Efficient counting of LEGO structures

Mikkel Abrahamsen and Søren Eilers

March 30, 2007

Abstract

We investigate experimentally the growth regimes of the number of LEGO structures which can be constructed contiguously from n blocks of equal shape and color.

1 Introduction

1.1 Background

The number of LEGO structures which can be constructed contiguously from n blocks of equal shape and color is an interesting but rather elusive combinatorial object. For instance, if the number $T_{b\times w}(n)$ is defined as the count of all contiguous structures made of n LEGO blocks of size $b \times w$, identified up to rotation in the XY-plane and translation, a value such as $T_{2\times 4}(10)$ is unknown although it is only, in all likelihood, a 17-digit number. The lack of apparent recursive structure in the problem leaves us little combinatorial machinery to understand and compute such numbers.

Our interest in these quantities stems from the observation by the second named author that a claim made by LEGO Company for decades to the effect that $T_{2\times 4}(6)$ equalled 102981500 was false. The authors in synchronized efforts, but independently, computed the correct value $T_{2\times 4}(6) = 915103765$. However, rather than the individual values our interest has shifted towards understanding the **asymptotic behavior** of each integer sequence $(T_{b\times w}(n))_{n\in\mathbb{N}}$.

Even this remains shrouded in mystery. We know only that certain associated sequences grow supermultiplicatively, and hence that quantities

$$h_{b\times w} = \exp\left(\lim_{n\to\infty} \frac{\log T_{b\times w}(n)}{n}\right)$$

are well-defined. Furthermore, it was proved in [1] that $78.32 \leq h_{2\times 4} \leq 191.35$.

We offer here experimental results concerning numbers such as $h_{b\times w}$ with a particular emphasis on how these numbers vary with b and w.

1.2 Overview of the paper

We shall present the results of extensive experimentation on these matters, develop conjectures based on the experimental results, and describe the algorithms involved.

Since there are a number of possible pitfalls in computing numbers $T_{b\times w}(n)$ we shall describe an algorithm in Section 3 below. We have devoted much effort to refining this basic algorithm to reduce computing times and shall describe our results in this direction as well. Taken together, our methods have lead to significant speedup, and we have been able to produce counts for a collection of relatively small block sizes, which have been submitted to *The Online Encyclopaedia of Integer Sequences* (see [4]).

However, since run times henceforth grow exponentially in n we are not able to compute more than 10 terms in but a few of these sequences, even though we have allowed computing times up to 450 CPU hours on an otherwise idle mainframe computer. Thus we have been forced to develop Monte Carlo methods to count far enough to give qualified input to a theoretical approach to understanding the asymptotics of sequences such as $(T_{b\times w}(n))_{n\in\mathbb{N}}$. These methods are described in Section 4 below.

Our experiments strongly suggest that the growth of $(T_{b\times w}(n))_{n\in\mathbb{N}}$ follows the scheme $C \cdot H^{n-1}n^P$ in all dimensions. Adopting this as a standing hypothesis we can give good estimates for $H = h_{b\times w}$ which are consistent with our theoretical knowledge. These estimates clearly indicate that $h_{b\times w}$ varies with the dimension of the block in a straightforward but nontrivial way given by a quadratic expression in b and w. Our best estimates for C and P vary very little, and the estimates for P are rather close to $-3/2$.

The appearance of this power indicates that the generating function of the $T_{b\times w}(n)$ has a square root singularity at $1/h_{b\times w}$ as is often the case in counting

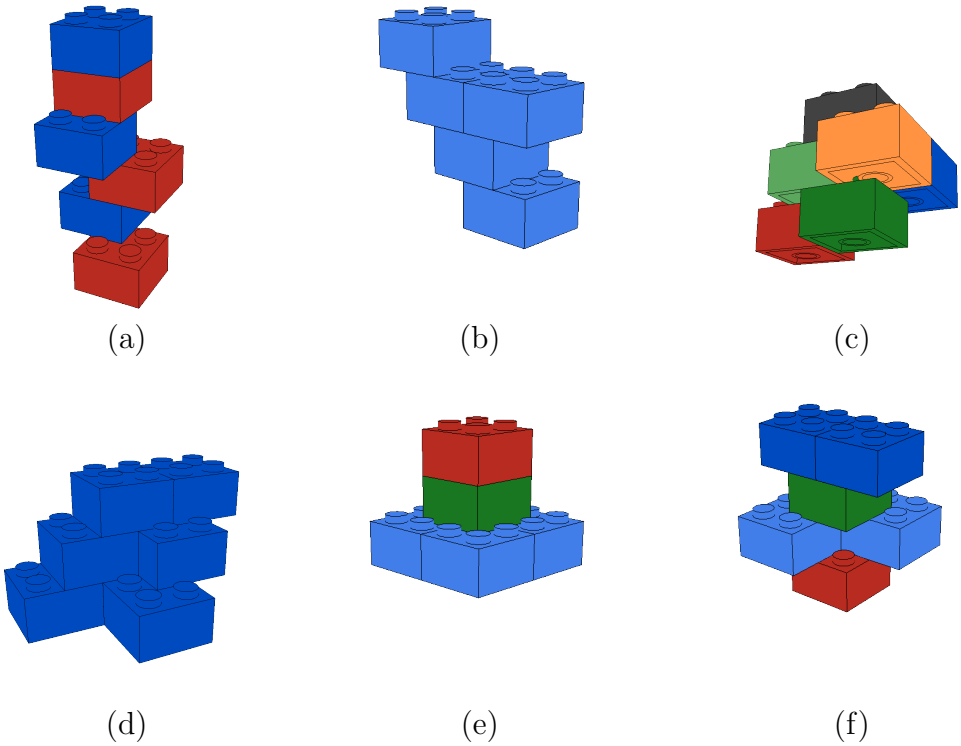


Figure 1: Examples of LEGO structures.

problems associated to tree structures, see e.g. [5]. Indeed, many of our – largely failed – efforts to describe the asymptotics better theoretically have been centered around building models based on trees for the studied LEGO structures. It remains our hope that this can be obtained with further work, and it has been a recurrent theme in our experimental efforts to attempt to shed light on similarities and differences in the asymptotic behavior of counting problems associated to LEGOs and trees, respectively.

1.3 Notation and conventions

Notation 1.1 *Whenever we are describing the dimensions of a LEGO block by "$b \times w$" it is tacitly assumed that $b \leq w$. Square blocks are denoted by "$w \times w$".*

As mentioned above, our main emphasis will be on the number $T_{b\times w}(n)$

defined as the count of all contiguous structures made of n LEGO blocks of size $b \times w$, identified up to rotation and translation. The set of all these equivalence classes is denoted $\mathcal{T}_{b\times w}[n]$.

We are also going to work with the (usually much smaller) numbers $T^{180}_{b\times w}(n)$ and $T^{90}_{b\times w}(n)$ consisting of those configurations which are symmetric after some rotation in the XY-plane by 180 or 90 degrees, respectively. For instance, the structures in Figure 1(e)(f) possess such symmetry properties. In situations where we wish to refer to all these three numbers at the same time we write $T^{\bullet}_{b\times w}(n)$.

The numbers $a_{b\times w}(n)$ are defined as all contiguous structures with n blocks, containing the *base block* $\mathsf{B}_{b\times w} = [0;b]\times[0;w]\times[0;1]$ with the further property that there is no other block in $\mathbb{R}^2 \times [0;1]$ and no block at all in $\mathbb{R}^2 \times [-1;0]$. Thus, the configuration can be thought of as sitting on the base block at a fixed position. No identification is performed in this case. As above, numbers $a^{180}_{b\times w}(n)$ and $a^{90}_{b\times w}(n)$ may be defined.

We say that a structure is *flat* when it is built out of $b \times w$-blocks which are all placed parallelly in the sector $[0;b]\times\mathbb{R}\times\mathbb{R}$ (as in Figure 1(b)). The dimension b is then, of course, irrelevant. Such structures are related to the work in [2] and may be somewhat easier to enumerate than the ones which are our primary interest here. The corresponding numbers grow relatively slowly, but most of the complications dealt with in this paper occur already for flat structures, and they shall be useful for us as illustrations. Numbers $\widehat{T}_w$ and $\widehat{a}_w$ are defined analogously to the non-flat case.

