Generating all modular lattices of a given size

BLAST 2013

Nathan Lawless

Chapman University

August 8, 2013

Outline

- Modular Lattices: Definitions
- The Objective: Generating and Counting Modular Lattices
- The Original Algorithm: Generating All Finite Lattices
- Improving the Algorithm
- Generating Modular and Semimodular Lattices
- Results
- Lower Bound on Modular Lattices
- Conclusion

Modular Lattices

- A **modular lattice** M is a lattice that satisfies the modular law for all $x, y, z \in M$:

$$x \geq z \text{ implies } x \wedge (y \vee z) = (x \wedge y) \vee z$$

or equivalently:

$$x \wedge [y \vee (x \wedge z))] = (x \wedge y) \vee (x \wedge z).$$

Modular Lattices

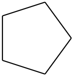
- A **modular lattice** M is a lattice that satisfies the modular law for all $x, y, z \in M$:

$$x \geq z \text{ implies } x \wedge (y \vee z) = (x \wedge y) \vee z$$

or equivalently:

$$x \wedge [y \vee (x \wedge z))] = (x \wedge y) \vee (x \wedge z).$$

- An alternative way to view modular lattices is by **Dedekind's Theorem**: *L is a nonmodular lattice iff N_5 can be embedded into L.*



N_5

Modular Lattices

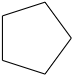
- A **modular lattice** M is a lattice that satisfies the modular law for all $x, y, z \in M$:

$$x \geq z \text{ implies } x \wedge (y \vee z) = (x \wedge y) \vee z$$

or equivalently:

$$x \wedge [y \vee (x \wedge z))] = (x \wedge y) \vee (x \wedge z).$$

- An alternative way to view modular lattices is by **Dedekind's Theorem**: *L is a nonmodular lattice iff N_5 can be embedded into L.*



N_5

- Examples of modular lattices are:

Modular Lattices

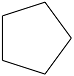
- A **modular lattice** M is a lattice that satisfies the modular law for all $x, y, z \in M$:

$$x \geq z \text{ implies } x \wedge (y \vee z) = (x \wedge y) \vee z$$

or equivalently:

$$x \wedge [y \vee (x \wedge z))] = (x \wedge y) \vee (x \wedge z).$$

- An alternative way to view modular lattices is by **Dedekind's Theorem**: *L is a nonmodular lattice iff N_5 can be embedded into L.*



N_5

- Examples of modular lattices are:
  - Lattices of subspaces of vector spaces.

Modular Lattices

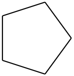
- A **modular lattice** M is a lattice that satisfies the modular law for all $x, y, z \in M$:

$$x \geq z \text{ implies } x \wedge (y \vee z) = (x \wedge y) \vee z$$

  or equivalently:

$$x \wedge [y \vee (x \wedge z))] = (x \wedge y) \vee (x \wedge z).$$

- An alternative way to view modular lattices is by **Dedekind's Theorem**: *L is a nonmodular lattice iff N_5 can be embedded into L.*



N_5

- Examples of modular lattices are:
  - Lattices of subspaces of vector spaces.
  - Lattices of ideals of a ring.

Modular Lattices

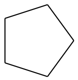
- A **modular lattice** M is a lattice that satisfies the modular law for all $x, y, z \in M$:

$$x \geq z \text{ implies } x \wedge (y \vee z) = (x \wedge y) \vee z$$

  or equivalently:

$$x \wedge [y \vee (x \wedge z))] = (x \wedge y) \vee (x \wedge z).$$

- An alternative way to view modular lattices is by **Dedekind's Theorem**: *L is a nonmodular lattice iff N_5 can be embedded into L.*



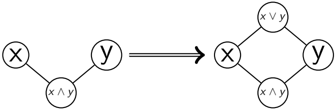
N_5

- Examples of modular lattices are:
  - Lattices of subspaces of vector spaces.
  - Lattices of ideals of a ring.
  - Lattices of normal subgroups of a group.

