



Minimal representatives and the isomorphism problem for graph-like objects

Petr Vojtěchovský

December 3, 2024

University of Denver

Table of contents

1. Computational complexity: theory versus practice
2. The graph isomorphism problem
3. Canonical and minimal representatives
4. Minimal representatives of endofunctions
5. Counting groupoids up to isomorphism

Computational complexity: theory versus practice

Computational complexity

Takeaway: Theoretically best algorithms are not always practical.

$f(x)$ is $O(g(x))$

if there are M, x_0 such that $|f(x)| \leq M|g(x)|$ for all $x \geq x_0$

$f(x)$ is $\tilde{O}(g(x))$

if it is $O(g(x)(\log g(x))^c)$ for some $c \geq 0$

Primality testing: The AKS algorithm

$\text{ISPRIME}(n)$ is the problem to decide whether n is a prime number.

Theorem (Agrawal-Kayal-Saxena 2002)

$\text{ISPRIME}(n)$ is in P , with running time $\tilde{O}((\log n)^{12})$.

- In terms of the number d of digits of n , the complexity is $\tilde{O}(d^{12})$.
- Improved to $\tilde{O}(d^6)$ by Pomerance and Lenstra in 2005.

Primality testing: The Miller-Rabin test (1980)

The Miller-Rabin test is a probabilistic primality test. It can be made deterministic under certain assumptions.

Let n be an odd integer with $n - 1 = 2^s d$, d odd.

Let a be coprime to n .

Then n is a **strong probable prime to base a** if

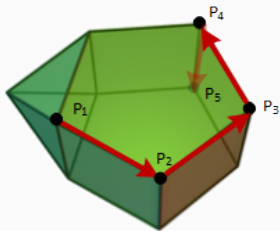
$$a^d \equiv 1 \pmod{n} \quad \text{or} \\ a^{2^r d} \equiv -1 \pmod{n} \quad \text{for some } 0 \leq r \leq s.$$

Primality testing: The Miller-Rabin test (1980)

- If n is not a strong probable prime to (some) base a , it is composite.
- A composite number is a strong probable prime for at most $1/4$ of all possible bases a .
- Running k rounds of the test with random choices of a will (incorrectly) declare a composite number to be prime with probability $\leq 4^{-k}$.
- The running time for k rounds is $O(k(\log n)^3)$.
- If ERH holds, it suffices to test fewer bases, yielding a **deterministic** algorithm with $\tilde{O}((\log n)^4)$.
- The number of bases that need to be tested for a deterministic algorithm is astonishingly small if n is small. If $n < 2^{64}$, it suffices to test $a \in \{2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37\}$.

Linear programming: The Simplex algorithm

Goal: Given $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ and $A \in \mathbb{R}_{m \times n}$, maximize $c^T x$ subject to $x \geq 0$ and $Ax \leq b$.



(wikipedia)

The **Simplex algorithm** (Dantzig 1947) travels along the vertices of the feasible region, never decreasing the value of the objective function, and eventually reaching an optimal solution.

Linear programming: The Simplex algorithm

- The key parameter of the simplex algorithm: **pivoting rules**, a procedure that decides which vertex should be visited next.
- For each popular pivoting rule it has been shown that the algorithm requires exponential time.
- With the **oracle pivoting rule** (which takes unit amount of time at any juncture and always ends up traveling along a shortest path in the end), the best bound is $O(n^{\log n})$, although $O(n)$ is suspected.
- The oracle rule is not an algorithm. The complexity of the simplex algorithm is not known.

Quasi-polynomial time algorithms

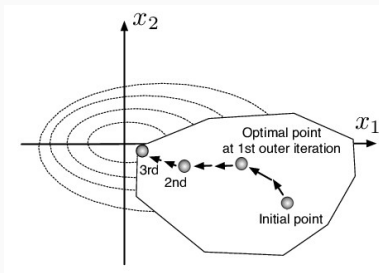
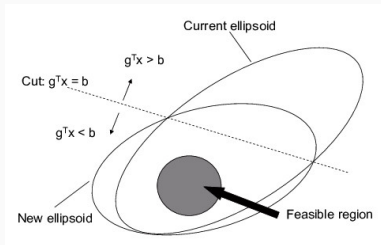
An algorithm is **quasi-polynomial** if it runs in $O(n^{(\log n)^c})$ for some constant c .