The number

$$k_{b\times w} = (2b-1)(2w-1) + (1-\delta_{b,w})(b+w-1)^2$$

counts the number of ways one block of a given dimension may be added to another. Similarly,

$$\widehat{k}_w = 2w-1.$$

1.4 Acknowledgment

We are grateful to colleagues at the Department of Matematical Sciences at the University of Copenhagen for fruitful discussions, notably Bergfinnur Durhuus, Ernst Hansen and Niels Richard Hansen, as well as the entire computer department lead by Rasmus Borup Hansen for their help in keeping our rather demanding processes alive. We also wish to extend our thanks to

w	2	3	4	5	6	7	8
$T_{1\times w}$	12	10	8	8	7	6	6
$T^{180}_{1\times w}$	13	11	9	9	8	8	7
$\widehat{T}_{1\times w}$	19	14	12	10	9	9	9
$\widehat{T}^{180}_{1\times w}$	20	15	13	11	11	10	10
$T_{2\times w}$	11	8	8	-	-	-	-
$T^{180}_{2\times w}$	12	9	9	-	-	-	-
$T_{w\times w}$	11	9	7	7	-	-	-
$T^{180}_{w\times w}$	12	10	9	8	-	-	-
$T^{90}_{w\times w}$	16	13	9	9	-	-	-

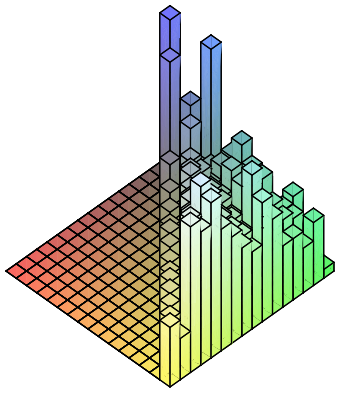


Figure 2: Left: number of exact terms computed and submitted to OEIS. Right: computing times for our Monte Carlo method as a function of the block size.

all the people involved in the LDraw project which we have benefited from in an obvious way.

2 Results and conjectures

2.1 Computing or estimating $T_{b\times w}$

The number of exact terms available to us by the methods described in Section 3 are listed in Figure 2. The computing times for numbers $T_{b\times w}(n)$ are roughly proportional to the values found. By contrast, the time consumption for applying the Monte Carlo methods of Section 4 varies in a non-obvious way with the dimensions as indicated on Figure 2. The figure illustrates the time needed to compute a 95% confidence interval for $T_{b\times w}(20)$ of a lenght which is less than 1% of the estimated value. The value for dimension 2×2 has been truncated as it is much larger than the others. As indicated, it turns out that dimensions 1×2, 1×14, 4×5 and 7×7 are especially prone to such an analysis. We have no full explanation for this phenomenon but will employ this observation in Section 2.2 below.

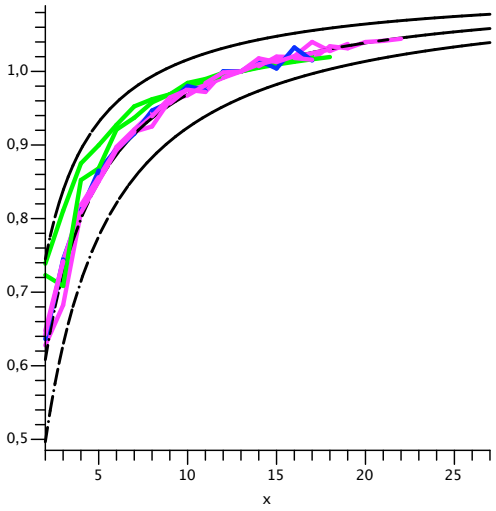


Figure 3: Growth regimes associated to various block sizes (green: flat, magenta: non-square, blue: square). Dotted lines (from below) correspond to subexponential terms of the order of $n^{-1}, n^{-3/2}$ and n^{-2}, respectively.

2.2 Growth of $T_{b\times w}(n)$

As mentioned in the introduction, our theoretical knowledge concerning the growth of $T_{b\times w}(n)$ is very limited, and, needless to say, an experimental approach to such a problem can only lead to circumstantial evidence. Furthermore, the experiments are rather sensitive towards imprecisions in the data. Consider, however, Figure 3 which has been obtained by plotting $T_{b\times w}(n+1)/T_{b\times w}(n)\cdot(T_{b\times w}(14)/T_{b\times w}(13))^{-1}$ for $b\times w\in\{1\times 2, 1\times 14, 4\times 5, 7\times 7\}$ as well as for flat structures with $w\in\{2,14\}$. The values for $T_{b\times w}(n)$ have been computed by our Monte Carlo method with confidence intervals only 1% of the observed value. The plots are consistent with a growth regime which is roughly $H^{n-1}\cdot n^{-3/2}$. Somewhat more carefully, we conjecture:

Conjecture 2.1 $T_{b\times w}(n)\sim CH^{n-1}n^P$ for suitably chosen $C=c_{b\times w}$, $H=h_{b\times w}$ and $P=p_{b\times w}$, and

$$\lim_{n\to\infty}\frac{T_{b\times w}(n+1)}{T_{b\times w}(n)}=h_{b\times w}.$$

2.3 Entropies and their variation

Based on Conjecture 2.1 and extensive experimentation we have found estimates for $c_{b\times w}, h_{b\times w}$ and $p_{b\times w}$ as indicated on Figure 4. These estimates have been achieved by a least squares appoximation of the form

$$C + H(n-1) + P\log(n)$$

to a semilogarithmic plot of approximated values of $T_{b\times w}(5), \ldots T_{b\times w}(20)$, generated by our Monte Carlo methods with a confidence interval at most 10% of the size of the estimated value. The found values for $h_{b\times w}$ are also tabulated in Figure 5.

We have analyzed these estimates as indicated in Figures 6 and 7. Figure 6 shows how well the observed entropies may be approximated by a quadratic fit. The case of square blocks requires special attention, but only to the effect that the fitting quadratic expression approximate twice the observed entropies. Similarly, the observed value for the constant term C in the square case is half of the observed value in the non-square case.

Figure 7 shows relevant contours of this quadratic approximation and how a linear fit is possible in the case of flat structures.

Conjecture 2.2 There exist functions

$$p(w,b) = Aw^2 + Bbw + Cb^2 + Dw + Eb + F$$
$$\widehat{p}(w) = \widehat{D}w + \widehat{F}$$

which approximate entropies to a high level of accuracy as follows:

$$p(w,b) \sim h_{b\times w} \qquad p(w,w) \sim 2h_{w\times w} \qquad \widehat{p}(w) \sim \widehat{h}_w.$$

We estimate

$$p(w,b) = 13.68bw + 2.92b^2 + 3.29w^2 - 12.62b - 8.89w + 4.23$$

and

$$\widehat{p}(w) = 3.57w - 1.88.$$

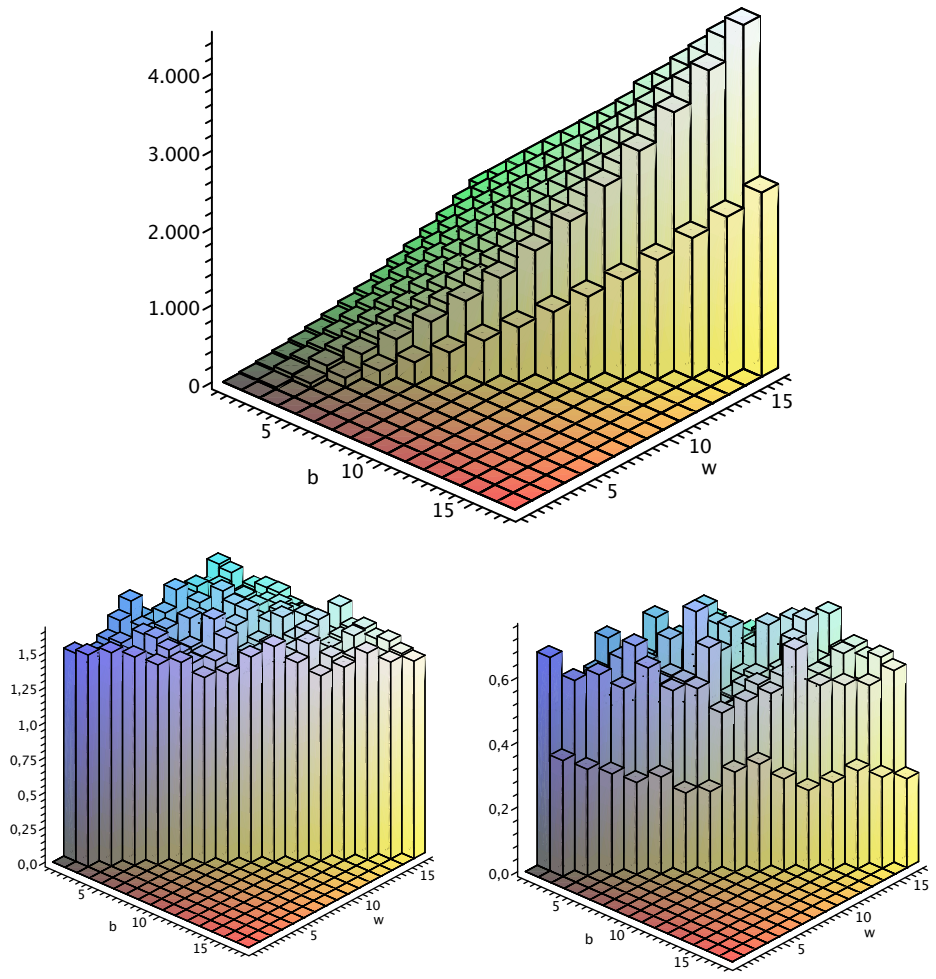


Figure 4: Estimated values for $h_{b\times w}$ (top), $-p_{b\times w}$ (bottom left) and $c_{b\times w}$ (bottom right).

