Counting Lattice Animals in High Dimensions

Sebastian Luther[1] **and Stephan Mertens**[1,2]

[1]Institut für Theoretische Physik, Otto-von-Guericke Universität, PF 4120, 39016 Magdeburg, Germany
[2]Santa Fe Institute, 1399 Hyde Park Rd, Santa Fe, NM 87501, USA

E-mail: SebastianLuther@gmx.de, mertens@ovgu.de

Abstract. We present an implementation of Redelemeier's algorithm for the enumeration of lattice animals in high dimensional lattices. The implementation is lean and fast enough to allow us to extend the existing tables of animal counts, perimeter polynomials and series expansion coefficients in d-dimensional hypercubic lattices for $3 \le d \le 10$. From the data we compute formulas for perimeter polynomials for lattice animals of size $n \le 11$ in arbitrary dimension d. When amended by combinatorial arguments, the new data suffices to yield explicit formulas for the number of lattice animals of size $n \le 14$ and arbitrary d. We also use the enumeration data to compute numerical estimates for growth rates and exponents in high dimensions that agree very well with Monte Carlo simulations and recent predictions from field theory.

1. Introduction

A *polyomino* of size n is an edge-connected set of n squares on the square lattice, a *polycube* of size n is a face-connected set of n cubes in the cubic lattice. Polyominoes and polycubes are a classical topic in recreational mathematics and combinatorics [1]. In statistical physics and percolation theory, polyominoes and polycubes are called *lattice animals* [2, 3]. Lattice animals are not restricted to dimension 2 or 3: a d-dimensional lattice animal of size n is a set of n face-connected hypercubes on $\mathbb{Z}^d$.

In this contribution we will address the problem of counting the number of *fixed* animals of size n in dimension $d > 2$. *Fixed* animals are considered distinct if they have different shapes or orientations. *Free* animals, on the other hand, are distinguished only by shape, not by orientation. Figure 1 shows all fixed polyominoes of size 3 and all free polycubes of size 4.

We denote the number of d-dimensional fixed animals of size n by $A_d(n)$. There is no formula for $A_d(n)$, but we know that $A_d(n)$ grows exponentially with n. Using subadditivity and concatenation arguments [4], one can show that there are constants $1 < \lambda_d < \infty$ such that

$$\lim_{n \to \infty} \sqrt[n]{A_d(n)} = \lambda_d. \tag{1}$$

The constant λ_2 is known as Klarner's constant.

A slightly stronger result due to Madras [5] asserts that

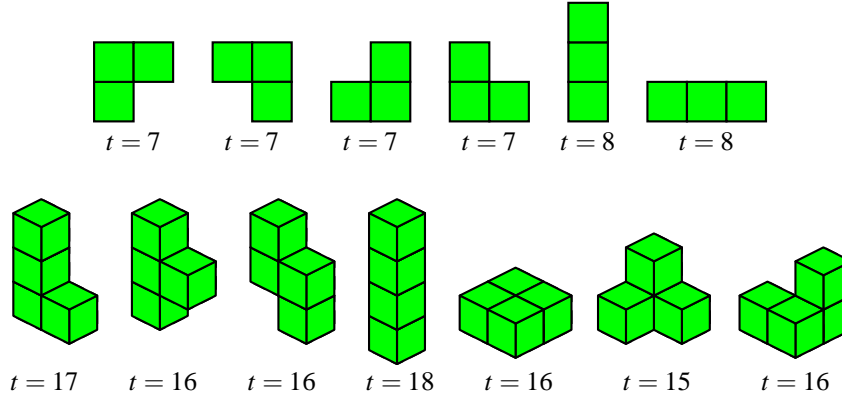$$\lim_{n \to \infty} \frac{A_d(n+1)}{A_d(n)} = \lambda_d. \tag{2}$$

Figure 1. All *fixed* polyominoes of size $n = 3$ (top) and all *free* polycubes of size $n = 4$ (bottom) and their perimeters.

Intuitively, the growth rate λ_d should grow with the coordination number $2d$ of the lattice. In fact, in [6] it is shown that

$$\lambda_d = 2d\mathrm{e} - o(d), \tag{3}$$

and in the same paper it is conjectured that $\lambda_d = (2d - 3)\mathrm{e} + O(1/d)$. For finite d, however, we know only lower and upper bounds for λ_d. Numerical estimates for λ_d can be derived from extrapolating $A_d(n+1)/A_d(n)$, which is one motivation to compute $A_d(n)$ for n as large as possible. We will try our hands at that in Section 6.

In percolation theory one is interested in counting lattice animals of a given size n according to their perimeter t, i.e., to the number of adjacent cells that are empty (see Figure 1). If each cell of the lattice is occupied independently with probability p, the average number of clusters of size n per lattice site reads

$$\sum_t g_{n,t}^{(d)} p^n (1-p)^t, \tag{4}$$

where $g_{n,t}^{(d)}$ denotes the number of fixed d-dimensional lattice animals of size n and perimeter t. The g's define the *perimeter polynomials*

$$P_d(n,q) = \sum_t g_{n,t}^{(d)} q^t. \tag{5}$$

We can easily compute $A_d(n)$ from the perimeter polynomial $P_d(n,q)$ through

$$A_d(n) = P_d(n,1) = \sum_t g_{n,t}^{(d)}. \tag{6}$$

In fact we can compute $A_d(n+1)$ from the perimeter polynomials up to size n,

$$A_d(n+1) = \frac{1}{n+1} \sum_{m \le n} m \sum_t g_{m,t}^{(d)} \binom{t}{n+1-m} (-1)^{n-m}. \tag{7}$$

This equation follows from the observation that below the percolation threshold p_c, each occupied lattice site belongs to some finite size lattice animal,

$$p = \sum_{n=1}^{\infty} n p^n P_d(n, 1-p) \tag{8}$$

for $p < p_c$. The right hand side is a power series in p, and Equation (7) follows from the fact that the coefficient of p^{n+1} must be zero.

As we will see in the next section, the algorithm for counting lattice animals keeps track of the perimeter anyway. Hence it is reasonable to use the algorithm to compute the perimeter polynomials and to apply (7) to get an extra value of A_d.

2. The Algorithm

The classical algorithm for counting lattice animals is due to Redelmeier [7]. Originally developped for the square lattice, Redelmeier's algorithm was later shown to work on arbitrary lattices and in higher dimensions [8] and to be efficiently parallelizable [9]. For two dimensional lattices there is a much faster counting method based on transfer matrices [10], but for $d \geq 3$ Redelmeier's algorithm is still the most efficient known way to count lattice animals.

The algorithm works by recursively generating all lattice animals up to a given size n_{max}. Given an animal of size n, the algorithm generates animals of size $n + 1$ by adding a new cell in the perimeter of the given animal. The lattice sites that are available for extending the current animal are stored in a set U called the *untried set*. To avoid generating the same fixed animal more than once, lattice sites that have previously been added to the untried set are marked on the lattice.
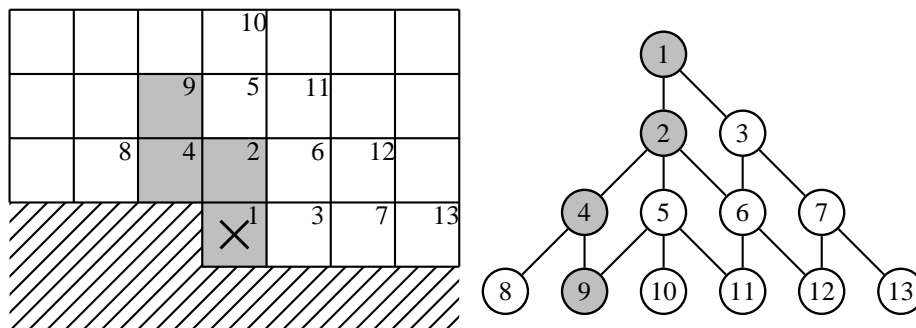

Figure 2. Part of the square lattice that can be reached from animals up to size 4 (left). The number of lattice animals of size $n \leq 4$ equals the number of subgraphs in the neighborhood graph (right) that contain vertex 1.

