arrive at a contradiction would be if $\sigma_2(i) = i+1$ and $\sigma_2(i+1) = i+2$ or if $\sigma_2(i) = i-1$ and $\sigma_2(i+1) = i$. However, if $\sigma_2(i) = i+1$ and $\sigma_2(i+1) = i+2$, then for all $k > i+1$, we must have $\sigma_2(k) \in [i+3, N]$. This is impossible by the pigeonhole principle, and the other case is analogous.

Lemmas 6.10 and 6.12 can also be used to produce enumerative results.

Definition 6.13. Let $(n_1, \dots, n_d) \in \mathbb{N}^d$, then we write $\mathcal{M}(n_1, \dots, n_d)$ for the number of point sets with the maximal number of stable states on the $(n_1 \times \dots \times n_d)$ grid.

Let Fib_k denote the k^{th} Fibonacci number (with $\text{Fib}_1 = \text{Fib}_2 = 1$, and $\text{Fib}_k = \text{Fib}_{k-1} + \text{Fib}_{k-2}$). It is known that for any N , there are exactly Fib_{N+1} functions satisfying Lemma 6.12. This follows from a simple recursion after splitting into the case where $\sigma(1) = 1$ and the case where $\sigma(1) = 2$. By Lemma 6.11, this gives Fib_{N+1} point sets with the same number of stable states as the identity. Furthermore, as long as $N > 2$, we obtain Fib_{N+1} more point sets with this property by applying Lemma 6.10. Thus, we have the following corollary.

Corollary 6.14. For $(n_1, n_2) \in \mathbb{N}^2$ with $n_1 n_2 > 2$, we have $\mathcal{M}(n_1, n_2) \geq 2 \text{Fib}_{n_1 n_2 + 1}$, where Fib_k denotes the k^{th} Fibonacci number.

Based on computer data and heuristic arguments, we suspect that this bound is almost always tight. In particular, we conjecture the following.

Conjecture 6.15. For $(n_1, n_2) \in \mathbb{N}^2$ with $n_1 > 1$, $n_2 > 1$, and $n_1 n_2 > 4$, we have $\mathcal{M}(n_1, n_2) = 2 \text{Fib}_{n_1 n_2 + 1}$.

Note that if $n_1 = 1$ or $n_2 = 1$, then all $N!$ point sets have a unique stable state. When $n_1 = n_2 = 2$, Corollary 6.14 gives 10 point sets with the maximum of 2 stable states. However, there are also 2 more point sets with this property corresponding to the permutations $\sigma_2 = (2, 4, 1, 3)$ and $\sigma_2 = (3, 1, 2, 4)$.

With the help of a computer, we verified that $\mathcal{M}(2, 3) = 26 = 2 \text{Fib}_7$, and we suspect that since Conjecture 6.15 holds for this grid, it should hold for larger grids as well.

7. Conclusion

In this article, we have investigated the neighborhood grid data structure Joselli et al. (2009, 2015) from a combinatorial viewpoint. This perspective allowed us to show that an ordered arrangement of points in the grid, a stable state, can be constructed in polynomial time for any grid shape in any dimension, see Theorem 3.1). In terms of run time of the construction of stable states, we proved a lower bound, see Theorem 4.3, and proved time optimality of the construction algorithm for the case where all side lengths of the grid are equal, see Corollary 4.4. Previous work did provide corresponding arguments, which, however, rested on a faulty uniqueness assumption. We disprove this assumption by showing that all point sets placed in grids of size 4×4 or larger admit to at least two stable states, see Propositions 5.5 and 5.6. However, we can also bound the number of stable states that a point set admits to from above, see Proposition 6.8.

In the two-dimensional case, for one grid side being of length $n_1 = 2$, we were able to prove that there always exist certain point sets that have a unique stable state. However, it is an open question to derive a closed-form expression of the number of these stable states, see Open 5.10. Similarly, for $n_1 = 3$, an exact formula is unknown and it remains an open question whether there are unique stable states for any n_2 . We do have a candidate point set that only allows for one unique placement in all instances we checked, but we were not able to prove this generally. This candidate motivates our Conjecture 5.11 that for n_3 there are also always unique stable states for any n_2 . On the other hand, given a larger grid, with side lengths $n_1, n_2 \geq 4$, it remains unclear what the minimum number of stable states is among all respective point sets, see Open 5.12. Furthermore, while several of our results hold for arbitrary dimension d , the investigation of unique stable states in higher-dimensional configurations is left as future work.

We also proved that the maximum number of stable states is achieved by the identity permutation in the two-dimensional case, and suspect that this is true for arbitrary dimension, see Conjecture 6.4. After this, we gave a partial classification of point sets which also achieve the maximum set by the identity permutation, which we suspect is a full

classification unless $n_1 = 1$, $n_2 = 1$, or $n_1 = n_2 = 2$, see Conjecture 6.15. This conjecture suggests an elegant formula for the number of such point sets related to the Fibonacci sequence.

Other questions rise in the investigation of the neighborhood grid data structure. These include, for instance, the neighborhood quality achieved by the grid. As neighborhood queries are only answered approximately, it remains unclear how close the reported neighbors are. Furthermore, in an application context, points representing, e.g., moving particles, will not move for large distances in simulations. Thus, it is probably advised to use an adaptive strategy based on the grid in the previous iteration and not to re-build the grid in every iteration of the simulation. However, these questions are out scope for the combinatorial consideration presented here and will have to be tackled in a separate work.