	1	2	3	4	5	6	7	8	9	10
1	1	16.6	36.9	64.1	98.4	140	189	240	298	365
2		21.5	76.4	117	165	218	282	346	427	514
3			61.7	176	233	307	377	462	544	643
4				122	314	399	483	574	687	793
5					201	497	594	702	808	936
6						299	710	829	976	1082
7							415	969	1120	1248
8								555	1253	1412
9									718	1597
10										905

	11	12	13	14	15	16
1	439	514	606	695	803	898
2	598	681	783	903	1022	1124
3	753	857	976	1089	1226	1361
4	903	1043	1161	1297	1431	1593
5	1066	1217	1344	1497	1657	1825
6	1245	1402	1543	1695	1877	2045
7	1405	1564	1750	1930	2101	2285
8	1587	1762	1970	2125	2305	2507
9	1781	1953	2171	2393	2558	2758
10	1966	2181	2407	2629	2811	3094
11	1093	2410	2588	2839	3086	3327
12		1300	2855	3069	3359	3619
13			1543	3346	3604	3884
14				1819	3877	4185
15					2083	4455
16						2385

	Flat
1	1
2	5.20
3	8.84
4	12.5
5	16.2
6	19.5
7	23.1
8	26.7
9	30.1
10	34.0
11	37.0
12	41.1
13	44.5
14	47.8
15	51.4
16	54.9
17	58.6
18	63.1
19	65.9
20	69.5

Figure 5: Estimates for $h_{b\times w}$ and $\widehat{h}_w$.

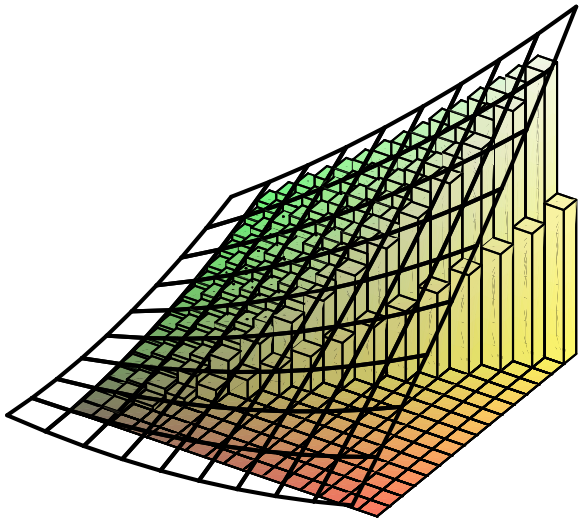


Figure 6: Entropies with quadratic fit.

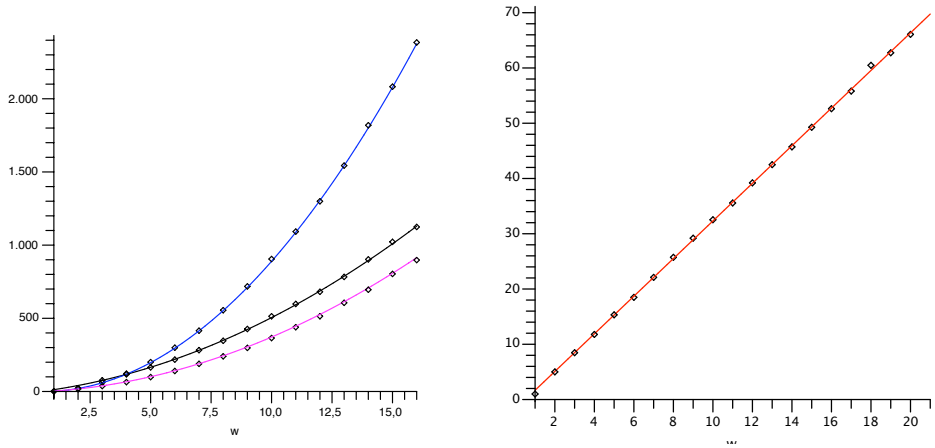


Figure 7: Left: Fits specialized to dimensions $1 \times w$ [magenta], $2 \times w$ [black], and $w \times w$ [blue]. Right: Flat entropies with linear fit.

2.4 A theoretical interlude

In this section we present and prove our best estimates for the entropies of general block sizes:

Theorem 2.3 *We have, for* $1 \leq b < w$,

$$2w - 1 = \widehat{k}_w \leq \widehat{h}_w \leq 7w$$
$$w^2 + 2w - 1 = k_{1\times w} \leq h_{1\times w} \leq 7w^2 + 7w - 14$$
$$w^2 + b^2 + 6bw - 4b - 4w + 2 = k_{b\times w} \leq h_{b\times w} \leq 24w^2 + 36bw - 48w$$
$$4w^2 - 4w + 1 = k_{w\times w} \leq h_{w\times w} \leq 18w^2.$$

We do this for several reasons. Firstly, we feel that having upper and lower bounds of the same order backs up Conjecture 2.2. Secondly, the approach forms the basis of our Monte Carlo method described in Section 4 below, and at least partially explains the phenomenon noted above (cf. Figure 2) that some block sizes are more prone to analysis than others (see Remark 2.8).

The result is proved following a strategy developed in [1] in order to prove that $T_{b\times w}(n)$ does not grow faster than exponentially, i.e. that always $h_{b\times w} < \infty$, and to give the estimates on $h_{2\times 4}$ mentioned above. To prove the theorem we recall a few concepts from [1]:

The set of all ways to attach one block of a fixed dimension on top of another is denoted the *set of positions*. A *partition* of this set is a subdivision into $b \times w$ sets $\mathcal{P}_1, \ldots, \mathcal{P}_{bw}$, one for each stud of the block, with the extra property that all positions in $\mathcal{P}_i$ employ stud i. We found in [1] that a partition of the 46 positions on a 2×4 block into a partition with

$$46 = 16 + 15 + 7 + 5 + 2 + 1 + 0 + 0$$

elements gave optimal estimates. However, in order to give estimates valid for any block size, and to simplify our Monte Carlo method described below we shall focus entirely on *even* subdivisions such as

$$46 = 8 + 8 + 8 + 8 + 8 + 6 + 0 + 0$$

which minimize the following quantities:

Definition 2.4 The *term count* T of a partition is the number of non-empty sets. The *element count* E is the maximal number of elements in a set in the partition.

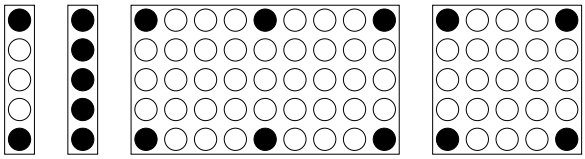


Figure 8: Choices of studs with non-empty partitions, case (i)–(iv).

More precisely, we aim to first minimize T and then E for the minimal T.

Definition 2.5 $f(x) = (1 - 1/x)^{-x}$.

Lemma 2.6 *There exist partitions with term and element counts as indicated below*

Case	*Dimension*	T	E
(i)	$1 \times w$, *flat*	2	w
(ii)	$1 \times w$	w	$w + 2$
(iii)	$b \times w, 1 < b < w$	$4 + 2\left\lfloor \frac{w-2}{b} \right\rfloor$	$2bw$
(iv)	$w \times w$	4	w^2

Proof: (i): Since each position employs at least one of the end studs, we can arrange $\mathsf{T} = 2$ as indicated on Figure 8. And each stud supports precisely w positions.

(ii): Since all positions with the blocks in orthogonal positions apply only one stud we need to take $\mathsf{T} = w$. There are w such positions for each stud. The remaining $2w - 1$ positions can be distributed with at most 2 block per stud.

(iii): Each position must meet one of the long sides of the block. We are forced to use the corners of the blocks, but each time a position has been selected to support a non-empty set of the partition we do not have to place any positions on the neighboring $b - 1$ studs in each direction of the long side. Distributing evenly as in Figure 8 we arrive at a partition with T as indicated. As in case (i), E must be chosen as the total number of positions supported by a stud.

(iv): We need only employ the corners as in Figure 8 and may then argue as in case (iii). □

One can prove that the values of T are optimal in all these cases.

For a fixed partition with term and element counts T and E, respectively, we define $\mathcal{U}_{b\times w}[n]$ as the collection of $(2\mathsf{T} + (n-2)(2\mathsf{T}-1))$-tuples with entries in $\{0, 1, \ldots, \mathsf{E}\}$ such that precisely $n-1$ of the entries are nonzero. As in [1] a surjective map

$$\Psi : \mathcal{U}_{b\times w}[n] \to \mathcal{T}_{b\times w}[n] \cup \{\mathsf{FAIL}\}$$

may be defined by reading off, in a systematic way, from the nonzero entries of the tuple how to attach blocks to reach a structure. The first $2\mathsf{T}$ entries specify what to attach to the base block $\mathsf{B}_{b\times w}$ by reading each nonzero integer as an instruction to place a block in one of the at most E positions on one of the T selected studs (or corresponding holes) on either side of the block. The added blocks will be enumerated $1, 2, \ldots,$ and the next $2\mathsf{T}-1$ entries will be used to specify what to attach to block 2. We may use one entry less by employing in some systematic way that one position was already used when adding block i for $i > 0$.