In order to break the translational symmetry we demand that the initial site, which is contained in all animals, is an extremal site with respect to the lexicographic order of lattice coordinates. Figure 2 illustrates how this can be achieved in the square lattice. We simply block all lattice sites from further consideration that are in a row below the initial site (marked with a cross) or in the same row and to the left of the initial site. The generalization to $d > 2$ is straightforward. These blocked sites are never added to the untried set, but they need to be taken into account when we compute the perimeter t.

We start with all lattice sites being marked "free" or "blocked," except for the initial site, which is marked "counted." Furthermore $n = 1, t = 1$ and the initial site being the only element of the untried set U. Redelemeier's algorithm works by invoking the following routine with this initial settings:

- Iterate until U is empty:
  (1) Remove a site s from U.

(2) $F :=$ set of "free" neighbors, $B :=$ set of "blocked" neighbors of s. $N := |U| + |B|$.

(3) Count new cluster: increase $g_{n,t+N-1}$ by one.

(4) If $n < n_{\max}$:

    (a) Mark all sites in F and B as "counted."

    (b) Call this routine recursively with $U' = U \cup F$, $n' = n + 1$ and $t' = t + N - 1$.

    (c) Relabel sites in F as "free" and sites in B as "blocked."

- Return.

Since the algorithm generates each lattice animal explicitly, its running time scales like $A_d(n)$. This exponential complexity implies hard limits for the accessible animal sizes. All we can do is to keep the prefactor in the time complexity function small, i.e., to implement each step of Redelemeier's algorithm as efficiently as possible by using an appropriate data structure for the untried set and for the lattice, see [8].

Another crucial element of tuning Redelmeier's algorithm is the computation of the neighborhood of a lattice cell. In a recent paper [11], Aleksandrowicz and Barequet observed that Redelmeier's algorithm can be interpreted as the counting of subgraphs in a graph that represents the neighborhood relation of the lattice. Figure 2 illustrates this for the square lattice.

For any lattice, the neighborhood graph can be precomputed and be represented as an adjacency list which is then fed to the actual subgraph counting algorithm. That way the computation of the neighbors of a lattice cell is taken out of the counting loop, and the prefactor in the exponential scaling is reduced.

The size of the neighborhood graph or, equivalently, the number of lattice points required to host lattice animals of size n scales like $\Theta(n^d)$. Aleksandrowicz and Barequet [11] claimed that this exponential growth of memory with d represents a serious bottleneck for Redelmeier's algorithm in high dimensions. In a subsequent paper [12], they therefore present a variation of the algorithm that avoids the storage of the full graph by computing the relevant parts of the graph on demand. This cuts down the space complexity to a low order polynomial in d, but it forfeits the gain in speed that can be obtained by precomputing the complete neighborhood graph.

We claim that in practice the space complexity of Redelmeier's algorithm is no bottleneck. The reason is that the prefactor in the $\Theta(n^d)$ scaling can be made small enough to hold the complete graph in memory for all values of n and d for which $A_d(n)$ is computable in reasonable time.

The key observation is that in [11], Aleksandrowicz and Barequet used a hyper-cube of side length $2n$ of the lattice to host the lattice animals, whereas a hyper-sphere of radius n suffices. This is a significant difference, as can be seen from the analogous situation in $\mathbb{R}^d$. Here the volume of a cube is much larger than the volume of the inscribed Euclidean sphere,

$$\frac{\text{volume hypercube}}{\text{volume inscribed hypersphere}} = \frac{2^d \Gamma(\frac{d}{2} + 1)}{\pi^{d/2}} \simeq \left(\frac{2d}{\pi e}\right)^{d/2} \frac{1}{\sqrt{\pi d}}. \tag{9}$$

In the lattice $\mathbb{Z}^d$, the number of lattice points in a cube of side length $2n + 1$ is

$$C(d,n) = (2n+1)^d. \tag{10}$$

Let $B(d,n)$ denote the number of lattice sites that are n steps or less away from the origin. This "volume of the crystal ball" can be computed recursively via

$$B(1,n) = 2n + 1$$
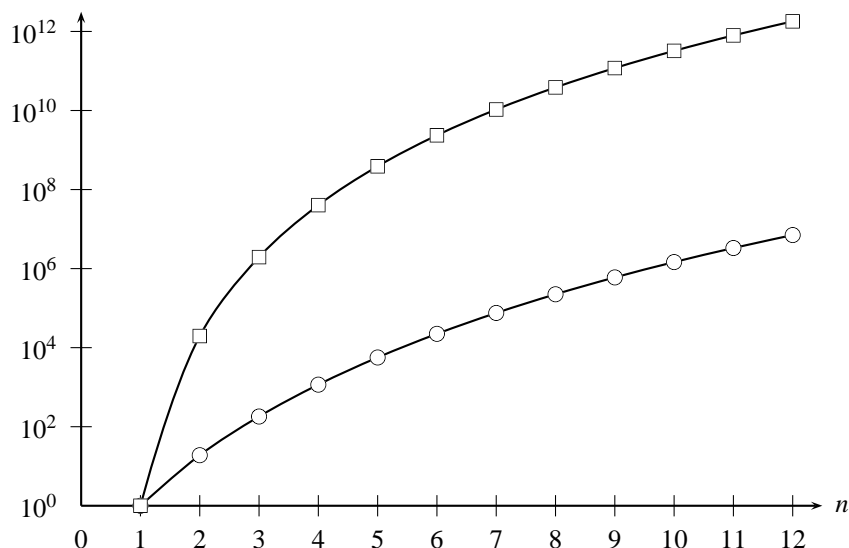
$$B(d,n) = B(d-1,n) + 2\sum_{k=0}^{n-1} B(d-1,k), \tag{11}$$

Figure 3. Volume needed to cage animals of size n on the hypercubic lattice of dimension $d = 9$. Cubic ($\square$) versus spherical cages ($\bigcirc$).

which reflects the fact that the crystal ball in dimension d can be decomposed into $(d-1)$-dimensional slices whose diameter decreases with increasing distance from the central slice. The recursion (11) tells us that $B(d,n)$ is a polynomial in n of degree d which can easily be computed, see A001845 to A001848 on oeis.org for the polynomials for $d = 3, \ldots, 6$. Note that $B(d,n)$ can also be computed through the generating function [13]

$$\frac{(1+x)^d}{(1-x)^{d+1}} = \sum_{n=0}^{\infty} B(d,n)x^n. \tag{12}$$

The number of lattice sites needed to cage lattice animals of size n is very close to $B(d, n-1)/2$ for spherical cages and $C(d, n-1)/2$ for cubical cages. Figure 3 shows both numbers for the case $d = 9$. As you can see, $d = 9$ and $n = 12$ requires memory on the Terabyte scale if one uses cubic cages, but only a few Megabytes for spherical cages.

3. Performance

Our implementation of the Redelmeier algorithm consists of two programs. The first program computes the neighborhood graph of a specified lattice and writes this graph as an adjacency list into a file. The second program reads this file and computes the corresponding perimeter polynomials. The programs are written in C++ and can be downloaded from the project webpage [15].