Semimodular Lattices

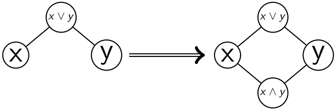
- A lattice L is **semimodular** if for all $x, y \in L$

$$x \wedge y \prec x, y \text{ implies that } x, y \prec x \vee y.$$



- A lattice L is **lower semimodular** if for all $x, y \in L$

$$x, y \prec x \vee y \text{ implies that } x \wedge y \prec x, y.$$



- **Theorem:** A finite lattice L is modular if and only if it is semimodular and lower semimodular.

Our Objective

We wish to come up with an algorithm which can efficiently generate all possible finite modular lattices of a given size n up to isomorphism. We further want to apply it to other types of lattices.

Why is this important?

1. Being used for generation of modular lattices and related structures.
2. Providing a tool to verify conjectures and/or find counterexamples.
3. Better understanding of modular lattices.
4. Discovering new structural properties of modular lattices.

Generating Finite Lattices

Heitzig and Reinhold [2000] developed an **orderly algorithm** to enumerate all finite lattices and used it to count the number of lattices up to size 18. To explain their algorithm, we give some definitions related to posets and lattices:

- We say that b is a **cover** of a if $a < b$ and there is no element c such that $a < c < b$, and denote this by $a \prec b$.

Generating Finite Lattices

Heitzig and Reinhold [2000] developed an **orderly algorithm** to enumerate all finite lattices and used it to count the number of lattices up to size 18. To explain their algorithm, we give some definitions related to posets and lattices:

- We say that b is a **cover** of a if $a < b$ and there is no element c such that $a < c < b$, and denote this by $a \prec b$.
- We say an element is an **atom** if it covers the bottom element.

Generating Finite Lattices

Heitzig and Reinhold [2000] developed an **orderly algorithm** to enumerate all finite lattices and used it to count the number of lattices up to size 18. To explain their algorithm, we give some definitions related to posets and lattices:

- We say that b is a **cover** of a if $a < b$ and there is no element c such that $a < c < b$, and denote this by $a \prec b$.
- We say an element is an **atom** if it covers the bottom element.
- We call $\uparrow A = \{x \in L \mid a \leq x \text{ for some } a \in A\}$ the **upper set** of A.

Generating Finite Lattices

Heitzig and Reinhold [2000] developed an **orderly algorithm** to enumerate all finite lattices and used it to count the number of lattices up to size 18. To explain their algorithm, we give some definitions related to posets and lattices:

- We say that b is a **cover** of a if $a < b$ and there is no element c such that $a < c < b$, and denote this by $a \prec b$.
- We say an element is an **atom** if it covers the bottom element.
- We call $\uparrow A = \{x \in L \mid a \leq x \text{ for some } a \in A\}$ the **upper set** of A.
- An **antichain** is a subset of L in which any two elements in the subset are incomparable.

Generating Finite Lattices

Heitzig and Reinhold [2000] developed an **orderly algorithm** to enumerate all finite lattices and used it to count the number of lattices up to size 18. To explain their algorithm, we give some definitions related to posets and lattices:

- We say that b is a **cover** of a if $a < b$ and there is no element c such that $a < c < b$, and denote this by $a \prec b$.
- We say an element is an **atom** if it covers the bottom element.
- We call $\uparrow A = \{x \in L \mid a \leq x \text{ for some } a \in A\}$ the **upper set** of A.
- An **antichain** is a subset of L in which any two elements in the subset are incomparable.
- The set of all maximal elements in L is called the first level of L ($Lev_1(L)$). The **(m+1)-th level** of L can be recursively defined by $lev_{m+1}(L) = Lev_1(L - \bigcup_{i=1}^{m} Lev_i(L))$.

Counting Finite Lattices (continued)

Let A be an antichain of a lattice L. If A satisfies A1, we call it a **lattice-antichain**.

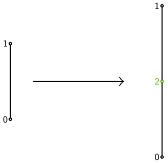
(A1) For any $a, b \in \uparrow A$, $a \wedge b \in \uparrow A \cup \{0\}$.