The procedure is carefully explained in [1], but to prepare for our discussion in Section 4 we revise the procedure to return with FAIL precisely under the following conditions:

- If at any point a block collides with one which has already been placed.
- If, after reading the specifications for the first $m < n-1$ blocks, no block $m+1$ has been introduced.
- If an entry $\mathsf{T} \geq i > 0$ is read at a stud supporting less than i positions.

Lemma 2.7 *If a partition of the $b \times w$ block exists with term and element counts T and E, then*

$$h_{b\times w} \leq \mathsf{E}(2\mathsf{T}-2)f(2\mathsf{T}-1)$$

Proof: Clearly

$$T_{b\times w}(n) \leq \#\mathcal{U}_{b\times w}[n] = \begin{pmatrix} 2\mathsf{T} + (n-2)(2\mathsf{T}-1) \\ n-1 \end{pmatrix} \mathsf{E}^{n-1}$$

and as in the proof of [1, Theorem 3.1] we get with Stirling's formula that

$$h_{b\times w} \leq \frac{\mathsf{E}(2\mathsf{T}-1)^{2\mathsf{T}-1}}{(2\mathsf{T}-2)^{2\mathsf{T}-2}}$$

and hence

$$\begin{aligned} h_{b\times w} &\leq \mathsf{E}(2\mathsf{T}-2)\left(\frac{2\mathsf{T}-1}{(2\mathsf{T}-1)-1}\right)^{2\mathsf{T}-1} \\ &= \mathsf{E}(2\mathsf{T}-2)f(2\mathsf{T}-1) \end{aligned}$$

since f is decreasing. □

Proof of Theorem 2.3: The four upward estimates correspond to the cases (i)–(iv) in Lemma 2.6. In the first two cases, we further use that $f(3) \leq 7/2$ and in the latter two cases, we use that $f(4) \leq 3$. □

Remark 2.8 The values which are especially prone to Monte Carlo analysis may be characterized at values where T and E are small compared to the found values of $h_{b\times w}$.

2.5 Average heights

The height of a LEGO structure with n blocks may vary between 2 and n except of course in the case 1×1. Using Monte Carlo methods we have estimated the average height of a structure with n blocks of a fixed size, expecting to find that this would universally grow as $\sqrt{n}$ as is the case for trees.

This, however, turns out not to be the case. We have

Conjecture 2.9 The average height of elements in $\mathcal{T}_{b\times w}[n]$ grow as a power n^α with α in the range $[0.6; 0.7]$

The exponent α varies quite a lot with the block size as indicated on Figure 9. It is interesting to note that although LEGO's features allows one to place blocks from below, thus not contributing to the height, the restrictions on the options for building upwards forces higher averages than for $(1, \ldots m)$-trees, where m is the maximal number of blocks which may be placed on top of another.

This phenomenon is more distinctive for small blocks, corresponding to the fact that there in these cases are very few different ways to construct low buldings. For instance, there are at most 2 flat structures of height two with 1×2-blocks, irrespective of the number of blocks involved.

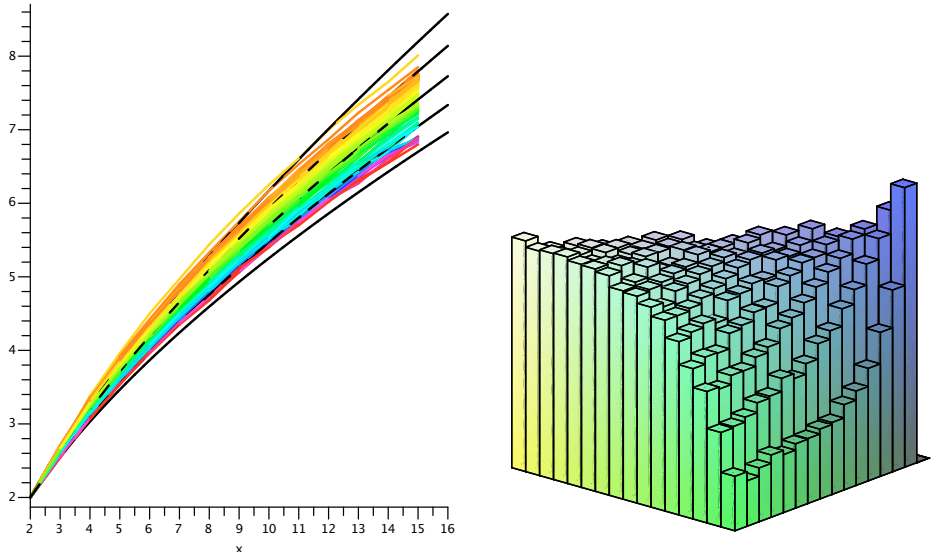


Figure 9: Left: growth of average heights color coded by assigning colors low in the spectrum to blocks with a small value w/b. The black lines follow growths between $n^{0.6}$ and $n^{0.7}$, equally spaced. Right: Average heights of structures with 13–15 blocks by block size.

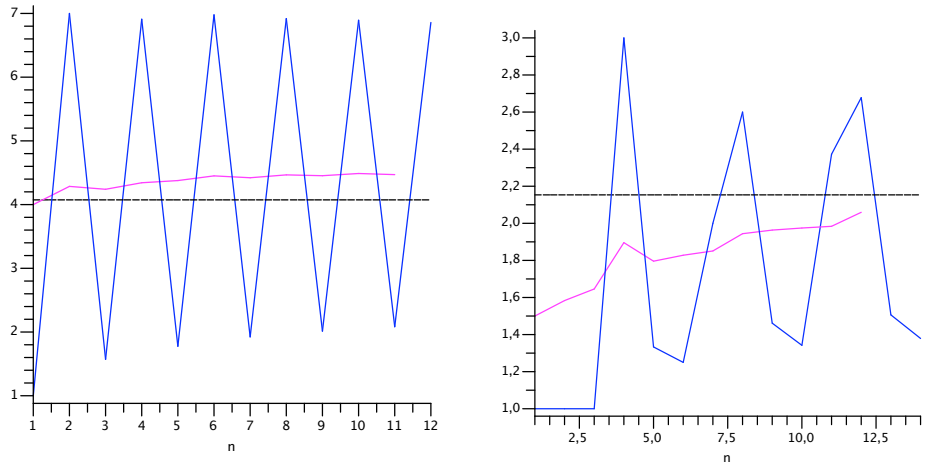


Figure 10: Left in blue: $T^{180}_{1\times 2}(n+1)/T^{180}_{1\times 2}(n)$. Right in blue: $T^{90}_{2\times 2}(n+1)/T^{90}_{2\times 2}(n)$. Magenta: Running averages as in Conjecture 2.10(i) and (ii), respectively. Dotted lines: Our estimates of $\sqrt{h_{1\times 2}}$ and $\sqrt[4]{h_{2\times 2}}$, respectively.

2.6 Symmetric structures

The number of symmetric structures grows less regularly than the total number. As an extreme case, consider the number $T^{90}_{b\times w}(n)$ $(b < w)$ which is zero unless $n = 4m$, but for which $T^{90}_{b\times w}(4m)$ tends to infinity. When the blocks are square, or when we consider $T^{180}_{b\times w}(n)$ instead, the numbers will never drop to zero, but the observations demonstrate periodic phenomena as indicated on Figure 10.

We hence do not expect an equivalent of Conjecture 2.2 to hold even in these cases. In fact, there is rather strong evidence that

$$\lim_{m\to\infty} \frac{T^{180}_{b\times w}(2m+1)}{T^{180}_{b\times w}(2m)} \neq \lim_{m\to\infty} \frac{T^{180}_{b\times w}(2m+2)}{T^{180}_{b\times w}(2m+1)}$$

for any dimension $b \times w$ except 1×1.

Note that, as was the case above, we do not know that these limits exist. In fact, as a strong irritant for us, we can not even prove that the entropy limits exist in these cases. We offer the following

Conjecture 2.10

(i) The numbers

$$\exp\left(\lim_{n\to\infty}\frac{\log T^{180}_{b\times w}(n)}{n}\right) \qquad \frac{1}{2}\sum_{i=0}^{1}\left[\lim_{m\to\infty}\frac{T^{180}_{b\times w}(2m+i+1)}{T^{180}_{b\times w}(2m+i)}\right]$$

are defined and coincide.

(ii) The numbers

$$\exp\left(\lim_{n\to\infty}\frac{\log T^{90}_{w\times w}(n)}{n}\right) \qquad \frac{1}{4}\sum_{i=0}^{3}\left[\lim_{m\to\infty}\frac{T^{90}_{w\times w}(4m+i+1)}{T^{90}_{w\times w}(4m+i)}\right]$$

are defined and coincide.

(iii) When $b < w$, the numbers

$$\exp\left(\lim_{n\to\infty}\frac{\log T^{90}_{b\times w}(4n)}{4n}\right) \qquad \sqrt[4]{\lim_{m\to\infty}\frac{T^{90}_{b\times w}(4m+4)}{T^{90}_{b\times w}(4m)}}$$

are defined and coincide.

