AUSTRALASIAN JOURNAL OF COMBINATORICS
Volume **92(1)** (2025), Pages 49–64

Simplifying modular lattices by removing doubly irreducible elements

Jukka Kohonen

Department of Mathematics and Systems Analysis
Aalto University
Espoo, Finland
jukka.kohonen@iki.fi

Abstract

Lattices are simplified by removing some of their doubly irreducible elements, resulting in smaller lattices called racks. All vertically indecomposable modular racks of $n \leq 40$ elements are listed, and the numbers of all modular lattices of $n \leq 40$ elements are obtained by Pólya counting. SageMath code is provided that allows easy access both to the listed racks, and to the modular lattices that were not listed. More than 3000-fold savings in storage space are demonstrated.

1 Introduction

One way to begin studies of a combinatorial family is to list all its members. Such listings can be quite large. A full listing of unlabeled vertically indecomposable modular lattices of n elements (here denoted by MV_n) for all $n \leq 30$ contains more than 828 million lattices, and measures over two gigabytes in a highly compressed form [13, 15]. Extending it to bigger n would be impractical.

Another approach to a large family is by structural theorems, such as Herrmann's theorem [10, Hauptsatz] that represents every modular lattice as an S-glued sum of its maximal complemented intervals (see also [7, Theorem 304]).

For practical computation, it is often useful to combine both approaches. A challenge is then to find structural theorems that balance two competing needs: being powerful enough to offer significant computational advantages, while remaining simple enough for efficient implementation.

Here we consider a structural simplification that allows us to represent the vast majority of MV_n as derivatives of a smaller set of lattices that we call *racks*. They are similar to Grätzer and Quackenbush's *frames* of planar modular lattices [9], but without the restriction to planarity. For a motivating example, consider the nonplanar modular lattices in Figure 1. The lattice on the left can be derived from

ISSN: 2202-3518

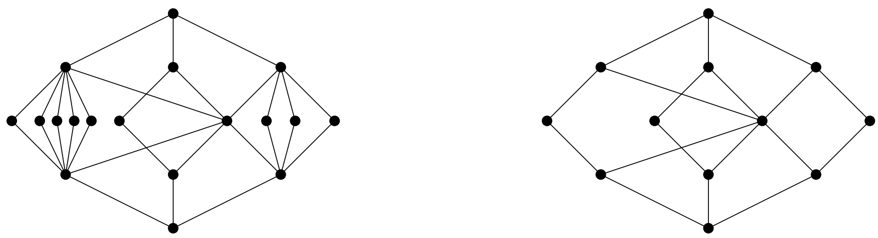

Figure 1: A modular lattice and its rack.

the one on the right by adding six doubly irreducible elements. By varying their placement we can obtain 64 nonisomorphic modular lattices, and by varying the number of added elements, we obtain many more.

In our approach only the racks are generated and stored. Other lattices in MV_n can then be Pólya-counted, or generated at will by adding doubly irreducible elements. To demonstrate the viability of this approach, a listing of all vertically indecomposable modular racks for $n \leq 40$ has been published [17]. It contains about 1.5 billion lattices and takes 5.7 gigabytes to store. In contrast, an explicit listing of MV_n for $n \leq 40$ would contain 5.2 *trillion* lattices and take about 20 terabytes. Our storage savings are thus more than 3000-fold. It would also take something like 60 cpu-core-years to produce the explicit listing.

Supplementary SageMath code [16] provides easy access to all modular lattices of $n \leq 40$ elements (both vertically decomposable and indecomposable). The access is through a virtual listing whose members can be accessed sequentially, by ordinal index, or uniformly at random. The accessed lattices are created on demand.

Another motivation for the work is that smaller collections are more meaningful for humans to study, and the removal of doubly irreducible elements may help in concentrating on other structural properties of the lattices.

2 Definitions and basic results

All our lattices are finite and nonempty. We write $\prec$ for the cover relation, $\overline{x}$ and $\underline{x}$ for the sets of upper and lower covers of x, and $|S|$ for set cardinality. A lattice element x is *doubly irreducible* if $|\overline{x}| = |\underline{x}| = 1$. An *unlabeled lattice* is an isomorphism class of lattices.

A lattice is *vertically decomposable* if it contains a *knot*, that is, an element distinct from top and bottom, and comparable with every element. Otherwise it is *vertically indecomposable*, or briefly *vi*. Throughout this work we focus in vi-lattices, since composing them into vertically decomposable lattices is straightforward.

Unrestricted addition or removal of doubly irreducible elements could severely affect the structure of a lattice. For example, adding such an element between the bottom and a coatom in a Boolean lattice B_3 yields a nonmodular lattice. To keep things in check, we consider addition and removal only at specific locations, called *decoration sites*.

For concreteness the following definitions are stated in terms of labeled lattices, though our main interest is in unlabeled lattices. We assume that the elements are equipped with an intrinsic linear order (for example, they could be integers). However, the only role of this intrinsic order is that from a number of similarly-placed doubly irreducible elements, we can choose a specific subset of a given size: the ones that have the highest labels. This ensures a well-defined operation for computation, though for unlabeled lattices the specific choice of elements is inconsequential.

Definition 2.1. A *decoration site* is a pair (a, b) of lattice elements with $\overline{a} = \underline{b}$ and $|\overline{a}| \geq 2$. We say that a is its *lower corner* and b is its *upper corner*.