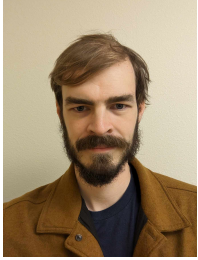
Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. DMS-1439786 and the Alfred P. Sloan Foundation award G-2019-11406 while the author was in residence at the Institute for Computational and Experimental Research in Mathematics in Providence, RI, during the Illustrating Mathematics program. This research was partially funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 455095046. We acknowledge the support by the DFG SFB/TR 109 ‘Discretization in Geometry and Dynamics’, ECMath, BMS, and the German National Academic Foundation. The authors would like to thank Shagnik Das for helpful discussions regarding the proof of Theorem 4.3 for the case of $n_i = n$ and $d = 2$, i.e., the statement that is now Corollary 4.4. Furthermore, the authors would like to thank Mathijs Molenaar and Mark van de Ruit for creating a C++ implementation to experimentally investigate stable states. Also, we would like to thank Per Alexandersson and Sam Hopkins for answering some questions we had about higher-dimensional generalizations of Young tableaux. Finally, we would like to thank the anonymous reviewers for their input on the article.

References

- Boris, J., 1986. A vectorized “near neighbors” algorithm of order n using a monotonic logical grid. *Journal of Computational Physics* 66, 1–20.
- DeMillo, R.A., Eisenstat, S., Lipton, R.J., 1978. On small universal data structures and related combinatorial problems, in: *Proc. Johns Hopkins Conf. on Inf. Sci. and Syst.*, pp. 408–411.
- Frame, J.S., Robinson, G.d.B., Thrall, R.M., 1954. The hook graphs of the symmetric group. *Canadian Journal of Mathematics* 6, 316–324.
- de Gomensoro Malheiros, M., Walter, M., 2015. Simple and efficient approximate nearest neighbor search using spatial sorting, in: *2015 28th SIBGRAPI Conference on Graphics, Patterns and Images*, IEEE. pp. 180–187.
- Joselli, M., Junior, J.R.d.S., Clua, E.W., Montenegro, A., Lage, M., Pagliosa, P., 2015. Neighborhood grid: A novel data structure for fluids animation with gpu computing. *Journal of Parallel and Distributed Computing* 75, 20–28.
- Joselli, M., Passos, E.B., Zamith, M., Clua, E., Montenegro, A., Feijó, B., 2009. A neighborhood grid data structure for massive 3d crowd simulation on gpu, in: *2009 VIII Brazilian Symposium on Games and Digital Entertainment*, IEEE. pp. 121–131.
- Malashina, A., 2021. The combinatorial analysis of n-gram dictionaries, coverage and information entropy based on the web corpus of english. *Baltic J. Modern Computing* 9, 363–376.
- Sagan, B., 2001. *The symmetric group: representations, combinatorial algorithms, and symmetric functions*. volume 203. Springer Science & Business Media.
- Skrodzki, M., 2019a. The k-d tree data structure and a proof for neighborhood computation in expected logarithmic time. CoRR abs/1903.04936. URL: <http://arxiv.org/abs/1903.04936>, arXiv:1903.04936.
- Skrodzki, M., 2019b. Neighborhood Data Structures, Manifold Properties, and Processing of Point Set Surfaces. Ph.D. thesis. Freie Universität Berlin. URL: <https://refubium.fu-berlin.de/handle/fub188/25320>.
- Skrodzki, M., Reitebuch, U., Polthier, K., Das, S., 2018. Combinatorial and asymptotical results on the neighborhood grid data structure, in: Korman, M., Mulzer, W. (Eds.), *EuroCG18 Extended Abstracts*, pp. 30:1–30:6. URL: https://conference.imp.fu-berlin.de/eurocg18/download/paper_30.pdf.
- Uhlmann, J., 2020. On the monotonic lagrangian grid as antecedent to the neighborhood grid. *Journal of Parallel and Distributed Computing* 142, 13–15.
- Wolter, J.D., 1991. A combinatorial analysis of enumerative data structures for assembly planning., in: *ICRA*, Citeseer. pp. 611–618.
- Woo, T., 1985. A combinatorial analysis of boundary data structure schemata. *IEEE Computer Graphics and Applications* 5, 19–27.

Combinatorial and Asymptotic Results on the Neighborhood Grid



Alex McDonough received a PhD in Mathematics from Brown University in 2021 and is currently a postdoctoral scholar at the University of Oregon. His research is primarily in algebraic combinatorics and discrete geometry.



Ulrich Reitebuch has a diploma in mathematics from Technische Universität Berlin. He works at the department of mathematics and computer science at Freie Universität Berlin and is part of the Mathematical Geometry Processing group. He is interested in differential geometry, discrete geometry, tessellations, symmetries, and connections between mathematics and arts.



Martin Skrodzki studied mathematics and computer science in Dortmund (Germany), Laredo (TX, USA), and Berlin (Germany), where he graduated with a Dr. rer. nat. in 2019 at the Freie University Berlin. Afterwards, he was a postdoctoral researcher at the Institute for Computational and Experimental Mathematics (ICERM) at Brown University (RI, USA), the RIKEN institute (Japan) funded by the German Academic Scholarship Foundation, and TU Delft (Netherlands) funded by the Walter-Benjamin-Program of the German Research Foundation. He is currently an assistant professor in the group of Computer Graphics and Visualization at TU Delft. His research interests include the use of illustrations in mathematics, visualization of high-dimensional data, discrete geometry processing as well as interactions between mathematics and arts.