The evidence for this conjecture is somewhat impaired by the fact that we have found no efficient Monte Carlo method for estimating the number of symmetric structures the way we are able to estimate general ones. Fortunately, this is somewhat countered by the fact that we can count symmetric structures faster using the methods of Section 3.2.3 below.

It is tempting to conjecture that the symmetric entropies, if they exist, coincide with the values $\sqrt{h_{b\times w}}$, $\sqrt[4]{h_{w\times w}}$ and $\sqrt[4]{h_{b\times w}}$ in the cases (i), (ii) and (iii) above, corresponding to the claim that the averages in Figure 10 coincide with the dotted lines in the limit, but the evidence for this is too weak at the moment.

3 Exact counts

3.1 The basic algorithm

Our method is based on counting all contiguous configurations which contain the base block $\mathsf{B}_{b\times w}$ at its lowest level, keeping track of how many are

symmetric and how many blocks the found configurations have in their lower level.

We denote the number of such configurations with n blocks of which i is in their lower level by $c_{b\times w}(n,i)$. The number of these which are symmetric after a rotation by 180° we denote by $c^{180}_{b\times w}(n,i)$. And the number of these which are invariant under a rotation by 90° we denote by $c^{90}_{b\times w}(n,i)$.

Note that $a_{b\times w}(n) = c_{b\times w}(n,1)$. We further have:

Proposition 3.1 *With $\delta_{b,w}$ defined as Kronecker delta we have*

$$\begin{aligned}
T_{b\times w}(n) &= 2^{-1-\delta_{b,w}} \sum_{i=1}^{n} \frac{c_{b\times w}(n,i) + c^{180}_{b\times w}(n,i) + 2c^{90}_{b\times w}(n,i)}{i} \\
T^{180}_{b\times w}(n) &= 2^{-\delta_{b,w}} \sum_{i=1}^{n} \frac{c^{180}_{b\times w}(n,i) + c^{90}_{b\times w}(n,i)}{i} \\
T^{90}_{b\times w}(n) &= 2^{1-\delta_{b,w}} \sum_{i=1}^{n} \frac{c^{90}_{b\times w}(n,i)}{i}
\end{aligned}$$

Proof: Each element in $\mathcal{T}_{b\times w}[n]$ is an equivalence class which we may assume is represented by a configuration C containing the base block $\mathsf{B}_{b\times w}$ at its lowest level. We need to compute the number of such structures representing the same element as C, and relate this count to the numbers $c^{\bullet}_{b\times w}(n,i)$.

Let m be the number of blocks in the lower level of C. First assume that $b \neq w$. If C is not symmetric after a rotation by 180 degrees we can obtain $2m$ different configurations by rotating and translating C, letting each block in the lower layer meet the base block $\mathsf{B}_{b\times w}$ in 2 different ways. This gives the equation

$$T_{b\times w}(n) - T^{180}_{b\times w}(n) = \sum_{i=1}^{n} \frac{c_{b\times w}(n,i) - c^{180}_{b\times w}(n,i)}{2i}. \tag{1}$$

Assume that C is symmetric after a rotation by 180 degrees but not after 90 degrees. If we have a block in the center of the lower layer, we have excactly one configuration in which this block meets $\mathsf{B}_{b\times w}$. We can subdivide all the other blocks into pairs invariant under rotation by 180 degrees about the center. Each block in each pair will meet $\mathsf{B}_{b\times w}$ in exactly one configuration.

This gives

$$T^{180}_{b\times w}(n) - T^{90}_{b\times w}(n) = \sum_{i=1}^{n} \frac{c^{180}_{b\times w}(n,i) - c^{90}_{b\times w}(n,i)}{i}. \tag{2}$$

Finally, if C is symmetric by 90 degrees, we may subdivide the m lower blocks into sets of 4 blocks each, and we can find exactly two configurations where a block in each of these sets meets B. Thus we have

$$T^{90}_{b\times w}(n) = 2\sum_{i=1}^{n} \frac{c^{90}_{b\times w}(n,i)}{i}. \tag{3}$$

Then the total number of structures with n blocks is given by adding (1), (2) and (3), and the number of structures symmetric after rotation by 180 degrees is the sum of (2) and (3).

If $b = w$ we can find twice as many configurations in which one of the lower blocks meets $\mathsf{B}_{b\times w}$, because each of the above described configurations may be rotated 90 degrees to create a new one. Then we have to divide by two in the sums in (1), (2), and (3). □

It is straightforward to write a recursive computer program which generates all the configurations containing the base block at the bottommost level, by simply going through all possibilities for adding new blocks to the configuration at hand at each stage of the recursive process.

However, making sure that each such configuration is generated exacly once (so that $c_{b\times w}(n,i)$ can be computed for each $i \leq n$) requires a little more thought. Indeed, as indicated in Figure 11, a generic configuration can be made by adding blocks to a base block in several different orders — as we shall see and use in Section 4 the number of such orders can be computed using Kirchhoff's theorem ([3]) — so our algorithm must devise a unique way to produce each configuration.

Our algorithm LegoCount (cf. Algorithm 1) achieves this by two different means. First, we fix an order of the $2k_{b\times w}$ positions in which one $b\times w$ block may be attached to another. The positions underneath a given block will be numbered as $0,\ldots,k_{b\times w}-1$ and the positions above as $k_{b\times w},\ldots,2k_{b\times w}-1$. The remaining choices of the used order is going to be irrelevant, but in examples with flat counts of 1×2-blocks we use the ordering  ,  ,  . Second, we keep an index of the order in which new blocks have been added to the configuration and make sure that each new block is thought of as being

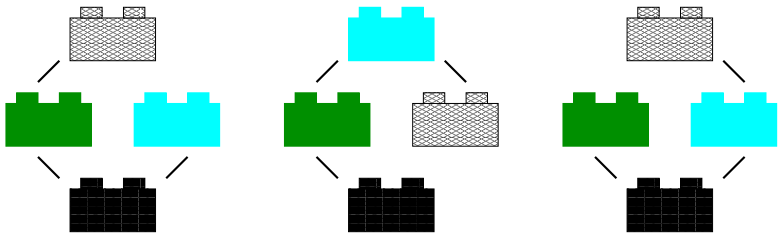


Figure 11: Same configuration.

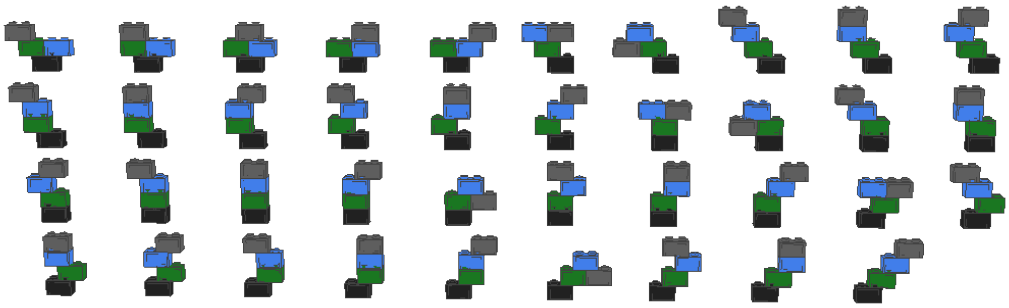


Figure 12: $\widehat{a}_{1\times 2}(4) = 39$.

placed on the previously existing block with the lowest possible index. More precisely, the algorithm disallows for $i_1 > i_2$ the placement of a new block on block i_1 if the new block could also have been placed on block i_2.

Here OkToAdd will check that a block b can be added to L by attaching it to a block at index i or higher, such that no collision occurs and such that it could not have been added to a block at index less than i. We claim that

$$(c_{b\times w}(n,1),\ldots,c_{b\times w}(n,n)) = \textsc{LegoCount}(n-1,[\mathsf{B}_{b\times w}],0,0,0) \quad (4)$$

$$(a_{b\times w}(n),0,\ldots,0) = \textsc{LegoCount}(n-1,[\mathsf{B}_{b\times w}],0,0,1) \quad (5)$$

can be checked by induction, but rather than give a formal proof of these facts we find it more instructive to give an example of how the algorithm works. For simplicity, let us compute $\widehat{a}_{1\times 2}(4) = 39$ using the algorithm. Figure 12 shows in what order the configurations are computed, with the blocks introduced in the order black, green, blue, grey.