When run on a laptop with an Intel® Core™ 2 Duo CPU at 2 GHz, the program enumerates perimeter polynomials at a rate of roughly $2 \cdot 10^7$ animals per second. This means that generating and counting one lattice animal and measuring its perimeter takes about 100 clock cycles, which is reasonable for a program compiled from C++.

At this rate, our laptop needs 35 days to enumerate the perimeter polynomials for $d = 9$ and $n \leq 11$ (see Table 2). Computing the next perimeter polynomial ($n = 12$) would take more than three years. Note that according to Figure 3, the neighborhood graph for $d = 9$ and

	Perimeter Polynomial		$A_d(n)$	
d	old n_{max}	new n_{max}	old n_{max}	new n_{max}
3	15	18	18	19
4	10	15	15	16
5	9	14	13	15
6	8	14	10	15
7	8	13	10	14
8		11	8	12
9		11	4	12
10		11		12

Table 1. Range of perimeter polynomials and animal numbers in dimensions $d \geq 3$ that have been found by exhaustive enumerations. The old perimeter polynomials are from [8] and [14], the old values of $A_d(n)$ are from [11] and [12].

n	$A_6(n)$	$A_7(n)$	$A_8(n)$	$A_9(n)$
1	1	1	1	1
2	6	7	8	9
3	66	91	120	153
4	901	1484	2276	3309
5	13881	27468	49204	**81837**
6	231008	551313	1156688	**2205489**
7	4057660	11710328	28831384	**63113061**
8	74174927	259379101	750455268	**1887993993**
9	1398295989	5933702467	**20196669078**	**58441956579**
10	27012396022	139272913892	**558157620384**	**1858846428437**
11	**532327974882**	**3338026689018**	**15762232227968**	**60445700665383**
12	**10665521789203**	**81406063278113**	**453181069339660**	**2001985304489169**
13	**227093585071305**	**2014611366114053**		
14	**4455636282185802**	**50486299825273271**		
15	**92567760074841818**			

Table 2. Number of lattice animals in the hypercubic lattice for $d = 6 \ldots 9$ obtained by *direct enumeration*. New results in boldface, the numbers of smaller animals are from [11, 12] and references therein. Note that in [16], $A_8(n)$ and $A_9(n)$ for $n \leq 9$ were computed rather than enumerated by the same method that we will use in Section 4 to extend this table to $n \leq 14$ and all values of d.

$n = 12$ easily fits into the memory of a run-of-the-mill laptop. These numbers illustrate that for for all practical purposes, the bottleneck of Redelemeier's algorithm is time, not memory.

Using a parallel implementation [9] that we ran on a Linux cluster with 128 Intel® Xeon® 3.2 GHz CPUs, or for the most demanding computations, on a SciCortex SC5832 with 972 MIPS64 6-core nodes, we could extend the table of known perimeter polynomials and animal counts considerably, see Table 1. The new values for $d \leq 5$ are

$$A_3(19) = \mathbf{651\,459\,315\,795\,897},$$
$$A_4(16) = \mathbf{692\,095\,652\,493\,483},$$
$$A_5(14) = \mathbf{227\,093\,585\,071\,305},$$
$$A_5(15) = \mathbf{3\,689\,707\,621\,144\,614}.$$

The numbers for $6 \leq d \leq 9$ are given in Table 2, the corresponding perimeter polynomials

can be found on the project webpage [15]. Before we evaluate the results, we will discuss a combinatorial argument that allows us to extend the enumeration data considerably.

Note that the most demanding computation in this paper was the enumeration of the perimeter polynomial for $n = 14$ in $d = 6$. On a single core of a MIPS64, this enumeration would have taken 77 CPU years, on our Laptop from above it would still have taken about 7 CPU years. In practice we used a parallel implementation that ran on many cores (and several different machines) such that no computation took longer than two weeks wall clock time.

4. Proper Animals

A lattice animal of size n can't span more than $n-1$ dimensions. This simple observation allows us to derive explicit formulas for $A_d(n)$ for fixed n. Obviously $A_d(1) = 1$ and $A_d(2) = d$. A lattice animal of size $n = 3$ is either a one-dimensional "stick" with d possible orientations or "L-shaped" and spanning 2 out of d dimensions. Within these 2 dimensions there are 4 possible orientations for the L-shaped animal (see Figure 1), hence

$$A_d(3) = d + 4\binom{d}{2} = 2d^2 - d\,.$$

For $n = 4$, we have again the "stick" that lives in one dimension, 17 animals that span 2 dimensions and 32 animals that span 3 dimensions:

$$A_d(4) = d + 17\binom{d}{2} + 32\binom{d}{3} = \frac{16}{3}d^3 - \frac{15}{2}d^2 + \frac{19}{6}d\,.$$

In general we can write

$$A_d(n) = \sum_{i=0}^{d} \binom{d}{i} \mathrm{DX}(n,i)\,, \tag{13}$$

where $\mathrm{DX}(n,i)$ denotes the number of fixed *proper* animals of size n in dimension i. An animal is called proper in dimension d if it spans all d dimensions. Equation (13) is due to Lunnon [17]. If we know $A_d(n)$ for a given n and $d \leq d_{\max}$, we can use (13) to compute $\mathrm{DX}(n,d)$ for the same value of n and all $d \leq d_{\max}$, and vice versa.

Since $\mathrm{DX}(n,i) = 0$ for $i \geq n$, Lunnon's equation tells us that $A_d(n)$ is a polynomial of degree $n-1$ in d, and since $A_0(n) = 0$ for $n > 1$, it suffices to know the values $A_1(n), A_2(n), \ldots, A_{n-1}(n)$ to compute the polynomial $A_d(n)$. From our enumeration data (Table 1), we can compute these polynomials up to $A_d(11)$, see Table 3.

In order to compute $A_d(12)$, we need to know $A_{11}(12)$ or equivalently, $\mathrm{DX}(12,11)$. The latter can actually be computed with pencil and paper. That's because an animal of size 12 in 11 dimensions has to span a new dimension with each of its cells to be proper. In particular, its cells can't form loops. Hence computing $\mathrm{DX}(12,11)$ is an exercise in counting trees. This is true for $\mathrm{DX}(n,n-1)$ in general, so let's compute this function.

The adjacency graph of a lattice animal of size n is an edge labeled graph with n vertices, in which each vertex represents a cell of the animal and two vertices are connected if the corresponding cells are neighbors in the animal. Every edge of the adjacency graph is labeled with the dimension along which the two cells touch each other.

In the case $\mathrm{DX}(n,n-1)$, every pair of adjacent cells must span a new dimension. Therefore the corresponding adjacency graph contains exactly $n-1$ edges, i.e. it is a tree, and each edge has a unique label. There are two directions for each dimensions that we represent by the orientation of the edge in the tree. Hence $\mathrm{DX}(n,n-1)$ equals the number of directed, edge-labeled trees of size n, where in our context "directed" means that each edge has an arbitrary orientation in addition to its label.