Note that $n^{\log n} \ll b^n$.

(Apply \log to get $(\log n)^2$ and $n \log b$, respectively.)

We will encounter a quasi-polynomial time algorithm shortly.

Linear programming: Two algorithms in P



(wikipedia)

Theorem (Khachiyan 1979, Karmarkar 1984)

The ellipsoid method and the interior point method are in P.

All three methods are used in practice, particularly the interior point method and the simplex method. The simplex method is probably most popular despite the fact that it has **not been proved to be in P**.

The group isomorphism problem

The **group isomorphism problem** is to decide if two groups are isomorphic.

The problem is sensitive to how the groups are given.

If the groups are given by presentations, the group isomorphism problem is undecidable. This is not surprising in view of:

Theorem (Novikov 1955)

The group word problem is undecidable in a strong sense. There is a finitely presented group G for which no algorithm can determine if two words in the generators of G represent the same element.

The group isomorphism problem for multiplication tables

If the groups are given by **multiplication tables**, the problem is decidable but its complexity is not known. It is a good candidate for a problem properly between P and NP-complete.

Naive algorithm: $O(n!n^2)$. Try all $n!$ permutations and check if they are homomorphisms.

Tarjan's 1978 algorithm: $O(n^{\log n + O(1)})$. How?

- By Lagrange's theorem, a group of order n has a generating set of size $\leq \log n$. (The bound is best possible, see \mathbb{Z}_2^n .)
- Assign f on a generating set and check if it extends to an isomorphism.

Sun improved this in 2023 to $O(n^{(\log n)^{5/6}})$ for p -groups of class 2 and exponent p . Babai: "After 50 years, we finally have something substantial to talk about."

The group isomorphism problem in practice

In practice, groups are given by presentations, or by a generating set of permutations or matrices.

There are efficient algorithms (very sensitive to the number of generators) for calculating various structural properties of the groups, such as the center, the derived subgroup, etc

Note: All FSGs are two-generated.

This allows us to work with groups much larger than the quasi-polynomial bound should permit.

But p -groups remain difficult to handle. There are no efficient invariants for distinguishing p -groups and there might not be any.

The graph isomorphism problem

Babai's result

Takeaway: Many problems can be reduced to questions about graphs. Once again, specialized algorithms, often with unknown complexity, rule.

Two graphs $G = (V, E)$ and $G' = (V', E')$ are **isomorphic** if there is a bijection $f : V \rightarrow V'$ such that for all $x, y \in V$ we have $(x, y) \in E$ if and only if $(f(x), f(y)) \in E'$.

The **graph isomorphism problem** is to decide if two graphs are isomorphic.

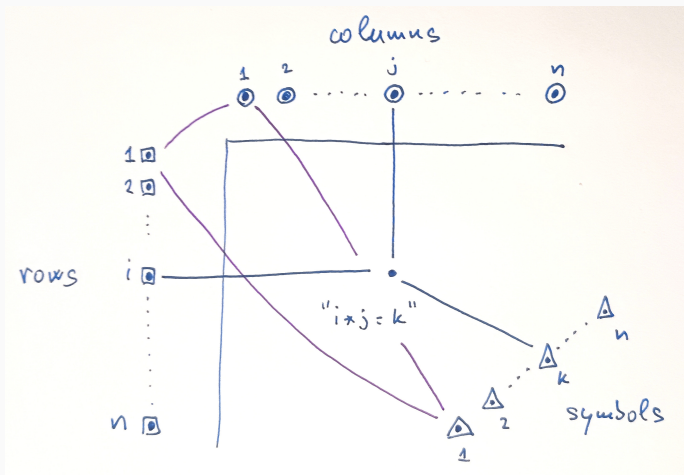
Theorem (Babai 2015, 2017)

The graph isomorphism problem is quasi-polynomial.