One should pay special attention to the way that the algorithm avoids counting the two configurations to the right in Figure 11 and chooses the

Algorithm 1 LegoCount(r, L, i_0, j_0, ℓ_0)

Require: r: number of blocks to be placed
Require: L: configuration of LEGO blocks
Require: i_0: index of first buildable block
Require: j_0: index of first usable position of block i_0
Require: ℓ_0: lowest buildable level
Ensure: $\underline{c}$: the count of configurations buildable with r blocks on L as specified by i_0, j_0 and ℓ_0.

```
1: if r = 0 then
2:    m ← the number of blocks at level 0
3:    return e_m
4: else
5:    c ← 0
6:    for all i ← i_0, ..., |L| do
7:       if L[i] is in a level ≤ ℓ_0 then
8:          j_0 ← max(j_0, k_{b×w})
9:       for all j ← j_0, ..., 2k_{b×w} − 1 do
10:         b ← the block at position j relative to L[i]
11:         if OkToAdd(L, b, j) then
12:            c ← c + LegoCount(r − 1, L + [b], i, j + 1, ℓ_0)
13:      j_0 ← 0
14:   return c
```

leftmost one, which is identical to the third configuration in Figure 12. The middle configuration is avoided by the fact that in the body of the call LegoCount$(1, $$, 1, 6, 1)$ it is no longer possible to place a block on the base block. And similarly, in the call LegoCount$(1, $  $, 0, 6, 1)$, OkToAdd will prevent the placement of a block on block 2 which could have been placed on block 1.

3.2 Memory-neutral improvements

In the following two sections we shall develop methods to reduce computing time for the counting problems studied. Most of these methods are implemented using a tree pruning strategy based on placing conditions regarding the *levels* of the structure. We introduce a new parameter

$$\textsc{LegoCount}(r, L, i_0, j_0, \ell, \mathcal{C})$$

into our fundamental counting algorithm, where $\mathcal{C}$ is of the form

$$(l_1 \vee \cdots \vee l_{k_1}) \wedge (l_{k_1+1} \vee \cdots \vee l_{k_2}) \wedge \cdots \wedge (l_{k_{m-1}+1} \vee \cdots \vee l_{k_m})$$

specifying a condition on what levels must be used when placing the remaining r blocks on L to the effect that at least one block must be placed in one of the levels $l_{k_{i-1}+1}, \ldots, l_{k_i}$, for each i.

We will not detail the implementation of this, but quickly note that this pruning strategy may be very efficient in some cases. Consider the configuration L given by the black and red blocks of Figure 13 with the condition $0 \vee 1$. The way our algorithm works, only the red block may be used to place ensuing blocks, and hence at least 5 blocks is required to reach the indicated levels.

When we are interested only in counting certain subclasses of objects, like those that are invariant under a rotation of 90°, we will use the notation LegoCount[90] etc.

3.2.1 Bottlenecks

Following an idea in [1] we say that a configuration has a *bottleneck* at a certain level if there is exactly one block in that level. We say that the configuration is *fat* if it contains no bottleneck, i.e. if every level of it contains at least two blocks. Of the six structures in Figure 1, (d) is fat. Define

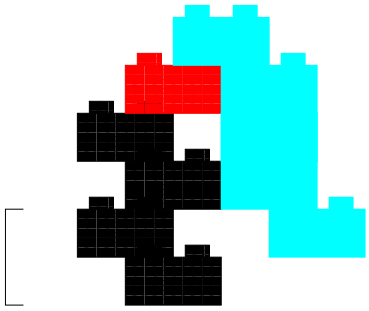


Figure 13: To reach level $[0, 1]$.

$\mathbf{cf}^{\bullet}_{b\times w}(n, i)$ as the number of configurations counted by $c^{\bullet}_{b\times w}(n, i)$ which are fat, and $\mathbf{af}^{\bullet}_{b\times w}(n)$ as the number of configurations counted by $a^{\bullet}_{b\times w}(n)$ which have more than one block at each level except the lower one (note that this is *not* fat).

We then have

Proposition 3.2 *The results in Proposition 3.1 hold true with* $c^{\bullet}_{b\times w}(n, i)$ *replaced by* $\widetilde{c}^{\bullet}_{b\times w}(n, i)$*, where*

$$\widetilde{c}^{\bullet}_{b\times w}(n, 1) = \sum_{j=1}^{n-1} a^{\bullet}_{b\times w}(j)\mathbf{af}^{\bullet}_{b\times w}(n - j + 1)$$

and

$$\widetilde{c}^{\bullet}_{b\times w}(n, j) = \mathbf{cf}^{\bullet}_{b\times w}(n, j), j \geq 2.$$

Proof: One sees by translating the configuration so that the lowest bottleneck, if any, is placed at level 0, that the argument of Proposition 3.1 holds true using a subdivision of $\widetilde{c}^{\bullet}_{b\times w}(n, i)$ with i denoting the number of blocks in the lowest level only in the fat case and 1 in the bottleneck case. The index j in the expression for $\widetilde{c}^{\bullet}_{b\times w}(n, 1)$ correponds to the highest level at which a bottleneck can be found.

□

Implementation notes 3.3 *The strategy reduces the computation af structures with* n *blocks to general counts of up to* $n-1$ *blocks placed on the base block and the computation of fat structures. Computing fat structures may*

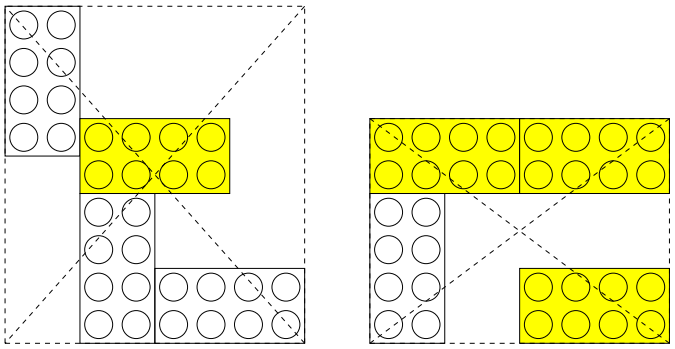


Figure 14: One and three centered blocks.

be done in reduced time by the obvious observation that if the levels $l_1, \ldots l_k$ *of* L *have only one block, then*

$$\textsc{LegoCount}[\textsc{fat}](r, L, i_0, j_0, \ell, \mathcal{C}) =$$
$$\textsc{LegoCount}[\textsc{fat}](r, L, i_0, j_0, \ell, \mathcal{C} \wedge l_1 \wedge \cdots \wedge l_k)$$

3.2.2 Bottom layer count

Our computing strategy has the inherent inefficiency that a generic configuration counted to get $c^{\bullet}_{b\times w}(n, j)$ must be recounted j times, and this problem grows in prominence with the strategy of Section 3.2.1 above, which focuses counting to structures with at least 2 blocks in the lower level. For large j, this binds resources, but the alternative strategy of choosing a way to just construct each structure once has a lot of overhead which renders it infeasible.

Thus we have to live with this problem, but it may be reduced by the following strategy. We define the *center* of a level of blocks as the center of the smallest rectangle containing all blocks. A block is then called *centered* if it has smaller distance to the center than any other block in the level, cf. Figure 14.

We introduce the number $\mathbf{cfc}^{\bullet}_{b\times w}(n, j)$ to denote the number of fat structures containing $\mathsf{B}_{b\times w}$ at the lowest level, with the properties that

- $\mathsf{B}_{b\times w}$ is centered in the lowest level
- There are j centered blocks in the lowest level.

With this we get

Proposition 3.4 *The results in Proposition 3.1 hold true with* $c^{\bullet}_{b\times w}(n,i)$ *replaced by* $\widetilde{c}^{\bullet}_{b\times w}(n,i)$*, where*

$$\widetilde{c}^{\bullet}_{b\times w}(n,1) = \sum_{j=1}^{n-1} a^{\bullet}_{b\times w}(i)\mathbf{af}^{\bullet}_{b\times w}(n-i+1) + \mathbf{cfc}^{\bullet}_{b\times w}(n,1)$$

and

$$\widetilde{c}^{\bullet}_{b\times w}(n,j) = \mathbf{cfc}^{\bullet}_{b\times w}(n,j), j \geq 2.$$

Proof: Proceed as in Proposition 3.1 and 3.4, but subdivide only after the number of centered blocks. □

Implementation notes 3.5 *With this strategy, if a configuration* L *does not have* $\mathsf{B}_{b\times w}$ *as a centered block, then the condition* 0 *may be imposed on further counting based on* L*.*

3.2.3 Counting symmetric configurations

It is straightforward to determine when a given configuration is symmetric after a rotation of 90° or 180° in order to compute $c^{90}_{b\times w}(n,i)$ and $c^{180}_{b\times w}(n,i)$, respectively. To reduce computing time for doing so we keep track of the center (cf. Section 3.2.2 above) of each layer in the given configuration. Only if all centers agree we need to perform the somewhat demanding task of actually rotating the configuration to check for symmetry.

If we are only interested in counting symmetric configurations we can quite easily compute what levels are symmetric in their own right, and add conditions if they are not. But to compute numbers $T^{90}_{b\times w}(n)$ and $T^{180}_{b\times w}(n)$ for n beyond the levels to which we can compute $T_{b\times w}(n)$ we need a more refined approach. For any given configuration we compute the minimal number of blocks to add to it to produce a symmetric configuration. If this number is larger than the remaining number of blocks we can prune the search tree accordingly.

For example, an algorithm ToComplete90 can be defined as indicated in Algorithm 2 to compute the minimal number of blocks to add to arrive at a configuration which is symmetric about a given point $\underline{c}$. It may very well be impossible to add such blocks, in which case the algorithm returns $+\infty$.