Definition 2.2. The *trinkets* of a decoration site (a, b) are the $\min(d,\ |\overline{a}|{-}2)$ highest-labeled doubly irreducible elements between a and b, where d is the number of doubly irreducible elements there. A decoration site without trinkets is *empty*.

Definition 2.3. A *rack* is a lattice that contains no trinkets. If L is a lattice, $\operatorname{Rack} L$ is the sublattice obtained by removing its trinkets.

Put another way, $\operatorname{Rack} L$ is obtained by removing from each decoration site as many double irreducible elements as possible, while leaving at least two (possibly doubly irreducible) elements between the corners. The rationale for leaving two elements is that we want to retain the overall structure of the lattice.

Example 2.4. Let M_k denote a modular lattice that has k atoms and length 2. Then M_k has $k-2$ trinkets, and $\operatorname{Rack} M_k \cong M_2$.

Example 2.5. Both lattices in Figure 1 contain five decoration sites. On the left, one site has 4 trinkets, one has 2 trinkets, and three are empty. On the right, all sites are empty.

It is easy to see that racks of isomorphic lattices are isomorphic. Thus we can define the rack of an unlabeled lattice as $\operatorname{Rack}[L] = [\operatorname{Rack} L]$, where $[\,\cdot\,]$ denotes "the isomorphism class of". The lower corner of a decoration site can be the upper corner of another site, but otherwise decoration sites are disjoint: no two sites can have the same lower corner, or the same upper corner; and trinkets of one site cannot be corners or trinkets of another site.

Now it should be emphasized that the trinket-adding operation considered here is not novel as such. Indeed it is a special case of *one-point extension* [7, §1.1]. Also, Grätzer and Quackenbush [9] define a similar operation when L is a planar modular lattice with a given planar diagram: From each interval $[a, b]$ isomorphic to some M_k, consider the k elements between a and b in the order that they appear in the planar diagram. Keep the first and the last, and remove the other $k-2$ *internal* elements, which are by construction doubly irreducible. The result is a planar distributive lattice called Frame L. In the planar case we have Frame $L \cong$ Rack L, but our construction is more general since L need not be planar, and Rack L need not be distributive. We prefer the name *rack* because "frame" and "modular frame" are overloaded with other meanings.

Next we observe some structural properties that are preserved upon trinket addition and removal.

Lemma 2.6. *L and* Rack L *have the same decoration sites, and* Rack L *is a rack.*

Proof. Let $R = \text{Rack}\, L$, and let $T = L - R$. First we show that all decoration sites of L are still present in R. Let (a, b) be a decoration site in L. Since $|\overline{a}| = |\underline{b}| \geq 2$, the elements a and b are not trinkets, so they are also present in R. Also in R the upper covers of a are the same as the lower covers of b, because

$$\overline{a}_R = \overline{a}_L - T = \underline{b}_L - T = \underline{b}_R.$$

Here subscripts indicate the ambient lattice, so $\overline{a}_R$ means the upper covers of a in R. By construction we also have $|\overline{a}_R| \geq 2$. Thus (a, b) is a decoration site in R.

Next we prove that removing all trinkets from L does not create any new decoration sites. If some elements (a, b) of L are not a decoration site, it is either because $\overline{a}_L \neq \underline{b}_L$, or because $|\overline{a}_L| < 2$. In either case, removing some trinkets does not make (a, b) a decoration site in R.

Because Rack L has the same decoration sites as L, and their trinkets have been removed, it follows that Rack L has no trinkets, and is indeed a rack. □

From Lemma 2.6 it follows that Rack(Rack L) = Rack L, or in other words, Rack is an idempotent operation. This is convenient for our computations: every modular lattice can be reduced into a rack in a single step of removing all trinkets. In the opposite direction, any modular lattice can be created from its rack by *decorating* with some trinkets. The result of such decoration is, up to the naming of the trinkets, uniquely determined by two things: the rack itself, and the numbers of trinkets added to each decoration site.

Lemma 2.7. Rack L *is vertically decomposable if and only if L is vertically decomposable.*

Proof. For the "if" direction, let L contain a knot x. If $a \prec x \prec b$ in L, then $|\overline{a}| = 1$, thus x is not a trinket. Then x is not removed, and it is a knot in Rack L as well.

For the "only if" direction, let Rack L contain a knot x. If $u \prec x \prec v$ in Rack L, then (u, v) is not a decoration site because $|\overline{u}| = 1$. Thus in L there are no other elements between u and v than x itself. To see that x is a knot in L as well, we observe that every $y \in L$ is either also in Rack L, thus comparable to x; or a trinket of a decoration site (a, b), with either $y \prec b \preceq x$ or $x \preceq a \prec y$. In either case, y is comparable to x in L. It follows that x is comparable to all elements of L, and L is vertically decomposable. □

Lemma 2.8. *If the dual of L is L^δ, then the dual of* $\operatorname{Rack} L$ *is* $\operatorname{Rack}(L^\delta)$.