Note: The group isomorphism problem seems to be of key importance to the graph isomorphism problem.

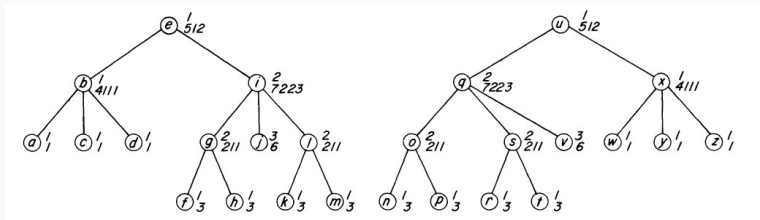
Reduction of group IP to graph IP

The group isomorphism problem (and more) is reducible to the graph isomorphism problem.



Planar graph isomorphism problem is in P

Edmonds proved that trees can be canonically labeled (see the next Section) in polynomial time.



(Colbourn and Booth)

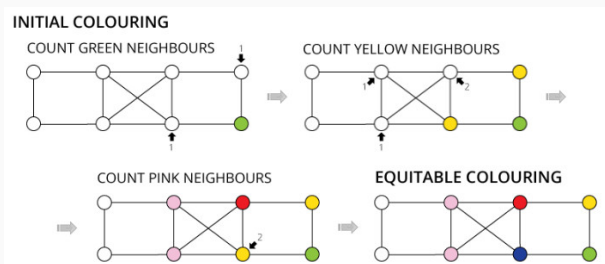
Theorem (Colbourn and Booth 1981)

Planar graphs can be canonically labeled in polynomial time.

They did not implement their algorithm (and I am not sure if it has been implemented).

Graph IP in practice: Canonical labelings in nauty and Traces

nauty and Traces (McKay and Piperno, 2014–2024) is a state of the art software for graph isomorphisms. It is based on **canonical labelings**, which are in turn based on **equitable colorings**.



(nauty and Traces manual)

Special graphs that require exponential time in nauty and Traces are known but do not typically appear.

Canonical and minimal representatives

Canonical and minimal representatives

Let G be an object and $[G]$ its equivalence class (say its isomorphism class).

A *canonical labeling* is a function $c : G \mapsto c(G) \in [G]$ that is constant on every $[G]$. Then $c(G)$ is a **canonical representative** of $[G]$.

This solves the equivalence problem: $G \equiv H$ iff $c(G) = c(H)$.

If the objects are linearly ordered, a **minimal labeling** is a function $m : G \mapsto m(G) \in [G]$ such that $m(G) = \min\{H : H \in [G]\}$. Then $m(G)$ is a **minimal representative** of $[G]$.

Clearly, minimal representatives are canonical representatives.

There exist general algorithms for finding minimal representatives of the action of a permutation group on k -element subsets (Linton, Jefferson et al).

A discouraging min rep example from graph theory

Identify a graph with its incidence matrix. Order incidence matrices lexicographically.

Problem: Find the minimal incidence matrix of a graph under the action of permuting the rows and columns of the incidence matrix.

Theorem (Crawford et al 1996)
The above problem is NP-complete.

“We trust that the above demonstration will discourage finding lex-leading-incidence-matrices as an approach to finding canonical forms for graphs and, thereby, to graph isomorphism.”

Why minimal representatives?

It is a **self-organizing principle**. Many general constructions were found by staring long enough at small minimal representatives.

It is a **space-saving measure** for extensive libraries of objects (due to a large overlap between consecutive members).

A silly example: Minimal (greedy) latin squares.

1	2	3		1	2	3		1	2	3	4
2	1	?		2	3	1		2	1	4	3
?	?	?		3	1	2		3	4	1	2
								4	3	2	1

Exercise: Call two lists equivalent if they coincide as multisets. Minimal representative = sorted list. Is there a better algorithm for equivalence?