If $\text{ToComplete90}(L,\underline{c}) < 3|L|$ at least one pair B_1, B_2 of blocks in L must have the property that B_1 is taken to B_2 under a rotation of $90, 180,$

Algorithm 2 ToComplete90 $(L, \underline{c})$

Require: L: A configuration with all blocks unflagged
Require: $\underline{c}$: A vector in $\mathbb{R}^2$
Ensure: The minimal number of blocks to add to L to get a configuration which is symmetric after a rotation of 90° about $\underline{c}$.

```
1: k ← 0
2: for all b ∈ L do
3:    if b is not flagged then
4:       b' ← b
5:       for all i ∈ {0, 1, 2} do
6:          b' ← the block b' rotated 90° around c
7:          if b' ∈ L then
8:             Flag b'
9:          else
10:            if OkToAdd(L, b', i) then
11:               k ← k + 1
12:            else
13:               return ∞
14: return k
```

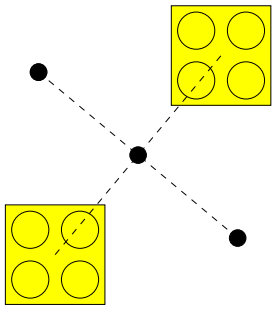


Figure 15: Points about which one block is rotated to cover another.

or 270 degrees about $\underline{c}$ (B_1 and B_2 could be equal). It is straightforward to compute the set C_0 of points with this property as indicated in Figure 15, and this allows us to prune the seach tree if the quantity to the right is larger than the remaining number of blocks:

Lemma 3.6

$$\min_{\underline{c}\in(\frac{1}{2}\mathbb{Z})^2} \textsc{ToComplete90}(L,\underline{c},i) \geq \min\left(3|L|, \min_{\underline{c}\in C_0} \textsc{ToComplete90}(L,\underline{c},i)\right).$$

3.3 Memory-demanding improvements

3.3.1 Symmetry

Since a configuration such as  can be gotten from the configuration  by rotation and translation it is natural to use symmetry arguments to avoid computing twice how many ways these configurations can be augmented to configurations with $n > 2$ blocks. However, one needs to take into account that since we have ordered the relative positions such that the configuration  comes before the configuration  , all the configurations which contain both  and  will be considered as belonging to  . For instance, one may inspect Figure 12 to see that

$$\textsc{LegoCount}(2,\,\text{  },0,4,1) = 16 > 11 = \textsc{LegoCount}(2,\,\text{  },0,6,1)$$

However, this is essentially the only problem in employing symmetry in this situation, and we can circumvent it by using the third parameter of LegoCount. Indeed we have

$$\textsc{LegoCount}(2,\,\text{  },1,0,1) = 11 = \textsc{LegoCount}(2,\,\text{  },1,0,1)$$

and more generally

Observation 3.7 If no block is placed on the blocks at index $i_0, \ldots, |L|$ of the configuration L then if L' is achieved from L by a rigid transformation of the XY-plane we have

$$\textsc{LegoCount}(n, L, i_0, 0, \ell, \mathcal{C}) = \textsc{LegoCount}(n, L', i_0, 0, \ell, \mathcal{C})$$

for all n and ℓ.

A problem is that Algorithm 1, as it stands, will never induce recursive calls of this form, but it is easy to revise the algoritm so that it does, and to store and reuse numbers in the situation where $i_0 = 1$ in Observation 3.7 above. Note also that we can use reflections in the observation above even though we do not identify structures up to reflection.

Implementation notes 3.8

(i) Notice that since the center of a level as described in Section 3.2.2 is invariant under rigid transformations the methods there can be combined with those of this section without problems.

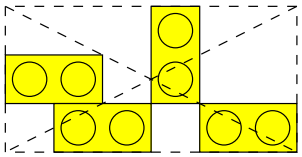
(ii) It would of course be possible to store numbers $\textsc{LegoCount}(n, L, i_0, 0, \ell_0)$ *for* $i_0 > 1$ *but they would only be reused in the relatively rare case where* L *were symmetric or self-reflective. Because it is difficult to implement this alongside the methods which shall be described in Section 3.3.2 below we have chosen not to implement this even though there most certainly would be a minor speedup benefit to gain.*

(iii) This strategy has been implemented in a way which uses only the trace *of the configuration, consistent with a strategy we shall describe below. However, as explained in Remark 3.9 this requires some extra care when determining the number of symmetric configurations.*

3.3.2 Upward counting

The numbers stored in Section 3.3.1 above can only be reused a maximum of 3 times in any given computation. To improve the range of memory-demanding time improvements, we make the following trivial observation

$$\textsc{LegoCount}(r, L, i_0, j_0, 0, \mathcal{C}) = \textsc{LegoCount}(r, L, i_0, j_0, h(L), \mathcal{C}) + \textsc{LegoCount}(r, L, i_0, j_0, 0, \mathcal{C} \wedge (1 \vee \cdots \vee h(L)))$$



0 0 0 1 0 0
1 1 0 1 0 0
0 1 1 0 1 1

Figure 16: Trace with binary key.

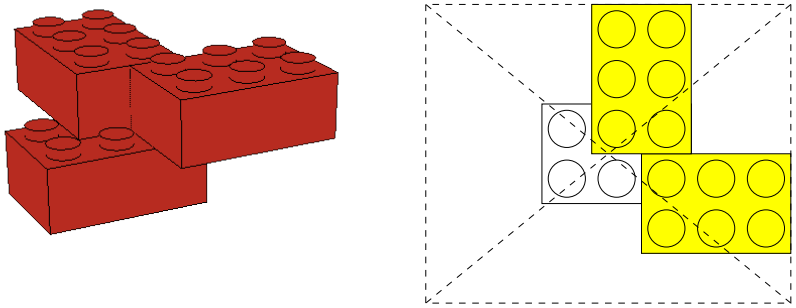


Figure 17: Trace of a structure with two levels

where $h(L)$ is the height of L, saying that either all future blocks are placed on top of L, or they are not.

The point of this is that LegoCount$(r, L, i_0, j_0, h(L), \mathcal{C})$ depends only on the topmost level of the structure L. In fact, the way the topmost level has been constructed out of individual blocks is also irrelevant, it is only the *trace* that matters, i.e. what cubes $[i-1, i] \times [j-1, j] \times [h(L)-1, h(L)]$ meet the structure. Such a trace can easily be transformed into a binary integer as indicated on Figure 16, and this integer can be used as a lookup key in, say, a hash table.

Similarly, symmetric counts can be established from knowing only the topmost layer and whether or not L is symmetric.

Implementation notes 3.9 *We have implemented the method of Section 3.3.1 above with binary integer keys also, but the strategy must be revised a little bit to allow that the key describes a structure with two levels. The fundamental idea is to encode a larger rectangle which is centered over the block in the lowest level, so that the relative position of the base block may be inferred. Again, it suffices to know the trace of the configuration rather that how it has been constructed, with one notable exception: Counts of*

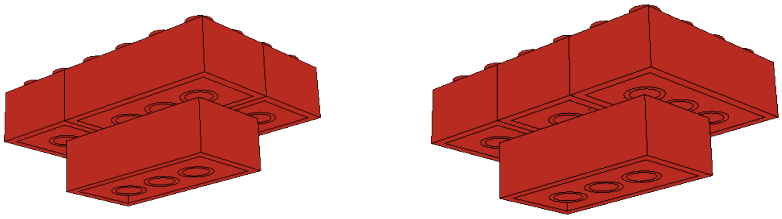


Figure 18: Same trace, different structure.

symmetric structures do depend on how the structure is analyzed into individual blocks. For instance, the number of ways to add one block to the structures in Figure 18 and get a structure which is invariant under a rotation of 180° (and has the base block in the lowest level) is 0 and 2, respectively, so the counts of symmetric structures must be computed in each case. The methods in Section 3.3.1 can be applied here.

4 Monte Carlo methods

In this chapter we want to discuss methods of estimating the number $T_{b\times w}(n)$ of structures made of n $b \times w$-blocks using Monte Carlo methods.