$$A_d(2) = d$$

$$A_d(3) = 2d^2 - d$$

$$A_d(4) = \frac{16}{3}d^3 - \frac{15}{2}d^2 + \frac{19}{6}d$$

$$A_d(5) = \frac{50}{3}d^4 - 42d^3 + \frac{239}{6}d^2 - \frac{27}{2}d$$

$$A_d(6) = \frac{288}{5}d^5 - 216d^4 + \frac{986}{3}d^3 - 231d^2 + \frac{926}{15}d$$

$$A_d(7) = \frac{9604}{45}d^6 - 1078d^5 + \frac{20651}{9}d^4 - \frac{14927}{6}d^3 + \frac{120107}{90}d^2 - \frac{827}{3}d$$

$$A_d(8) = \frac{262144}{315}d^7 - \frac{26624}{5}d^6 + \frac{132320}{9}d^5 - \frac{65491}{3}d^4 + \frac{1615991}{90}d^3 - \frac{113788}{15}d^2 + \frac{52589}{42}d$$

$$A_d(9) = \frac{118098}{35}d^8 - 26244d^7 + \frac{447903}{5}d^6 - \frac{511082}{3}d^5 + \frac{23014949}{120}d^4 - \frac{1522261}{12}d^3$$
$$+ \frac{38839021}{840}d^2 - \frac{30089}{4}d$$

$$A_d(10) = \frac{8000000}{567}d^9 - \frac{2720000}{21}d^8 + \frac{14272000}{27}d^7 - \frac{11092360}{9}d^6 + \frac{239850598}{135}d^5$$
$$- \frac{14606026}{9}d^4 + \frac{1067389643}{1134}d^3 - \frac{42595493}{126}d^2 + \frac{2804704}{45}d$$

$$A_d(11) = \frac{857435524}{14175}d^{10} - \frac{67319318}{105}d^9 + \frac{2884481974}{945}d^8 - \frac{380707987}{45}d^7 + \frac{40341440233}{2700}d^6$$
$$- \frac{1260803635}{72}d^5 + \frac{79118446751}{5670}d^4 - \frac{19252021283}{2520}d^3 + \frac{17126616179}{6300}d^2 - \frac{7115086}{15}d$$

$$A_d(12) = \frac{509607936}{1925}d^{11} - \frac{15925248}{5}d^{10} + \frac{607592448}{35}d^9 - \frac{1956324864}{35}d^8 + \frac{2930444704}{25}d^7$$
$$- \frac{2522387284}{15}d^6 + \frac{17894522696}{105}d^5 - \frac{1242881121}{10}d^4$$
$$+ \frac{22272055467}{350}d^3 - \frac{4225468993}{210}d^2 + \frac{181356011}{66}d$$

$$A_d(13) = \frac{551433967396}{467775}d^{12} - \frac{75047226332}{4725}d^{11} + \frac{166095324499}{1701}d^{10} - \frac{48436628461}{135}d^9$$
$$+ \frac{49499551181119}{56700}d^8 - \frac{1335959158369}{900}d^7 + \frac{248648897740349}{136080}d^6 - \frac{25156285613453}{15120}d^5$$
$$+ \frac{757565736903221}{680400}d^4 - \frac{5607318230581}{10800}d^3 + \frac{12648671104037}{83160}d^2 - \frac{135165335}{6}d$$

$$A_d(14) = \frac{4628074479616}{868725}d^{13} - \frac{23612624896}{297}d^{12} + \frac{3309261190144}{6075}d^{11} - \frac{304034058496}{135}d^{10}$$
$$+ \frac{12648090831712}{2025}d^9 - \frac{553376997376}{45}d^8 + \frac{758347226205724}{42525}d^7 - \frac{2633038200122}{135}d^6$$
$$+ \frac{98388569956577}{6075}d^5 - \frac{2734657007119}{270}d^4 + \frac{11824147558382}{2475}d^3 - \frac{560344373791}{330}d^2$$
$$+ \frac{97500388612}{273}d$$

Table 3. Number of lattice animals of sizes $2, \ldots, 14$ in hypercubic lattices of dimension d. The polynomials for $n \leq 11$ have been obtained by direct enumeration and confirm those listed in [6]. Polynomials for $n > 11$ have been computed from enumeration data and known values of $DX(n, n-k)$.

The number of *vertex* labeled trees of size n is given by n^{n-2}, the famous fomula published by Cayley in 1889 [18]. The number of *edge* labeled trees seems to be much less known, at least it is proven afresh in recent papers like [19]. The following nice derivation is from [6]. Start with a vertex labeled tree of size n and mark the vertex with label n as the root. Then shift every label smaller than n from its vertex to the incident edge towards the root. This gives an edge labeled tree with a single vertex marked (the root). Since the mark can be on any vertex, the number of edge labeled trees equals the number of vertex labeled trees divided by the number of vertices. According to Cayley's formula, this number is n^{n-3}. And since each directed edge can have two directions, we get

$$\text{DX}(n,n-1) = 2^{n-1} n^{n-3}. \tag{14}$$

This formula has been known in the statistical physics community for a long time [20]. We used it to compute $\text{DX}(12,11)$ and then $A_d(12)$ (Table 3).

We can proceed further along this line. To compute $A_d(13)$ we have to extend our enumeration data by $A_{12}(13),\dots,A_8(13)$ or equivalently by $\text{DX}(13,12),\dots,\text{DX}(13,8)$. What we need are formulas $\text{DX}(n,n-k)$ for $k > 1$.

For $k > 1$, there is no longer a simple correspondence between edge labeled trees and proper animals. We need to take into account that there are edge labels with the same value, that the adjacency graph may contain loops, and that some labeled trees represent a self-overlapping and therefore illegal lattice animal. A careful consideration of these issues yields

$$\text{DX}(n,n-2) = 2^{n-3} n^{n-5}(n-2)(9-6n+2n^2), \tag{15}$$

see [6] for the derivation of (15).

For $k > 2$, the computation of $\text{DX}(n,n-k)$ gets very complicated and is better left to a computer. In Appendix A we show that

$$\text{DX}(n,n-k) = 2^{n-2k+1} n^{n-2k-1} g_k(n), \tag{16}$$

where $g_k(n)$ is a polynomial of degree $3k-3$. Hence we can compute g_k from $3k-2$ data points, like the values of $\text{DX}(n,n-k)$ for $n = k,\dots,4k-3$. Our enumeration data suffices to compute g_2 and g_3 with this method, but not g_4.

However, there is a trick that allows us to compute g_k from many fewer data points. The free energy

$$f_n = \frac{1}{n}\log A_d(n)$$

has a well defined $1/d$ expansion whose coefficients depend on n. If we *assume* that these coefficients are bounded in the limit $n \to \infty$, most of the coefficients in g_k are fixed, and we only need to know $k+1$ data points to fully determine g_k. See Appendix B for the details of this argument. In our case this enables us to compute g_k up to $k = 7$, see Table 4, and consequently $A_d(13)$ and $A_d(14)$, see Table 3.

We actually know all data to compute $A_d(15)$ with the exception of the number $A_7(15)$. On our laptop, the enumeration of the missing number $A_7(15)$ would take about 80 years. On a parallel system with a few hundred CPUs this would still take several months, which is not out of reach. Computing the formula for $A_d(16)$, however, is definitely beyond the power of our machinery.