Proof. The decoration sites of L^δ are exactly the pairs (b, a) such that (a, b) is a decoration site of L, so L and L^δ have the same trinkets. Thus $\operatorname{Rack} L$ and $\operatorname{Rack}(L^\delta)$ have the same elements. The duality of their order is clear. □

Lemma 2.9. $\operatorname{Rack} L$ *is semimodular if and only if L is semimodular.*

Proof. We use Birkhoff's condition [7, Theorem 375]. Let $R = \operatorname{Rack} L$.

For the "if" direction, let L be semimodular, and let $x, y, a \in R$ be distinct elements such that $x, y \succ a$. Then also in L we have $x, y \succ a$, so there exists $b \in L$ such that $b \succ x, y$. Clearly b is not a trinket, so $b \in R$ and $b \succ x, y$ in R. Thus R is semimodular.

For the "only if" direction, let R be semimodular, and let x, y, a be distinct elements of L such that $x, y \succ a$. There are two cases: (1) If $x, y \in R$, then by semimodularity there exists $b \in R$ such that $b \succ x, y$. Then $b \succ x, y$ also in L. (2) If x or y is a trinket in L, then it belongs to a decoration site whose lower corner is a. Let b be the upper corner of that site. Since $\overline{a} = \underline{b}$, it follows that $b \succ x, y$. In both cases Birkhoff's condition is satisfied, so L is semimodular. □

Theorem 2.10. $\operatorname{Rack} L$ *is modular if and only if L is modular.*

Proof. Apply Lemmas 2.8 and 2.9 to the duals of $\operatorname{Rack} L$ and L. □

Theorem 2.11. *Every distributive lattice is a rack.*

Proof. Suppose that L is a lattice that is not a rack. Then it contains a decoration site (a, b) with a trinket t, and $\overline{a} = \underline{b}$ contains at least three elements t, u, v. Thus the elements a, t, u, v, b are a diamond (a sublattice isomorphic to M_3). By [7, Theorem 102], L is not distributive. □

Theorem 2.12 (Grätzer and Quackenbush [9])**.** *If L is a planar modular lattice, then* $\operatorname{Rack} L$ *is planar and distributive.*

Corollary 2.13. *Every planar modular rack is distributive.*

In contrast to Theorem 2.11 and Corollary 2.13, a modular rack need not be distributive, and a distributive rack need not be planar. Some examples can be found in Figure 2, where all vertically indecomposable modular racks of 1 to 13 elements are displayed. Nondistributive racks include 10.0 (no decoration sites) and 12.22 (B_3^+ in [9]; has five decoration sites). Distributive nonplanar racks include 8.0 (B_3, no decoration sites) and 10.1 (has one decoration site).

Planar distributive lattices are well understood. Every planar distributive lattice can be obtained from the direct product of two finite chains by removing two arbitrarily shaped "corners" from left and right [8]. With this characterization they are easily counted: OEIS gives a simple recurrence and counts them up to $n = 1000$ [19, A343161]. In the nonplanar, nondistributive case we have no such nice characterization, and computational methods are needed to obtain all racks.