As introduced in Section 2.4, we can construct a surjective map

$$\Psi : \mathcal{U}_{b\times w}[n] \to \mathcal{T}_{b\times w}[n] \cup \{\mathsf{FAIL}\},$$

where $\mathcal{U}_{b\times w}[n]$ is the set of tuples of length $2\mathsf{T} + (n-2)(2\mathsf{T}-1)$ with exactly $n-1$ non-zero entries from $\{1, \ldots, \mathsf{E}\}$. The virtue of $\mathcal{U}_{b\times w}[n]$ is that we can easily calculate the the number of its elements by basic combinatorics. Our main idea is now to estimate the size of $\mathcal{T}_{b\times w}[n]$ by randomly making many tuples in $\mathcal{U}_{b\times w}[n]$. For each tuple t, we want to find out if $\Psi(t) \in \mathcal{T}_{b\times w}[n]$, and in this case determine $\Psi^{-1}(\Psi(t))$, i.e., the number of tuples which are mapped into the same structure as t

Assume that we have created m tuples from $\mathcal{U}_{b\times w}[n]$ in a random way. Let $t_1, \ldots, t_r$ be the ones mapped into $\mathcal{T}_{b\times w}[n]$ by Ψ, and let $p_k := \sharp\Psi^{-1}(\Psi(t_k))$ be the number of tuples in $\mathcal{U}_{b\times w}[n]$ which are mapped into the same structure

as t_k. Then an estimate α of $\sharp\mathcal{T}_{b\times w}[n]$ is

$$\alpha = \frac{\sharp\mathcal{U}_{b\times w}[n]}{m}\sum_{k=1}^{r}\frac{1}{p_k}.$$

Let

$$\beta = \frac{(\sharp\mathcal{U}_{b\times w}[n])^2}{m}\sum_{k=1}^{r}\frac{1}{p_k^2} - \sqrt{\alpha},$$

and we have the 95% confidence interval for $\sharp\mathcal{T}_{b\times w}[n]$

$$\left[\alpha - 1.96\sqrt{\frac{\beta}{m}}, \alpha + 1.96\sqrt{\frac{\beta}{m}}\right].$$

Thus, to compute α and β, for each randomly found configuration we have to efficiently calculate the number of tuples which create this configuration. Let $L = [B_1, \ldots, B_n]$ be a configuration containing the n blocks $B_1, \ldots, B_n$. We say, that a graph $G = (V, E)$ is a *characteristic graph* of L if a bijective map $\phi : L \to V$ exists, so that for every pair of blocks $B_i, B_j \in L$ there is an edge connecting $\phi(B_i)$ and $\phi(B_j)$ in G precisely when B_i is attached to B_j.

Now, we use the following result:

Proposition 4.1 *The number of tuples creating L is the number of spanning trees in a characteristic graph G of L.*

Proof: Let $G = (V, E)$ be a characteristic graph of L with a bijective map ϕ as described above. Let a tuple creating L be given, and assume that the blocks are introduced in the order $B_1, \ldots, B_n$ when decoding the tuple. We create a spanning tree of G while decoding the tuple. Let $G' = (\{v_1\}, \emptyset)$ be the empty graph of one vertex v_1 where $\phi(B_1) = v_1$. Assume that a new block B_i is added to the construction, introduced by the block B_j. Then we add the vertex $\phi(B_i)$ and the edge $\phi(B_i)\phi(B_j)$ to G'. By induction, we see that G' always is connected. When the whole tuple is decoded, we have handled $n-1$ introductions (i.e., G' contains $n-1$ edges), and then G' must be a tree. Since G' contains the same vertices as G and since the set of edges in G' is a subset of the edges in G, G' must be a spanning tree in G.

We are going to show that different tuples give different spanning trees. Assume that the different tuples p, q are representing the same configuration. Assume that the first entry at which they differ from each other is the i'th.

Let p_i and q_i be the i'th entry in p and q respectivly. It is impossible that $p_i \neq 0$ and $q_i \neq 0$ because then the blocks introduced would be different, and the configurations represented by p and q would be different.

Without loss of generality we can assume that $p_i \neq 0$ and $q_i = 0$. Let B_j be the block introduced by p_i and let B_k be the block on which p_i is representing a stud. Then the spanning tree representing p contains the edge $\phi(B_k)\phi(B_j)$ and the other one does not.

In the same way, we can construct a tuple from a spanning tree in G by letting the blocks be introduced by the blocks corresponding to the vertices to which they are adjencent in the spanning tree. If we have two different spanning trees, there must be a block introduced by two different blocks. Hence the tuples are different. □

We can use Kirchhoff's matrix tree theorem when calculating the number of spanning trees. Let $G = (V, E)$ be a graph, $V = \{v_1, \ldots, v_n\}$. Let δ_i be the degree of v_i and let $|E_{ij}|$ be the number of vertices from v_i to v_j. The theorem states, that the number N of spanning trees in a graph is the determinant of the degree matrix minus the adjacency matrix of the graph when we have deleted the i'th row and column where $i = 1, \ldots, n$, so let us delete the n'th row and column. That is, $N = \det A[t]$ where $A[t]$ has entries

$$a_{ij} = \begin{cases} \delta_i, & i = j \\ -|E_{ij}|, & i \neq j, \end{cases} \quad \text{where } i, j = 1, \ldots, n-1.$$

Proposition 4.2 *When $\Psi(t) \in \mathcal{T}_{b\times w}[n]$ we have*

$$\#\Psi^{-1}(\Psi(t)) = \det A[t] \cdot n \cdot \begin{Bmatrix} 1/2 & \Psi(t) \textit{ symmetric by } 90^\circ \\ 1 & \Psi(t) \textit{ symmetric by } 180^\circ \textit{ (not } 90^\circ\textit{)} \\ 2 & \textit{otherwise} \end{Bmatrix} 2^{-\delta_{b,w}}.$$

Proof: Proceed as in Proposition 3.1. □

Example 4.3 Consider the structure showed in Figure 19 with the characteristic graph is indicated. When we examine the vertices from the top down

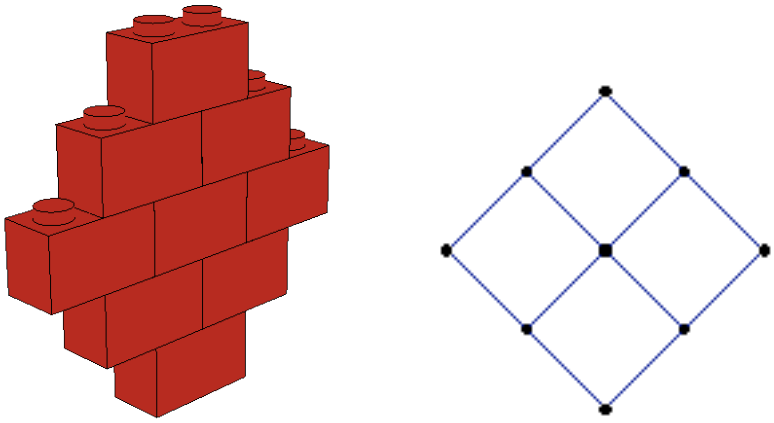


Figure 19: The characteristic graph of a structure.

we find that its degree matrix minus its adjacency matrix is

$$A = \begin{bmatrix} 2 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 3 & 0 & -1 & -1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 3 & 0 & -1 & -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 2 & 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & -1 & 0 & 4 & 0 & -1 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 2 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 & -1 & 0 & 3 & 0 & -1 \\ 0 & 0 & 0 & 0 & -1 & -1 & 0 & 3 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 2 \end{bmatrix},$$

and hence the characteristic graph has 192 spanning trees since this is the determinant of the 8×8 submatrix by deleting the last row and column. The structure is symmetric after a rotation by 180 degrees, and therefore the number of tuples corresponding to configurations representing the structure is $192 \cdot 9 = 1728$.

Implementation notes 4.4 *To concretely implement this strategy, we first have to choose the* T *distribution studs. This is of course no problem in the cases (i), (ii) and (iv) from Lemma 2.6. In the case (iii) we choose the distribution studs as equidistantly as possible along the long edges of the block. This seems to make it easy to make an even distribution. We sort the positions after how many of the distribution studs they employ, and distribute*

the positions greedily. For each number of employed distribution studs (starting with 1*, ending with* T*), we repeatedly find the position employing that number which employs a distribution stud that has had a minimal number of positions assigned and assign the position to that stud, until every position employing that certain number of distribution studs has been distributed. We have no guarantee that this will always produce an optimal distribution, but it seems to work well.*

We find the element count E *of the distribution, and then randomly make elements in the assoaciated set of tuples,* $\mathcal{U}_{b\times w}[n]$*. We construct configurations specified by each tuple when possible. Whenever we introduce a new block to the current configuration, we update the matrix relevant for using Kirchhoff's theorem, by finding the blocks attached to the new block. Each time we find a new configuration, we calculate the number of spanning trees in the characteristic graph from the matrix and determine whether the configuration is symmetric after a rotation by 90 or 180 degrees. Proposition 3.1 gives the number of tuples representing the same structure as the found configuration.*

References

[1] Bergfinnur Durhuus and Søren Eilers, *On the entropy of* LEGO. Submitted for publication.

[2] D. Zeilberger: Automated Counting of LEGO Towers. *J. Difference Eq. Appl.* **5** (1999), 323–333.

[3] G. Kirchhoff: *Über die Auflösung der Gleichungen, auf welche man bei der untersuchung der linearen verteilung galvanischer Ströme geführt wird.* Ann. Phys. Chem. **72** (1847), 497–508.

[4] N. J. A. Sloane, *The On-Line Encyclopedia of Integer Sequences*, www.research.att.com/~njas/sequences/A123762–www.research.att.com/~njas/sequences/A123849

[5] P. Flajolet and R. Sedgwick: Analytic combinatorics. Web edition, eighth printing.

Department for Mathematical Sciences
University of Copenhagen

Universitetsparken 5
DK-2100 Copenhagen Ø
Denmark
Mikkel.A@math.ku.dk,eilers@math.ku.dk