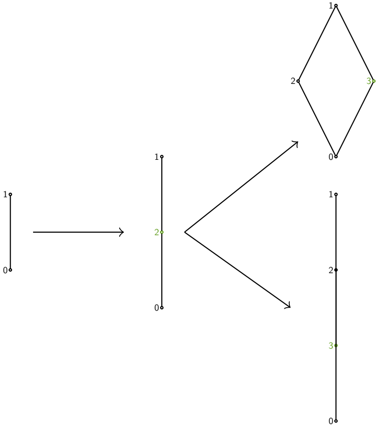
$\mathbf{L^A}$ is constructed from L by adding an atom which is covered by exactly the elements in A. If A satisfies (A1), then L^A is a lattice. (Heitzig and Reinhold, 2000).

A recursive algorithm can be formulated that generates for a given natural number $n \geq 2$ exactly all canonical lattices up to n elements starting with the two element lattice:

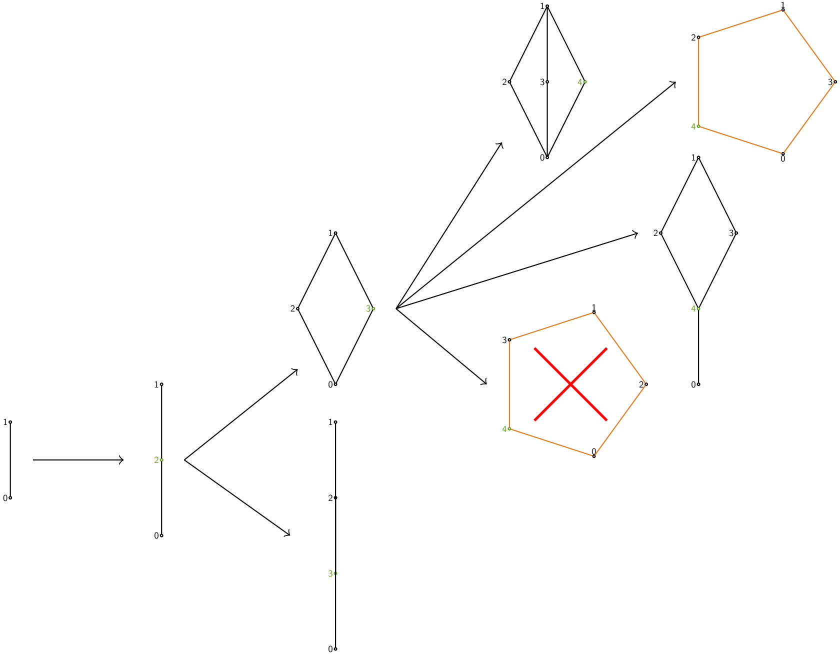
```
next_lattice(integer m, canonical m-lattice L)
begin
    if m < n then
        for each lattice-antichain A of L do
            if L^A is a canonical lattice then
                next_lattice (m + 1, L^A)
    if m = n then output L
end
```