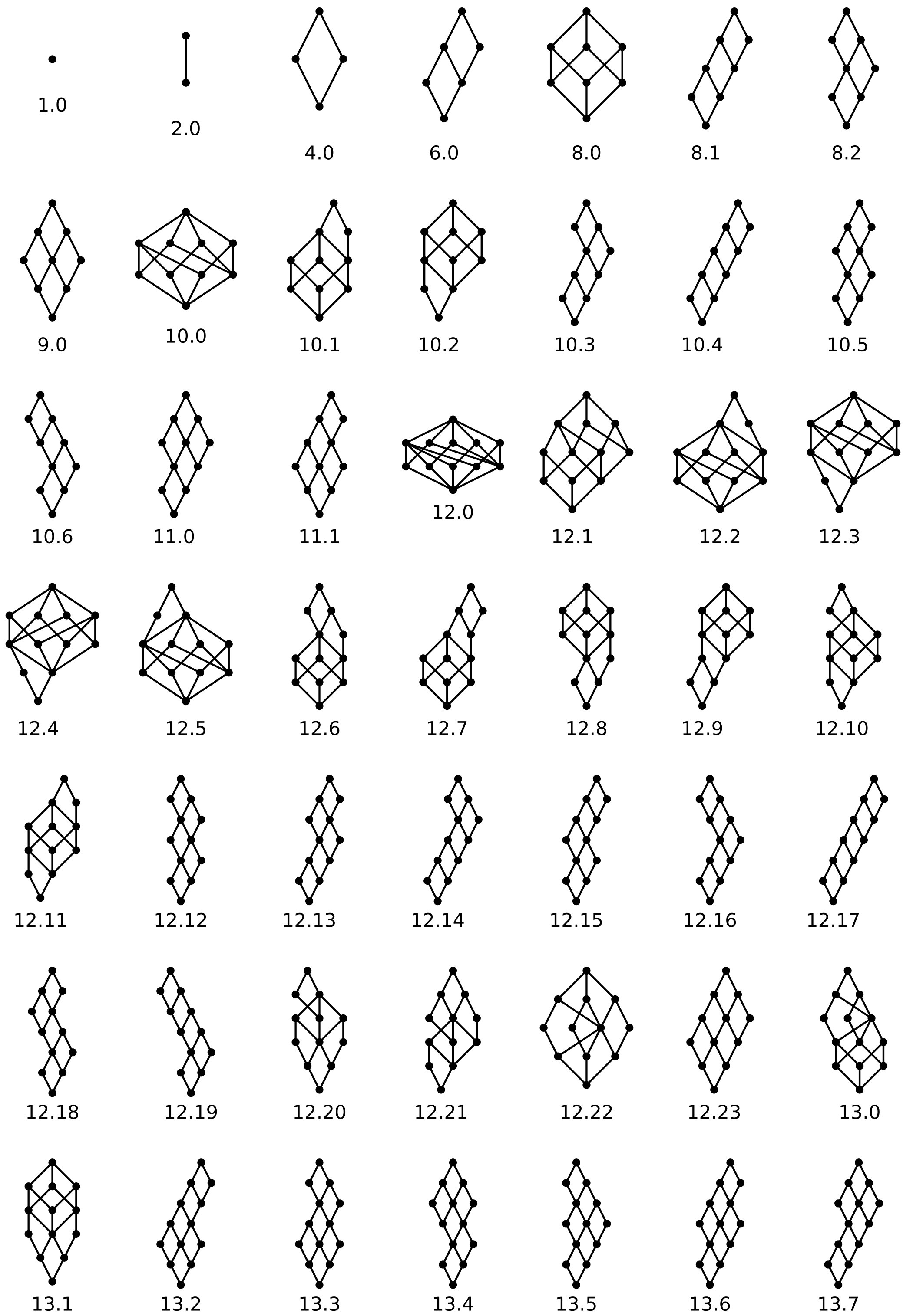

Figure 2: All modular vi-racks of 1 to 13 elements, labeled with the number of elements and an ordinal index.

3 Decoration under symmetry

From the previous section we know that every modular lattice L can be created by *decorating* a modular rack with trinkets. Let us now investigate two questions: Given a rack R and an integer m, how many nonisomorphic modular lattices can be created by placing m trinkets, and how can we actually construct them?

These tasks are complicated by the symmetries of the rack. Consider the rack in Figure 3. It has four decoration sites: $(0,4)$, $(1,6)$, $(2,7)$ and $(4,8)$. We can treat them as boxes where indistinguishable balls, or trinkets, are distributed. With (say) two trinkets, the number of ways is $\binom{2+4-1}{2} = 10$ by the stars-and-bars method, but only 7 of the resulting lattices are nonisomorphic.

In order to count the nonisomorphic results, we employ Pólya counting as follows. Suppose that we are decorating a rack that has k decoration sites, and their symmetry group is G. Let Z be the cycle index of G, that is, the polynomial

$$Z(t_1, \ldots, t_k) = \frac{1}{|G|} \sum_{g \in G} t_1^{c_1(g)} t_2^{c_2(g)} \cdots t_k^{c_k(g)},$$

where $c_i(g)$ is the number of cycles of length i in permutation g. Define the figure-counting series

$$A(x) = 1 + x + x^2 + x^3 + \ldots = 1/(1-x),$$

indicating that each decoration site can be allocated any nonnegative integer number of trinkets. Now by the Cycle Index Theorem [3, p. 77], the series

$$B(x) = Z(A(x), A(x^2), \ldots, A(x^k))$$

is the so-called function-counting series: the coefficient of its x^m term is the number of nonisomorphic ways to distribute a total of m trinkets to the k decoration sites.

Example 3.1. The rack in Figure 3 has mirror symmetry. Let us refer to the decoration sites by their lower corners. G has two elements, the identity $(0)(1)(2)(4)$

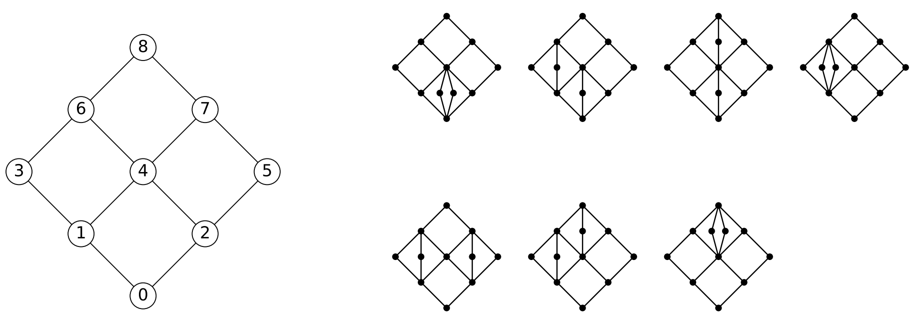

Figure 3: A modular rack and its seven nonisomorphic decorations with two trinkets.

and the mirroring (0)(1 2)(4), so $Z = \frac{1}{2}t_1^4 + \frac{1}{2}t_1^2 t_2$. We obtain

$$B(x) = 1 + 3x + 7x^2 + 13x^3 + 22x^4 + 34x^5 + \dots,$$

which tells us that there is 1 decoration with zero trinkets, 3 nonisomorphic decorations with one trinket, 7 with two trinkets, and so on.

Example 3.2. The rack in Figure 4 is more complicated. It has eleven decoration sites, and their symmetry group is isomorphic to the dihedral group D_4. The group fixes three sites and moves eight sites in a nontrivial way. It would be tedious to work out the symmetry and count the decorations manually. But in the supplementary SageMath code we have the function `count_decorations` that implements the method described above. With this function we find (in a few milliseconds) that if we were to decorate this rack with, say, 20 trinkets, we would obtain exactly 5 371 900 nonisomorphic modular lattices.