Before we turn our attention to the analysis of the enumeration data we note that Lunnon's equation (13) has a corresponding equation for perimeter polynomials:

$$g_{n,t}^{(d)} = \sum_i \binom{d}{i} G_{n,t-2(d-i)n}^{(i)}. \tag{17}$$

$$g_2(n) = (n-2)(9-6n+2n^2)$$

$$g_3(n) = \frac{n-3}{6}(-1560+1122n-679n^2+360n^3-104n^4+12n^5)$$

$$g_4(n) = \frac{n-4}{6}(204960-114302n+41527n^2-17523n^3+7404n^4-2930n^5+828n^6-128n^7+8n^8)$$

$$g_5(n) = \frac{n-5}{360}(-3731495040+1923269040n-535510740n^2+150403080n^3-42322743n^4$$
$$+12397445n^5-4062240n^6+1335320n^7-356232n^8+62240n^9-6000n^{10}+240n^{11})$$

$$g_6(n) = \frac{n-6}{360}(1785362705280-939451308048n+248868418932n^2-56265094748n^3$$
$$+11984445891n^4-2448081038n^5+535284255n^6-127651774n^7+33940138n^8$$
$$-9580440n^9+2398912n^{10}-440688n^{11}+51856n^{12}-3424n^{13}+96n^{14})$$

$$g_7(n) = \frac{n-7}{45360}(-156017752081551360+85163968967728896n-22517704978919136n^2$$
$$+4585470174542376n^3-851686123590540n^4+146137469433102n^5$$
$$-24441080660523n^6+4148836864606n^7-747463726205n^8$$
$$+149724735468n^9-33793043592n^{10}+8322494124n^{11}-1946680944n^{12}$$
$$+363148352n^{13}-47679184n^{14}+4019904n^{15}-193536n^{16}+4032n^{17})$$

Table 4. Polynomials $g_k(n)$ that appear in $\text{DX}(n,n-k)$ (16). The polynomials $g_2,\ldots,g_6$ can be found as $g_{k,0}$ in Appendix 2 of [21]. As far as we know, the polynomial g_7 has not been published before. See the Appendix of this paper for the method how to compute the g_k.

$G_{n,t}^{(d)}$ denotes the number of proper d-dimensional animals of size n and perimeter t. Since $G_{n,t}^{(d)} = 0$ for $d > n-1$, we can write

$$P_d(n,q) = q^{2dn-2(n-1)}\sum_{i=1}^{n-1}\binom{d}{i}\sum_t G_{n,t}^{(i)}q^{t-2-2n(i-1)}. \tag{18}$$

For a given value of n, (18) represents the perimeter polynomial for general dimension d. Our enumeration data allowed us to compute the $G_{n,t}^{(d)}$ and hence the formulas (18) for $n \leq 11$ (see [15] for the data), extending the previously known formulas for $n \leq 7$ [14]. A computation of the next formula $P_d(12,q)$ requires the knowledge of the perimeter polynomials for $d \leq 11$ and $n \leq 12$. The enumeration for $d = 11$ and $n = 12$ alone would take ca. 38 years on our laptop.

5. Mean Cluster Size

From the perimeter polynomials we can compute moments of the cluster statistics like the mean cluster size

$$S(p) = \frac{1}{p}\sum_{n=1}^{\infty}n^2p^nP_d(n,1-p) = \sum_r b_d(r)p^r. \tag{19}$$

The coefficients of the series expansion read

$$b_d(r) = \sum_{n=1}^{r+1}n^2\sum_t g_{n,t}^{(d)}\binom{t}{r+1-n}(-1)^{r+1-n}$$

$$= (r+1)^2A_d(r+1)+\sum_{n=1}^{r}n^2\sum_t g_{n,t}^{(d)}\binom{t}{r+1-n}(-1)^{r+1-n}. \tag{20}$$

r	$d=3$	$d=4$	$d=5$	$d=6$
1	6	8	10	12
2	30	56	90	132
3	114	320	690	1272
4	438	1832	5290	12252
5	1542	9944	39210	115332
6	5754	55184	293570	1091472
7	19574	290104	2135370	10159252
8	71958	1596952	15839690	95435172
9	233574	8237616	113998170	**883192392**
10	870666	45100208	**840643170**	**8258076192**
11	2696274	**229502616**	**6017266290**	**76196541732**
12	10375770	**1254330128**	**44178511010**	**710151162432**
13	30198116	**6307973352**	**315024296150**	**6540805549192**
14	**122634404**	**34574952952**	**2307462163110**	**60831844077672**
15	**327024444**	**171364602736**		
16	**1460721616**			
17	**3347244554**			
18	**17795165832**			

r	$d=7$	$d=8$	$d=9$
1	14	**16**	**18**
2	182	**240**	**306**
3	2114	**3264**	**4770**
4	24542	**44368**	**74322**
5	280238	**595632**	**1146834**
6	3210074	**8012384**	**17720514**
7	36394302	**107053424**	**272530194**
8	414610014	**1434259248**	**4198328082**
9	**4685293438**	**19125485024**	**64487361906**
10	**53201681162**	**255662267296**	**991886672898**
11	**600207546946**	**3405928921264**	**1522687319670**
12	**6800785109594**	**45466350310880**	**233996383280898**
13	**76649757121000**		

Table 5. Series coefficients of the mean cluster size $S(p) = \sum_r b_r p^r$ in hypercubic lattices of dimension d. New values in boldface, older values from [8] ($d=3$) and [14] ($d=4,\dots,7$) and references therein.

Since we can compute $A_d(r+1)$ from the perimeter polynomials $P_d(n,q)$ for $n \leq r$ via (7), we can also compute the series coefficients $b_d(r)$ from this set of perimeter polynomials. If we happen to know $A_d(r+2)$, we can get an extra coefficient through

$$b_d(r+1) = (r+2)A_d(r+2)$$
$$+ \sum_{n=1}^{r} n(n-r-1)\sum_t g_{n,t}^{(d)} \binom{t}{r+2-n}(-1)^{r-n}. \tag{21}$$

This formula can be derived by solving (7) for $\sum_t g_{n,t} t$ and plugging the result into (20).

We used (21) to extend the table of coefficients (Table 5) for $d \geq 8$, since here we know perimeter polynomials only up to $n = 11$, but the cluster numbers A_d up to $n = 14$.
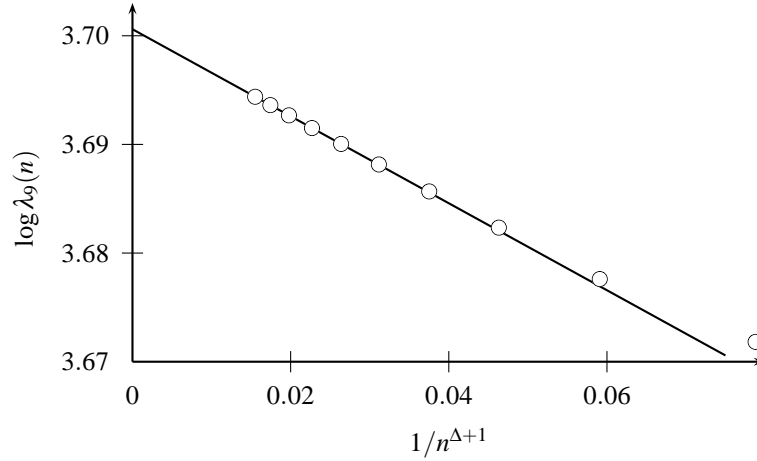
Figure 4. Growth rate λ_d for $d = 9$. The symbols are $\lambda_9(n)$ computed from Eqs. (23). The correction exponent $\Delta = 0.58$ and the line are the result of a numerical fit to the three leftmost data points.
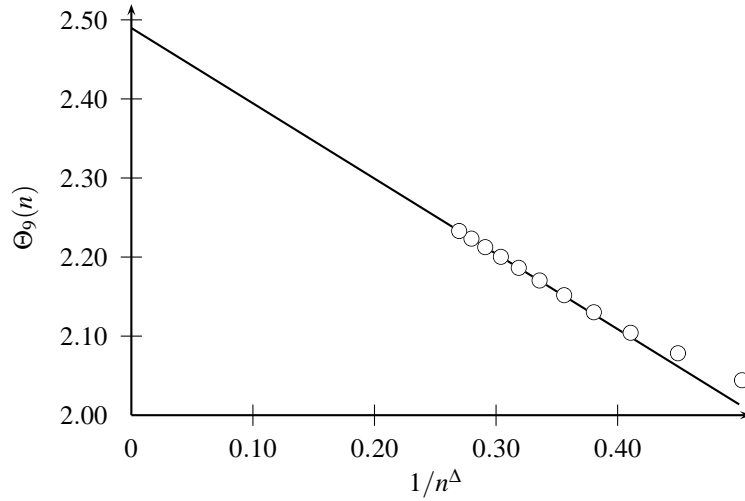

Figure 5. Exponent Θ_d for $d = 9$. The symbols are $\theta_9(n)$ computed from Eqs. (23). The correction exponent $\Delta = 0.50$ and the line are the result of a numerical fit to the three leftmost data points.