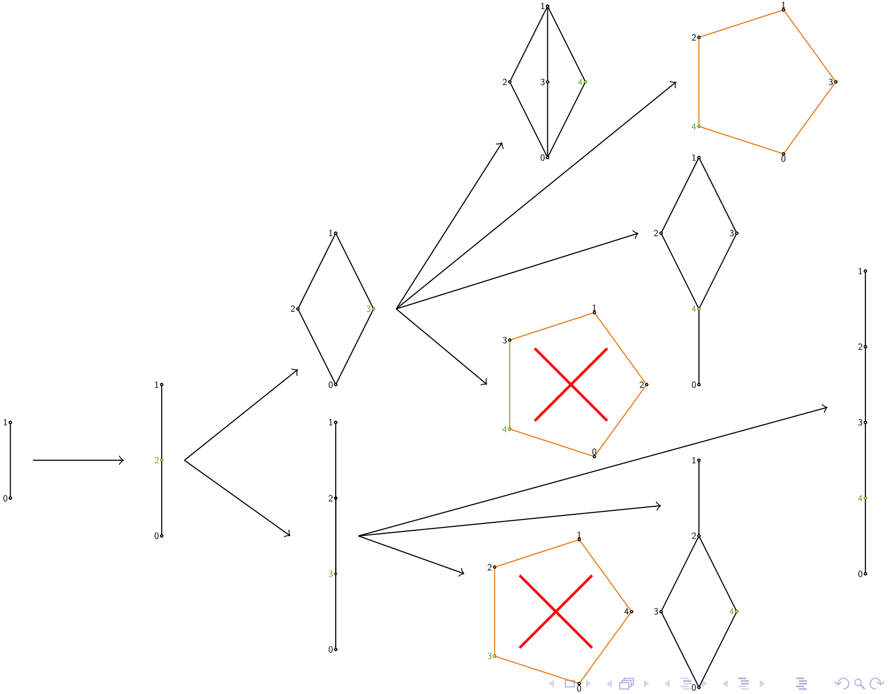

1
0


1
0
1
2
0



1
0
1
2
0
1
2
3
0
1
2
3
0


1
0
1
2
0
1
2
3
0
1
2
3
0
1
2
3
4
0
1
2
3
4
0
1
2
3
4
0
1
2
3
4
0



Dealing with Isomorphisms

- In order to select one isomorphic copy, a weight is defined for each lattice. If a lattice has the lowest weight among all it's permutations, it is considered canonical.
- However, this is an expensive check since it requires checking all permutations for each lattice (with some restrictions).
- The algorithm runtime can be improved by introducing a *canonical path extension*, introduced by McKay (1998):
  - We only use one arbitrary representative of each orbit in the lattice antichains of L.
  - When L^A is generated, we perform a "canonical deletion". If L is automorphic to the generated lattice, we consider L^A canonical.

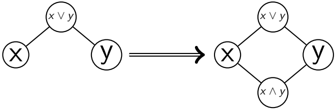
Counting Finite Lattices: Semimodular Lattices

This algorithm can be modified such that when a lattice of size n is generated, the algorithm checks if it is (semi)modular.
Since semimodular and modular lattices are a very small fraction of all lattices, we present some results to reduce the search space of the algorithm. Here, $Lev_k(L)$ and $Lev_{k-1}(L)$ denote the bottom and second bottom levels of L respectively.

- **Semimodular Lattices Theorem:** When generating semimodular lattices, for a lattice L, we only consider antichains A which satisfy **(A1)** and all of the following conditions:
  (A2) $A \subseteq Lev_{k-1}(L)$ or $A \subseteq Lev_k(L)$.
  (A3) If $A \subseteq Lev_k(L)$, there are no atoms in $Lev_{k-1}(L)$.
  (A4) For all $x, y \in A$, x and y have a common cover.

- **Modular Lattices Theorem:** When generating modular lattices, for a lattice L, we only consider antichains A which satisfy **(A1-4)** and

  (A5) If $A \subseteq Lev_k(L)$, $Lev_{k-1}(L)$ satisfies lower semimodularity (ie: for all $x, y \in Lev_{k-1}(L)$, $x, y \prec x \vee y$ implies $x \wedge y \prec x, y$)



Runtime Analysis

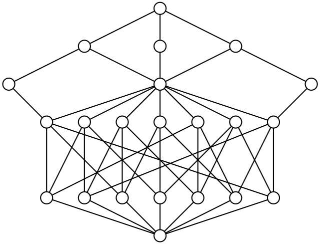
- Calculation of modular lattices of size n takes approximatelly 5.5 times the time used to generate all modular lattices of size $n-1$.
- In order to reach higher numbers, the algorithm was parallelized using the Message Passing Interface (MPI).
- Approximately **50 hours** were required to calculate all modular lattices of size 22 running the algorithm in parallel on 64 CPUs. It is estimated it would have taken **1 month** with the serial version.