If we require the actual lattices (and not just their count), then we need a different tool. For this we use `IntegerVectorsModPermutationGroup` developed by Borie [2] and incorporated into the SageMath Combinatorics library. Given the symmetry group of a rack's decoration sites, and a number of trinkets m, this tool lists the different ways of distributing m balls to boxes under that symmetry. It is then straightforward to create the lattices by adding those numbers of trinkets to the sites. This is implemented in our function `list_decorations`. Explicit listing is of course much slower than counting.

Example 3.3. With the rack of Figure 3 (left), the possible allocations of two trinkets to the four decoration sites, subject to the symmetry, are $(2, 0, 0, 0)$, $(1, 1, 0, 0)$, $(1, 0, 0, 1)$, $(0, 2, 0, 0)$, $(0, 1, 1, 0)$, $(0, 1, 0, 1)$, and $(0, 0, 0, 2)$. By adding these numbers of trinkets, we obtain the modular lattices in Figure 3 (right).

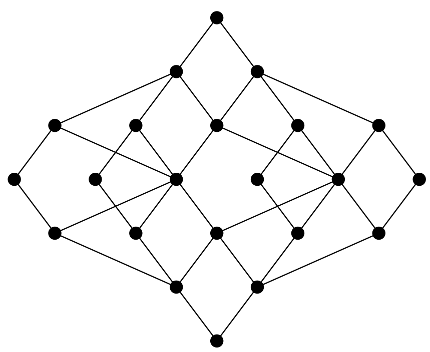

Figure 4: A rack with eleven decoration sites.

4 Generating and classifying the racks

Computations were performed in phases. First, all unlabeled modular vi-racks of $n \leq 40$ elements were listed using essentially the same C++ program that has been used in earlier works [13, 14]. Conditions were added to the program so that lattices with trinkets are not generated. This phase took 70.4 cpu-core days in total, running on a variety of AMD EPYC 7713 and Intel Xeon processors with nominal clock rates mostly between 2.0 and 2.5 GHz. The parts for $n = 36, 37, 38, 39, 40$ took 1.6, 3.6, 8.1, 17.7 and 38.5 cpu-core days, respectively, showing roughly 2.2-fold growth when n increases by one.

The speed benefits of our method were tested as follows. All unlabeled modular vi-lattices of 30 elements were explicitly listed using the same C++ program. This took 27.7 hours; in comparison, listing only the 30-element modular vi-racks took 0.36 hours. The time savings grow with n, and we estimate, very roughly, that an explicit listing for $n = 40$ would have taken 60 cpu-core years.

In the second phase the modular racks were analyzed and postprocessed. For each modular rack, the decoration sites were located, and the cycle index of their symmetry group was computed. A tally of racks was kept for each different cycle index encountered. The racks were also converted to a canonical form for ease of later use, and stored in XZ-compressed dig6 format [21]. This phase took 237.6 cpu-core days, somewhat more than the first phase, but this can be accounted to the relatively slow SageMath code that was used. A faster (e.g. C++) program for this phase could be written if necessary.

In the third phase all unlabeled modular vi-lattices of $n \leq 40$ elements were counted. This is a matter of seconds, because all that remains to do in this phase is to combine the counts of racks, from each cycle index, with the numbers of decorations per rack. More precisely, we have

$$|\mathrm{MV}_n| = \sum_{k=1}^{n} \sum_{Z \in \mathcal{Z}(k)} R(k, Z) \cdot D(Z, n-k),$$

where $\mathcal{Z}(k)$ is the set of all different cycle indices in k-element modular vi-racks, $R(k, Z)$ is the number of unlabeled k-element modular vi-racks whose decoration sites have a symmetry group with cycle index Z, and $D(Z, n-k)$ is the number of decorations of each such rack with $n-k$ trinkets. Here $R(k, Z)$ comes from our tallying in the second phase, and $D(Z, n-k)$ is calculated with the method described in Section 3. The summation is fast, because it does not involve very many terms: even $|\mathcal{Z}(40)|$ is only 1614 (see Table 1). Although there are hundreds of millions of different racks, they can be grouped into a relatively few types by their decoration symmetry.

Finally the numbers of unlabeled modular lattices of n elements (M_n) were calculated using the well-known recurrence [11]

$$|\mathrm{M}_n| = \sum_{j=2}^{n} |\mathrm{MV}_j| \cdot |\mathrm{M}_{n-j+1}|,$$

which counts the ways of composing the vi-lattices vertically.

Several consistency checks were performed in order to increase the reliability of the computational results. First, the racks were generated twice on different computer systems. The output files were verified to be byte-by-byte identical by comparing their MD5 checksums.

Secondly we counted the occurrences of each rank sequence in the rack listings, and verified that all those counts are consistent with duality. For example, among all unlabeled modular vi-racks of 40 elements, exactly 4 265 have the rank sequence $1, 3, 5, 6, 7, 7, 6, 4, 1$, and another 4 265 (their duals) have the reverse of that, namely $1, 4, 6, 7, 7, 6, 5, 3, 1$.