6. Growth Rates and Exponents

The cluster numbers $A_d(n)$ are expected to grow asymptotically as

$$A_d(n) \sim C\lambda_d^n \, n^{-\Theta_d} \left(1 + \frac{b}{n^\Delta} + \text{corrections}\right), \tag{22}$$

where the exponents Θ_d and Δ are universal constants, i.e., their value depends on the dimension d, but not on the underlying lattice, while C and b are nonuniversal, lattice dependent quantities [22]. The universality facilitates the computation of Θ_d for some values of d using field theoretic arguments. We know $\Theta_3 = 3/2$ [23, 24], $\Theta_4 = 11/6$ [25] and

	$\log \lambda_d$		Θ_d	
d	enum.	MC	enum.	exact, MC
3	2.12169	2.1218588(25)	1.489	3/2
4	2.58750	2.587858(6)	1.796	11/6
5	2.92254	2.922318(6)	2.113	2.080(7)
6	3.17838	3.178520(4)	2.232	2.261(12)
7	3.38403	3.384080(5)	2.357	2.40(2)
8	3.55484	3.554827(4)	2.441	5/2
9	3.70057	3.700523(10)	2.489	5/2

Table 6. Growth rates λ_d and exponents θ_d obtained from extrapolating the enumeration data. The columns marked MC contain values from large scale Monte Carlo simulations [27, 28].

$\Theta_d = 5/2$ (the value for the Bethe lattice) for $d \geq d_c = 8$, the critical dimension for animal growth [26].

The enumeration data for $A_d(n)$ can be used to estimate both λ_d and Θ_d. For that we compute $\lambda_d(n)$ and $\Theta_d(n)$ as the solutions of the system

$$\log A_d(n-k) = \log C + (n-k)\log \lambda_d(n) - \Theta_d(n)\log(n-k) \tag{23}$$

for $k = 0, 1, 2$. We need three equations to eliminate the constant $\log C$. Growth rate λ_d and exponent Θ_d are obtained by extrapolating the numbers $\lambda_d(n)$ and $\Theta_d(n)$ to $n \to \infty$. From (22) we expect that

$$\log \lambda_d(n) \sim \log \lambda_d + \frac{b}{n^{\Delta+1}} \tag{24}$$

for large values of n. We used the data points $\lambda_d(n)$ for the three largest values of n to fit the parameters $\log \lambda_d$, b and Δ in (24). A plot of $\log \lambda_d(n)$ versus $n^{-\Delta-1}$ (Figure 4) then shows that the data points in fact scale like (24). The resulting estimates for $\log \lambda_d$ are listed in Table 6. They agree very well with the high precision values from large scale Monte Carlo simulations [27, 28].

The same approach can be used to compute the exponent Θ_d. Here we expect

$$\Theta_d(n) \sim \Theta_d + \frac{b}{n^{\Delta}}. \tag{25}$$

Again we used the data points $\Theta_d(n)$ for the three largest values of n to fit the parameters Θ_d, b and Δ. Figure 5 shows that $\Theta_d(n)$ in fact scales like (25). The resulting estimates for Θ_d (Table 6) deviate from the Monte Carlo results and the exact values by no more than 3%.

The estimates for λ_d and Θ_d based on the current known values of $A_d(n)$ are much more precise than previous extrapolations based on shorter sequences $A_d(n)$, see [14, 16].

7. Conclusions

We have seen that the memory requirements of Redelmeier's algorithm can be kept low by using hyperspherical regions of the lattice. Even in high dimensions, the limiting resource in Redelmeier's algorithm is time, not space.

We used a lean and efficient implementation of Redelmeier's algorithm to compute new perimeter polynomials in hypercubic lattices of dimensions $d \leq 10$. We have used these new perimeter polynomials together with combinatorial arguments based on proper animals to compute new values of the cluster numbers $A_d(n)$ and new formulas for $A_d(n)$ for $n \leq 14$ and
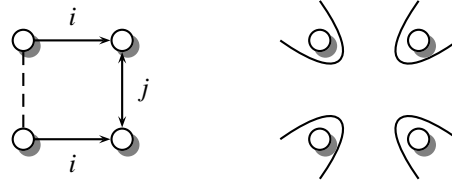
Figure A1. A labeled spanning tree that contains a part $\bigcirc \xrightarrow{i} \bigcirc \xleftrightarrow{j} \bigcirc \xleftarrow{i} \bigcirc$ corresponds to a 4-loop in the lattice animals, a quadrilateral that lives in the i-j plane (left). To count the number of spanning trees with such a 4-loop, the edges of the quadrilateral are removed and the vertices of the quadrilateral are considered the root vertices of disconnected trees (right). The number of the latter is given by (A.1) with $\ell = 4$.

arbitrary d. We've also used the new data to compute formulas for the perimeter polynomials $P_d(n;q)$ for $n \leq 11$ and arbitrary d. We haven't shown these formulas here, but you can download them from the project webpage [15].

We've also used our data to compute the formula for $DX(n, n-7)$, the number of proper animals of size n in dimension $n-7$, and new coefficients in the series expansion of the mean cluster size $S(p)$.

Based on the enumeration data, we've finally computed numerical values for the growth rates λ_d and the critical exponents θ_d that agree very well with the results of Monte Carlo simulations and field theoretical predictions.

All in all we have explored the limits of computerized counting of lattice animals in dimensions $d \geq 3$. Any significant extension of the results presented here would require either a considerable amount of CPU time or an algorithmic breakthrough comparable to the transfer matrix methods for $d = 2$.

Appendix A. Formulas for $DX(n, n-k)$: Structure

In the physics literature like [21], equation (16) is usually *assumed* to be true just because it is supported by the available enumeration data. But as a matter of fact, one can actually *motivate* (16) using the type of arguments that were used in [6] to prove the formula for $DX(n, n-2)$. The idea is to show that the leading order of $DX(n, n-k)$ is $\sim 2^n n^{n+k-4}$ whereas the lowest order contributions are $\sim 2^n n^{n-2k-1}$. This is exactly the range of terms in (16) if g_k is a polynomial of degree $3(k-1)$.

Equation (16) is obviously correct for $k = 1$ (with $g_1 = 1$). For $k > 1$, we can still represent lattice animals by trees, namely the spanning trees of their adjacency graph. Each spanning tree is again an edge labeled tree, but this time there are only $n - k$ labels for $n - 1$ edges, i.e., $k - 1$ edges will carry a label that is also used elsewhere in the tree. Consider a tree whose $n - 1$ edges are labeled with numbers $1, \ldots, n - 1$. If we identify each of the high value labels $n - k + 1, n - k + 2, \ldots, n - 1$ with one of the low value labels $1, \ldots, n - k$, we get the right set of labels. Since there are $(n - k)^{k-1}$ ways to this, the number of edge labeled trees with $n - k$ distinct labels scales like $n^{k-1} n^{n-3} = n^{n+k-4}$ in leading order. With two directions for every edge we get $2^n n^{n+k-4}$ for the leading order in $DX(n, n-k)$.