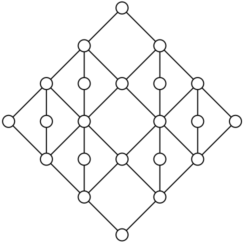
Results

n	Lattices	Semimod. Latt.	Mod. Latt.
1	1	**1**	1
2	1	**1**	1
3	1	**1**	1
4	2	**2**	2
5	5	**4**	4
6	15	**8**	8
7	53	**17**	16
8	222	**38**	34
9	1,078	**88**	72
10	5,994	**212**	157
11	37,622	**530**	343
12	262,776	**1376**	766

Results

n	Lattices	Semimod. Latt.	Mod. Latt.
1	1	**1**	1
2	1	**1**	1
3	1	**1**	1
4	2	**2**	2
5	5	**4**	4
6	15	**8**	8
7	53	**17**	16
8	222	**38**	34
9	1,078	**88**	72
10	5,994	**212**	157
11	37,622	**530**	343
12	262,776	**1376**	766
13	2,018,305	**3,693**	**1,718**
14	16,873,364	**10,232**	**3,899**
15	152,233,518	**29,231**	**8,898**
16	1,471,613,387	**85,906**	**20,475**
17	15,150,569,446	**259,291**	**47,321**
18	165,269,824,761	**802,308**	**110,024**
19	–	**2,540,635**	**256,791**
20	–	**8,220,218**	**601,991**
21	–	**27,134,483**	**1,415,768**
22	–	**91,258,141**	**3,340,847**





Lower Bound on Modular Lattices

- **Theorem:** The number of modular lattices of size n up to isomorphism is greater or equal to 2^{n-3}.
- **Outline of proof:** Let L_3 be the three element lattice with 0 and 1 as bottom and top respectively, and let $n-1$ the last element added. Consider the following two extensions of an n-lattice L:
  $L_\alpha = L^A$ where $A = \{x \in L \mid x \succ 0\}$
  $L_\beta = L^{\{a\}}$ for an arbitary a such that $a \succ n-1$

  Idea: Each modular lattice L will generate two unique modular lattices L_α and L_β.

Open Question: Upper Bound on Modular Lattices?

- Current upper bound for the number of all lattices up to isomorphism is approximately

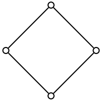
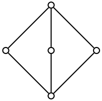
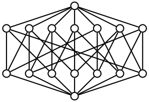
$$6.111344^{[n^{3\backslash 2}+o(n^{3\backslash 2})]} \quad \text{(Kleitman, 1980)}$$

- Upper bound for the number of all distributive lattices is

$$2.39^{n} \quad \text{(Erné, Heitzig and Reinhold, 2002).}$$

Future Work: Another Approach?

- Is it possible to generate modular lattices more efficiently with a different approach?
- Distributive lattices have been counted up to size 49 (Erné, Heitzig and Reinhold, 2002).
- Since the difference between distributive and modular lattices is that any lattice containing M_3 is non-distributive and is modular, we tried to generate all modular lattices by inserting points in the M_2 (and other M_k) sublattices.
- This worked up to size 15, but failed to generate the projective geometry lattice. When introducing it as a construction block, it worked up to size 20.
- Difficult to prove the algorithm would generate all modular lattices.

M_2  M_3  $\mathbb{Z}_2^3$ 

Future Work: Generating Other Lattices

- The algorithm for generating all lattices along with the implementation of the canonical path construction provides a tool to generate any type of lattice up to size 17 or 18.
- The algorithm can be adapted to other types of lattices.
- Some lattices of interest are:
  1. Semidistributive lattices.
  2. Almost distributive lattices.
  3. Two distributive lattices.
  4. Selfdual lattices.
- Numbers will be added to the On-Line Encyclopedia of Integer Sequences (OEIS).

References

1. R. Belohlavek and V. Vychodil, *Residuated Lattices of Size $\leq$ 12*, Order 27 (2010), 147–161.
2. R. Dedekind, *Über die von drei Moduln erzeugte Dualgruppe*, Math. Ann. 53 (1900), 371–403.
3. M. Erné, J. Heitzig and J. Reinhold, *On the number of distributive lattices*, Electron. J. Combin. 9 (2002), 23 pp.
4. J. Heitzig and J. Reinhold, *Counting Finite Lattices*, Algebra Univers. 48 (2002) 43–53.
5. D. J. Kleitman and K. J. Winston, *Combinatorial mathematics, optimal designs and their applications*, Ann. Discrete Math. 6 (1980), 243249.
6. B. D. McKay, *Isomorph-free exhaustive generation*, J. Algorithms 26 (1998) 306–324.