Thirdly the numbers of modular vi-lattices were verified against previous results, which went to $n = 30$ [13] and to $n = 35$ [14]. We must note that those two previous countings are based on the same underlying lattice-listing C++ program that was also used here. However, the combinatorial methods are quite different, so we are reaching the same numbers by three different methods.

Fourthly, a listing of n-element modular vi-racks must contain all distributive vi-lattices of that size. We scanned the rack lists for distributive lattices of up to 40 elements, and verified that the counts match earlier results [6, 14].

Finally, the rack listings and their decorations were comprehensively compared, lattice by lattice, against the explicit listings of modular vi-lattices published earlier [13, 15]. This test went both ways. We scanned the explicit listings for racks, and verified that they are the same racks as those listed in the present work. Also, we listed all decorations from our racks, and verified that this exactly re-creates the modular vi-lattices that were listed earlier, up to isomorphism. This test was computationally intensive, and was performed only up to 19 elements.

5 Numerical results

The counting results are displayed in Table 1. The second column contains $|\mathcal{Z}(n)|$, the number of different cycle indices of the decoration symmetries in n-element modular vi-racks. The last three columns contain the numbers of unlabeled modular vi-racks, modular vi-lattices, and modular lattices, respectively. The last two columns were previously known up to $n = 35$ from a different method of counting [14].

Figure 5 illustrates how the numbers of unlabeled lattices in different families depend on the number of elements. The data are from Table 1 and the OEIS entries A072361 and A345734 [19]. Although precise asymptotics are not known, empirically the growth rate of modular vi-racks is about $\Theta(1.9^n)$, and it is closer to distributive vi-lattices than to modular vi-lattices. We can see that much of the apparent multitude of modular vi-lattices is just decoration.

Table 1: Numbers of unlabeled modular lattices of n elements.

n	Cycle indices	Mod. vi-racks	Mod. vi-lattices A342132	Mod. lattices A006981 [19]
1	1	1	1	1
2	1	1	1	1
3	0	0	0	1
4	1	1	1	2
5	0	0	1	4
6	1	1	2	8
7	0	0	3	16
8	2	3	7	34
9	1	1	12	72
10	3	7	28	157
11	1	2	54	343
12	7	24	127	766
13	2	8	266	1718
14	8	70	614	3899
15	13	44	1356	8898
16	12	215	3134	20475
17	16	173	7091	47321
18	23	711	16482	110024
19	27	657	37929	256791
20	33	2367	88622	601991
21	42	2561	206295	1415768
22	57	7989	484445	3340847
23	60	9745	1136897	7904700
24	80	27540	2682451	18752943
25	98	36744	6333249	44588803
26	115	95975	15005945	106247120
27	140	137895	35595805	253644319
28	179	337911	84649515	606603025
29	212	514821	201560350	1453029516
30	251	1200282	480845007	3485707007
31	318	1915896	1148537092	8373273835
32	375	4291336	2747477575	20139498217
33	440	7113503	6579923491	48496079939
34	549	15430316	15777658535	116905715114
35	655	26356273	37871501929	282098869730
36	772	55742330	90998884153	681357605302
37	944	97509982	218856768070	1647135247659
38	1133	202116488	526836817969	3985106742170
39	1319	360362439	1269255959032	9649048527989
40	1614	735089580	3060315929993	23379906035595

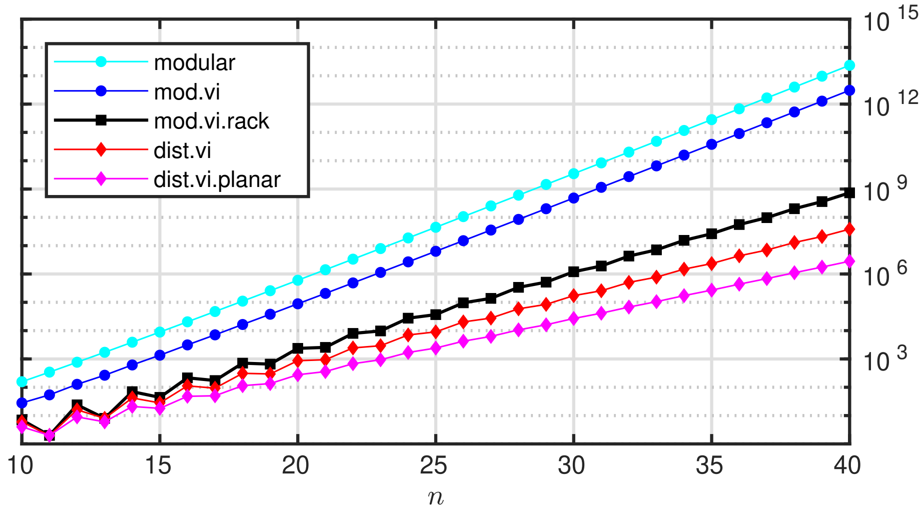

Figure 5: Numbers of modular lattices, modular vi-lattices, modular vi-racks, distributive vi-lattices, and planar distributive vi-lattices of n elements (all up to isomorphism).

6 SageMath implementation

A library of SageMath code was developed to support the use of the rack listings [16]. An overview of its capabilities is given here.