For $k > 1$, a proper animal can contain loops. Figure A1 shows the simplest case: a loop that arises because one dimension (i) is explored twice. The result is a quadrilateral that lives in the i-j plane of the lattice. In the spanning tree, this corresponds to a part that is labeled $\bigcirc \xrightarrow{i} \bigcirc \xleftrightarrow{j} \bigcirc \xleftarrow{i} \bigcirc$. Since graphs with loops have several spanning trees, our count of edge labeled trees overcounts the number of proper animals. We need to subtract
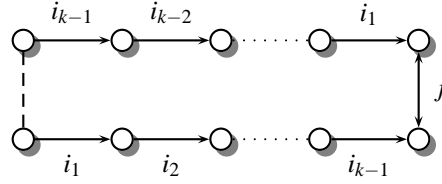
Figure A2. An animal that contributes to $\mathrm{DX}(n, n-k)$ uses at most $k-1$ dimensions more than once, and the longest loop arises when each of these $k-1$ dimensions is explored twice. Such a loop contains $2k$ vertices, as shown here.

some contributions from loopy animals. The idea is to break up the part of the spanning tree that corresponds to the loop and to separately count the number of trees that are attached to the vertices on the loopy part.

Consider an animal with a loop that contains ℓ cells and one of its spanning trees. If we remove all the edges from the spanning tree that connect the vertices in the loop, the remaining graph is a forest, i.e., a collection of trees, where each tree is rooted in one of the ℓ vertices. The forest has $n - \ell$ vertices with a total of $n - \ell$ edges, and it is independent from the way that the root vertices have been connected in the loop.

Now the number of ordered sequences of $\ell \geq 1$ directed rooted trees with a total of $n - \ell$ edges and $n - \ell$ distinct edge labels is

$$2^{n-\ell} n^{n-\ell-1} \ell. \tag{A.1}$$

See [6] for a proof of (A.1). The lowest order corrections come from those animals for which the number ℓ of cells in a loop is maximal. This is the case for a loop that joins all $k-1$ non-unique edge labels, see Figure A2. The number of vertices in these loops is $\ell = 2k$, hence the lowest order corrections are $\sim 2^{n-2k} n^{n-2k-1}$.

If we want the exact number of spanning trees for loopy animals, we need to count the number of ways to reconnect the roots of the forest to form a single tree. But since this number does depend on k but not on n and we are interested only in the scaling with n, we don't need to enter this discussion here. The same is true for animals that contain several small loops instead of a single loop of maximal length. If we apply the separation trick to one of the shorter loops, we get a scaling of order larger than $\sim 2^{n-2k} n^{n-2k-1}$, and the resulting forest is then labeled with fewer labels than edges, which increases the order even further. So the lowest order corrections from loops come in fact from single loops of maximum length, and these contributions are of order $\sim 2^{n-2k} n^{n-2k-1}$, as claimed in (16).

Besides the non-uniqueness of spanning trees for loopy graphs, there is another type of error that needs to be corrected: some edge labeled trees correspond to animals with overlapping cells, i.e., to illegal animals. For instance, if a spanning tree contains the subtree

$$\bullet \xrightarrow{i} \circ \xleftarrow{i} \bullet \quad \text{or} \quad \bullet \xleftarrow{i} \circ \xrightarrow{i} \bullet,$$

the two $\bullet$'s represent the very same cell of the animal. But these "colliding" configurations can be interpreted as 3-loops, or more generally, as ℓ loops, and counted in the same way as the legal loops above. Again the lowest order contributions come from the longest "colliding" loops which are formed by $k-1$ labels assigned to two edges each and arranged like

$$\bullet \xrightarrow{i_1} \circ \xrightarrow{i_2} \circ \cdots \circ \xrightarrow{i_{k-1}} \circ \xleftarrow{i_1} \circ \xleftarrow{i_2} \circ \cdots \circ \xleftarrow{i_{k-1}} \bullet.$$

These longest collision loops contain $2k-1$ vertices of the tree (representing $2k-2$ cells of the animal). According to (A.1), their number scales like $2^{n-2k+1} n^{n-2k}$, one order above the lowest order of legal loops.

This concludes the motivation of (16). Note that a *proof* of (16) would require a thorough analysis to exclude contributions outside the range covered by (16).

Appendix B. Formulas for $DX(n, n-k)$: Coefficients

Having established the fact that $DX(n, n-k)$ is given by (16) we still have to determine the coefficients of the polynomials $g_k(n)$. Since g_k has degree $3(k-1)$, it seems that we need to know the $3k-2$ values $DX(k,0), DX(k+1,1), \ldots, DX(4k-3, 3k-3)$ to compute the coefficients. In terms of our enumeration data this means knowledge of $A_d(n)$ for $n \leq 4k-3$ and $d \leq n-k$. The data in Table 2 suffices to compute the coefficients of g_k for $k = 2, 3$, but not for $k \geq 4$. Nevertheless we can compute g_k for $k \leq 7$ by *assuming* that the "free energy"

$$\lim_{n \to \infty} \frac{1}{n} \log A_d(n)$$

has a well defined $1/d$ series expansion. This approach has been used to compute g_k for $k \leq 6$ from much less enumeration data in [21] and [29], and we used it to compute g_7 from the new enumeration data. Since the method hasn't been described in detail elsewhere, we provide a description in this Appendix.

Let's start with Lunnon's equation, which tells us that $A_d(n)$ is a polynomial of degree $n-1$ in d with coefficients that depend on n. For $d \geq n$ we have

$$A_d(n) = \sum_{k=1}^{n-1} DX(n,k) \binom{d}{k} = \sum_{j=1}^{n-1} a_j(n) d^j \tag{B.1}$$

with

$$a_j(n) = \sum_{k=1}^{n-1} \frac{DX(n,k)}{k!} \begin{bmatrix} k \\ j \end{bmatrix}, \tag{B.2}$$

where $\begin{bmatrix} k \\ j \end{bmatrix}$ denotes the Stirling number of the first kind. In particular we get

$$a_{n-1}(n) = \frac{DX(n, n-1)}{(n-1)!}$$

$$a_{n-2}(n) = \frac{DX(n, n-2)}{(n-2)!} + \frac{DX(n, n-1)}{(n-1)!} \begin{bmatrix} n-1 \\ n-2 \end{bmatrix}$$

$$a_{n-3}(n) = \frac{DX(n, n-3)}{(n-3)!} + \frac{DX(n, n-2)}{(n-2)!} \begin{bmatrix} n-2 \\ n-3 \end{bmatrix} + \frac{DX(n, n-1)}{(n-1)!} \begin{bmatrix} n-1 \\ n-3 \end{bmatrix}$$

and so on. From (B.1) we get

$$A_d(n) = a_{n-1}(n) d^{n-1} \left(1 + \sum_{j=1}^{n-2} \frac{a_{n-1-j}}{a_{n-1}} d^{-j} \right),$$

and with $\ln(1+x) = x + x^2/2 + x^3/3 \ldots$ this gives the $1/d$ series for the "free energy"

$$\frac{1}{n} \ln A_d(n) = \left(1 - \frac{1}{n} \right) \ln d + \frac{1}{n} \ln a_{n-1}(n) + \frac{1}{n} \frac{a_{n-2}(n)}{a_{n-1}(n)} \frac{1}{d} + \mathcal{O}\left(\frac{1}{d^2} \right). \tag{B.3}$$