Minimal representatives of endofunctions

Endofunctions and the transformation monoid

Takeaway: You usually win if you can convert the problem to a graph isomorphism problem for planar graphs.

An **endofunction** is a function $t : X \rightarrow X$.

All endofunctions on X form a monoid under composition, the **full transformation monoid** T_X of X . The identity function on X is the identity element of T_X .

If $|X| = n$ then $|T_X| = n^n$.

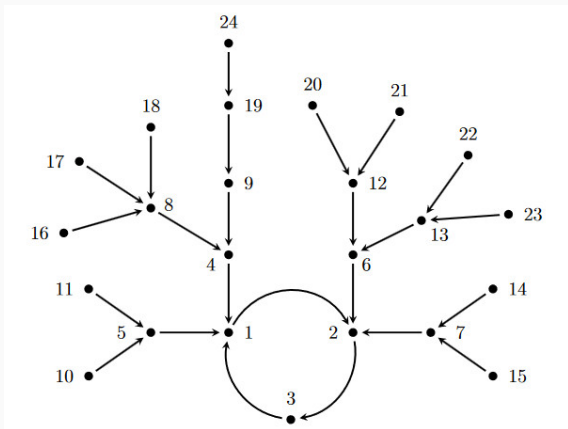
For $X = \{1, \dots, n\}$, identify $t : X \rightarrow X$ with $(t(1), t(2), \dots, t(n))$.

Order T_X lexicographically.

Labeled functional digraphs

A digraph $G = (V, E)$ is a **functional digraph** if for every $u \in V$ there is a unique $v \in V$ such that $u \rightarrow v$ is in E .

Example: A connected component of a labeled functional digraph:



Conjugation versus relabeling

Given an endofunction $t : X \rightarrow X$, $G(t) = (X, E)$ with $E = \{(x, t(x)) : x \in X\}$ is a functional digraph.

Given a functional digraph $G = (X, E)$, define $t_G : X \rightarrow X$ by $(x, t_G(x)) \in E$.

These constructions are inverse to each other.

The symmetric group S_X acts on T_X by conjugation:
 $f : t \mapsto ftf^{-1}$.

The effect of conjugation on $G(t)$:

If $t(x) = y$ then $ftf^{-1}(f(x)) = ft(x) = f(y)$,
so $(x, y) \in G(t)$ iff $(f(x), f(y)) \in G(ftf^{-1})$.

To obtain $G(ftf^{-1})$, relabel the vertices of $G(t)$ according to f .

Minimal representative of a conjugacy class

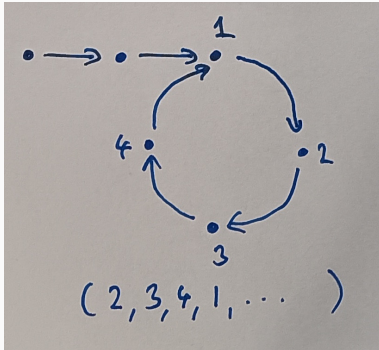
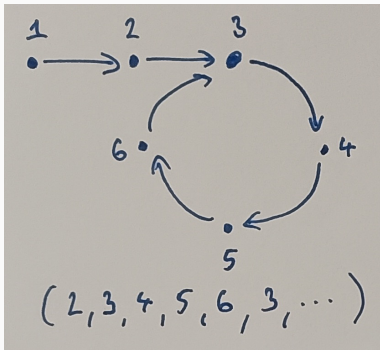
Task: Find the minimal representative in the S_X -conjugacy class of $t \in T_X$.

Equivalent task: Find a minimal labeling of the corresponding functional digraph $G(t)$.

It is reasonable to expect an efficient algorithm since $G(t)$ is planar.