The library contains utility functions for reading lattices from text files, adding and removing trinkets, finding the decoration sites of a lattice, and so on. But the central part of the library is the combinatorial core of our approach: given a rack and a number of trinkets, functions `count_decorations` and `list_decorations` can count and list all the ensuing nonisomorphic decorations. Counting is much faster than listing, because it is done through Pólya counting as described in Section 3. Instead of listing all decorations, one can also ask for a specific decoration by its ordinal index. This is useful, for example, if one wants to sample uniformly at random from a large set of decorations.

Both the file access and the combinatorial core are encapsulated into classes belonging to the category `FiniteEnumeratedSets`. From the outside, such a class appears as a virtual list whose members can be *iterated* (accessed sequentially) and *unranked* (accessed by an integer index). It is up to the implementation how the members are produced when asked for. We have nested levels of wrappers. The lowest-level wrapper encapsulates a rack listing that resides in an XZ-compressed text file. The unranker seeks to the correct position and reads one lattice from there. Further wrappers encapsulate decoration and vertical composition. The unrankers obtain the appropriate racks from the file wrapper, decorate with trinkets, and compose vertically as needed. As a result we have easily accessible virtual lists of modular vi-lattices and modular lattices of $n \leq 40$ elements.

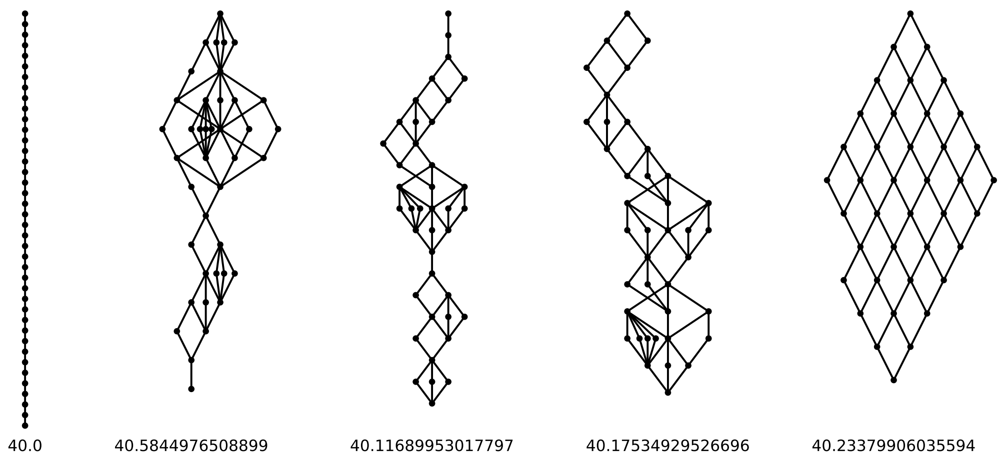

Figure 6: An evenly spaced sample of five 40-element modular lattices, from the first to the last in our virtual listing.

Example 6.1. The following code asks for a nonrandom, approximately evenly spaced sample of five modular lattices of 40 elements, ranging from the first one to the last one in the virtual listing, according to an intrinsic order. The sample is promptly returned without ever having to construct 23 trillion lattices.

```
sage: M = ModularLattices(40)
sage: card = M.cardinality(); print(card)
23379906035595
sage: LL = list(M[round((card-1)*i/4)] for i in [0,1,2,3,4]]
```

The first line takes five seconds on a laptop computer, as it opens the files and sets things up for fast retrieval. The next lines take less than a second. The sample is displayed in Figure 6. Note that in our virtual listing, racks are ordered by the number of decoration sites, and vertically decomposable lattices are ordered by the size of the lowest vertically indecomposable component. So it is not a coincidence that the first lattice in the listing is the chain, and the last one is a planar distributive lattice with many (empty) decoration sites.

We note in passing that for automatically generated modular lattices, SageMath's `plot` often produces diagrams that contain unnecessarily many crossing edges. Our `lattice_plot` attempts to draw somewhat prettier lattice diagrams. It recognizes trinkets and hangs them out in an aptly trinketlike fashion. All lattice diagrams in this paper, including Figure 6, were automatically produced with this function.

7 Related work

This work was initially inspired by the simple observation that modular lattices contain lots of doubly irreducible elements. Several previous works share the idea of adding single elements or other simple features to a lattice, sometimes using enumerative combinatorics to count the ways of doing that.

As already mentioned, Grätzer and Quackenbush described a reduction similar to ours, where internal doubly irreducible elements are removed from a planar modular lattice [9]. The present work removes the restriction to planarity, so that we can represent all modular lattices.

Jipsen and Lawless proved an $\Omega(2^n)$ lower bound for the number of unlabeled modular lattices using a recursive construction, where each step either extends the lattice vertically, or adds a doubly irreducible element [11].

Bhavale and Waphare studied dismantlable lattices whose reducible elements are comparable to each other; such lattices consist of a single main chain with attached side chains. The possible placements of the side chains were counted using binomial coefficients [1].

The present author studied rank-three graded lattices (without modularity) and reduced them by removing all doubly irreducible atoms; those atoms were then treated as indistinguishable balls to be placed into partially distinguishable boxes, leading to Pólya counting [12].

8 Concluding remarks

We have seen that the removal of some doubly irreducible elements can lead to big savings in computation and storage. Both theoretical and practical tools were needed to make it happen. Structural theorems provide the foundation for such work, but algorithms and computations bring the theorems to life. The operation considered here, where racks are decorated with trinkets, is at the same time a special case of one-point extension (for lattices in general), and a generalization of Grätzer and Quackenbush's operation of adding eyes (for planar modular lattices). The level of specialization has been chosen so as to be amenable to efficient computation (counting and access to individual lattices).