We *assume* that all coefficients in this series remain bounded in the limit $n \to \infty$. This is definitely true for the zeroth order term:

$$\lim_{n \to \infty} \frac{1}{n} \ln a_{n-1}(n) = 1 + \ln 2.$$

For the first order coefficient we get

$$\frac{1}{n}\frac{a_{n-2}}{a_{n-1}} = \frac{n-1}{n}\frac{\mathrm{DX}(n,n-2)}{\mathrm{DX}(n,n-1)} + \frac{1}{n}\begin{bmatrix}n-1\\n-2\end{bmatrix}$$
$$= \left(1-\frac{1}{n}\right)\left(\frac{g_2(n)}{4n^2} - \frac{n-2}{2}\right).$$

This is only bounded if the $g_2(n)$ term balances the second term, i.e., if the n^3 coefficient of g_2 equals 2. Using also the fact that $\mathrm{DX}(2,0) = 0$, we can write

$$g_2(n) = (n-2)(2n^2 + bn + c).$$

To compute the remaining coefficents, we only need to know $\mathrm{DX}(3,1) = 1$ and

$$\mathrm{DX}(4,2) = A_2(4) - 2 = 17$$

to get

$$g_2(n) = (n-2)(2n^2 - 6n + 9).$$

The postulation of bounded coefficients in the series (B.3) has saved us from knowing the value $\mathrm{DX}(5,3)$ to compute g_2. How much does it help us to compute g_k?

The polynomial g_k enters the series expansion (B.3) via the term

$$\frac{1}{n}\frac{\mathrm{DX}(n,n-k)}{\mathrm{DX}(n,n-1)}\frac{(n-1)!}{(n-k)!} = 2^{2-2k}n^{1-2k}g_k(n)\underbrace{(n-1)(n-2)\cdots(n-k+1)}_{\Theta(n^{k-1})}$$

in the coefficient of $d^{-(k-1)}$. The leading order of this term is $n^{-k}g_k(n)$. All terms of degree larger than k in the polynomial g_k lead to unbounded contributions to the series coefficient that need to be counterbalanced by other terms. These balancing terms always exist, a fact that gives additional support for the claim of bounded coefficients. The coefficients of the terms of order larger than k in g_k are therefore computable from the known terms $g_{k-1}(n), g_{k-2}(n), \ldots$ that also enter the same coefficient. Only the $k+1$ low order terms of g_k are not fixed by the postulate of bounded coefficients and we need $k+1$ data points $\mathrm{DX}(k,0), \mathrm{DX}(k+1,1), \ldots, \mathrm{DX}(2k,k)$ to complete g_k.

Our enumeration data suffices to compute g_7 (see Table 4). The computation of g_8 requires knowledge of $\mathrm{DX}(15,7)$ and $\mathrm{DX}(16,8)$, or in terms of $A_d(n)$,

$$\mathrm{DX}(15,7) = A_7(15) - 572521427068702741$$

and

$$\mathrm{DX}(16,8) = A_8(16) + 48366334433679758 - 56 * A_5(16)$$
$$+ 28 * A_6(16) - 8 * A_7(16).$$

References

[1] Solomon W. Golomb. *Polyominoes*. Princeton University Press, Princeton, New Jersey, 2nd edition, 1994.

[2] Dietrich Stauffer and Amnon Aharony. *Introduction to Percolation Theory*. Taylor & Francis, 2nd edition, 1992.

[3] Anthony J. Guttmann, editor. *Polygons, Polyominoes and Polycubes*, volume 775 of *Lecture Notes in Physics*. Springer-Verlag, Heidelberg, 2009.

[4] David A. Klarner. Cell growth problems. *Canadian Journal of Mathematics*, 19:851–863, 1967.

[5] Neal Madras. A pattern theorem for lattice clusters. *Annals of Combinatorics*, 3:357–384, 1999.

[6] Ronnie Barequet, Gill Barequet, and Günther Rote. Formulae and growth rates of high-dimensional polycubes. *Combinatorica*, 30(3):257–275, 2010.

[7] D. Hugh Redelmeier. Counting polyominoes: Yet another attack. *Discrete Mathemetics*, 36(2):191–203, 1981.

[8] Stephan Mertens. Lattice animals: A fast enumeration algorithm and new perimeter polynomials. *J. Stat. Phys.*, 58(5/6):1095–1108, 1990.

[9] Stephan Mertens and Markus E. Lautenbacher. Counting lattice animals: A parallel attack. *J. Stat. Phys.*, 66:669–678, 1992.

[10] Iwan Jensen. Enumerations of lattice animals and trees. *Journal of Statistical Physics*, 102(3/4):865–881, 2001.

[11] Gadi Aleksandrowicz and Gill Barequet. Counting *d*-dimensional polycubes and nonrectangular planar polyominoes. *International Journal of Computational Geometry & Applications*, 19(3):215–229, 2009.

[12] Gadi Aleksandrowicz and Gill Barequet. Counting polycubes without the dimensionality curse. *Discrete Mathematics*, 309:4576–4583, 2009.

[13] J. H. Conway and N. J. A. Sloane. Low-dimensional lattices. VII coordination sequences. *Proceedings of the Royal Society A*, 453:2369–2389, 1997.

[14] D.S. Gaunt, M.F. Sykes, and Heather Ruskin. Percolation processes in *d*-dimensions. *Journal of Physics A: Mathematical and General*, 9(11):1899–1911, 1976.

[15] Stephan Mertens. Lattice animals. http://www.ovgu.de/mertens/research/animals, 2011.

[16] D.S. Gaunt. The critical dimension for lattice animals. *Journal of Physics A: Mathematical and General*, 13:L97–L101, 1980.

[17] W.F. Lunnon. Counting multidimensional polyominoes. *The Computer Journal*, 18(4):366–367, 1975.

[18] Arthur Cayley. A theorem on trees. *Quarterly Journal of Pure and Applied Mathematics*, 23:376–378, 1889.

[19] Peter J. Cameron. Counting two-graphs related to trees. *Electronic Journal of Combinatorics*, 2:R4, 1995.

[20] Michael E. Fisher and John W. Essam. Some cluster size and percolation problems. *Journal of Mathematical Physics*, 2:609–619, 1961.

[21] P.J. Peard and D.S. Gaunt. $1/d$-expansions for the free energy of lattice animal models of a self-interacting branched polymer. *Journal of Physics A: Mathematical and General*, 28:6109–6124, 1995.

[22] J. Adler, Y. Meir, A.B. Harris, A. Aharony, and J.A.M.S. Duarte. Series study of random animals in general dimensions. *Physical Review B*, 38, 4941–4954 1988.

[23] Giorgio Parisi and Nicolas Sourlas. Critical behavior of branched polymers and the Lee-Yang edge singularity. *Physical Review Letters*, 46(14):871–874, 1981.

[24] John Z. Imbrie. Dimensional reduction and crossover to mean-field behavior for branched polymers. *Annales Henri Poincaré*, 4:S445–S458, 2003.

[25] Deepak Dhar. Exact solution of a directed-site animals-enumeration problem in three dimensions. *Physical Review Letters*, 51(10):853–856, 1983.

[26] T. C. Lubensky and Joel Isaacson. Statistics of lattice animals and dilute branched polymers. *Physical Review A*, 20(5):2130–2146, 1979.

[27] Hsiao-Ping Hsu, Walter Nadler, and Peter Grassberger. Simulations of lattice animals and trees. *Journal of Physics A: Mathematical and General*, 38:775–806, 2005.

[28] Hsiao-Ping Hsu, Walter Nadler, and Peter Grassberger. Statistics of lattice animals. *Computer Physics Communications*, 169:114–116, 2005.

[29] D.S. Gaunt and P.J. Peard. $1/d$-expansions for the free energy of weakly embedded site animal models of branched polymers. *Journal of Physics A: Mathematical and General*, 33:7515–7539, 2000.