From Figure 2 one may observe that many of our racks could be composed from smaller components by gluing constructions. For example, lattice 10.1 is the Dilworth gluing of a B_3 with a B_2 over a two-element lattice, and lattice 9.0 could be represented as an S-glued system of four copies of B_2 (cf. Figures 1 and 2 of Day and Freese [5] for a similar gluing of copies of B_3). Conceivably, such decompositions could be used to further simplify and reduce our exhaustive listings of lattices. The challenge then is to employ gluing in a computationally efficient manner, so that one can still effectively access the individual lattices at will. This is an interesting topic for further study.

This work provides a virtual *listing* of modular lattices. For some uses one might want a *database* that would allow efficient queries according to various criteria, such as number of levels, numbers of elements at specific levels, whether the lattice is slim, complemented, planar, and so on. This could be similar to what the online House of Graphs provides for graphs [4]. An intriguing prospect is a virtual database, one that would represent a large collection in terms of a smaller explicit collection, similarly to what was done here, and would still support efficient queries. For this to work, one would have to take into account how some properties are preserved and others affected by whatever structural reductions one is employing. Making it click together might involve some interesting combinatorics.

Acknowledgments

The author wishes to thank the anonymous referees for their illuminating comments. Thanks are also due to the numerous authors of the computational tools that were essential for this work. SageMath [20] and its component libraries facilitate computation with graphs, symmetry groups, cycle indices, and symbolic power series. The underlying rack-listing program relies on Nauty [18] for isomorph avoidance. XZ [21] compresses the lattice files efficiently while enabling fast random access to individual lattices. Computational resources were generously provided by CSC — IT Center for Science and by the Aalto Science-IT project.

References

[1] A. N. Bhavale and B. N. Waphare, Basic retracts and counting of lattices, *Australas. J. Combin.* **78** (2020), 73–99.

[2] N. Borie, Generating tuples of integers modulo the action of a permutation group and applications, in DMTCS Proceedings vol. AS, 25th Int. Conf. on Formal Power Series and Algebraic Combinatorics (FPSAC 2013), pp. 767–778.

[3] P. J. Cameron, Combinatorics 2: Structure, symmetry and polynomials, lecture notes, `https://cameroncounts.files.wordpress.com/2016/04/acnotes2.pdf`, 2015.

[4] K. Coolsaet, S. D'hondt and J. Goedgebeur, House of Graphs 2.0: a database of interesting graphs and more, *Discrete Appl. Math.* **325** (2023), 97–107.

[5] A. Day and R. Freese, The role of gluing constructions in modular lattice theory, in K. P. Bogart, R. Freese and J. P. S. Kung (editors), *The Dilworth Theorems: Selected Papers of Robert P. Dilworth*, Birkhäuser, 1990, pp. 251–260.

[6] M. Erné, J. Heitzig and J. Reinhold, On the number of distributive lattices, *Electron. J. Combin.* **9** (2002), article #R24, 23 pp.

[7] G. Grätzer, *Lattice Theory: Foundation*, Birkhäuser, 2011.

[8] G. Grätzer and E. Knapp, Notes on planar semimodular lattices I, construction, *Acta Sci. Math. (Szeged)* **73** (2007), 445–462.

[9] G. Grätzer and R. W. Quackenbush, The variety generated by planar modular lattices, *Algebra Universalis* **63** (2010), 187–201.

[10] Ch. Herrmann, S-verklebte Summen von Verbänden, *Math. Z.* **130** (1973), 255–274.

[11] P. Jipsen and N. Lawless, Generating all finite modular lattices of a given size, *Algebra Universalis* **74** (2015), 253–264.

[12] J. Kohonen, Counting graded lattices of rank 3 that have few coatoms, *Internat. J. Algebra Comput.* **29** (2019), 663–680.

[13] J. Kohonen, Generating modular lattices of up to 30 elements, *Order* **36** (2019), 423–435.

[14] J. Kohonen, Cartesian lattice counting by the vertical 2-sum, *Order* **39** (2022), 113–141.

[15] J. Kohonen, Lists of finite lattices (modular, semimodular, graded and geometric), data files at EUDAT B2SHARE, `http://doi.org/10.23728/b2share.dbb096da4e364b5e9e37b982431f41de`, 2018.

[16] J. Kohonen, Modular racks, SageMath library, `https://bitbucket.org/jkohonen/modular-racks`, 2023.

[17] J. Kohonen, Racks of vertically indecomposable modular lattices (version 1), data files at Fairdata.fi, 2023, `https://doi.org/10.23729/1c446a98-23f7-49a3-a040-f75fbd111c0d`.

[18] B. D. McKay and A. Piperno, *Nauty and Traces User's Guide (version 2.8.6)*, 2022.

[19] The OEIS Foundation, The On-line Encyclopedia of Integer Sequences, `https://oeis.org`.

[20] The Sage developers, SageMath, the Sage mathematics software system.

[21] The Tukaani project, XZ utils, `https://tukaani.org/xz/`.

(Received 27 Nov 2023; revised 27 Mar 2025)