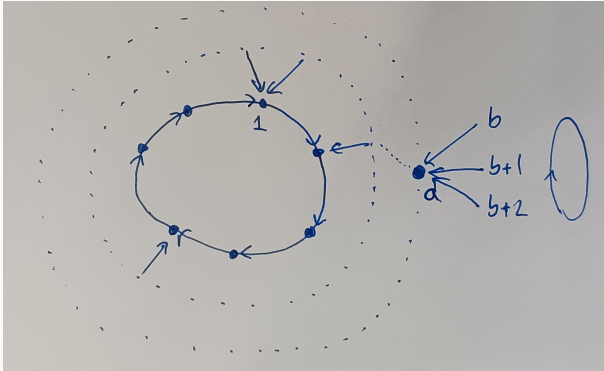
1 is in a cycle

The smallest label must be in a cycle:



Consecutive layers

Layers are populated consecutively:

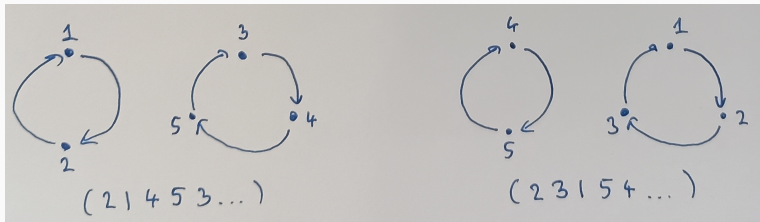


(a = first unfinished vertex, b = first unused label)

Ordering connected components

Suppose that every connected component has been minimally labeled. How should the components be ordered?

Shorter cycles come first:

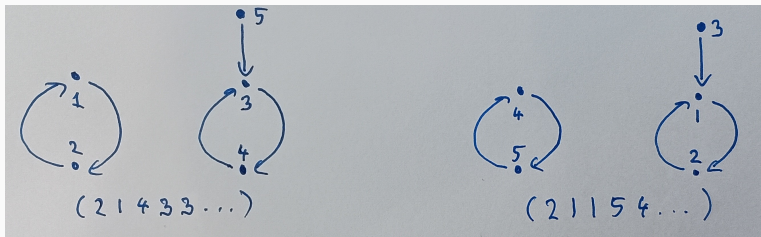


$$(2, 1, 4, 5, 3, \dots) < (2, 3, 1, 5, 4, \dots)$$

Makes sense: $(2, 1) < (2, 3, 1)$.

Ordering connected components

Cycles of the same length:



$$(2, 1, 4, 3, 3, \dots) > (2, 1, 1, 5, 4, \dots)$$

Yet $(2, 1) < (2, 1, 1)$. What's going on?

Well, $(2, 1) < (2, 1, 1)$ because $(2, 1)$ is a proper prefix of $(2, 1, 1)$.

Cycle length comparison =



Ordering connected components: The spelling bee ordering

Answer: Use the **spelling bee ordering**. This is just like the lexicographic ordering except that proper prefixes come later.

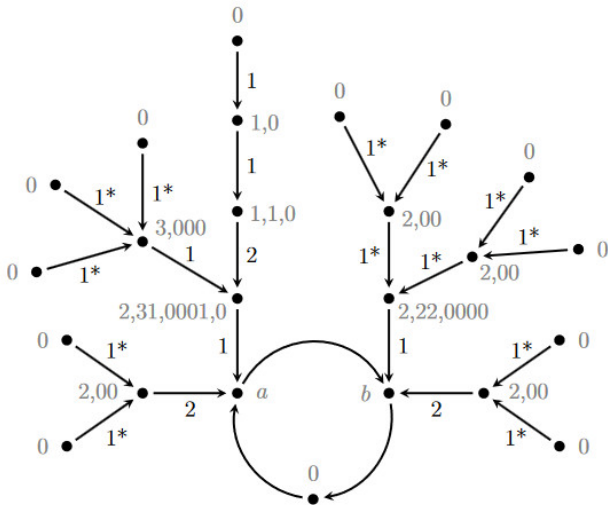
Example: aardvark < antelope < ant < a < beer < bee



(2024 Scripps National Spelling Bee finalists, photo by Craig Hudson)

Structural labels

We can now assume WLOG that $G(t)$ is connected. It remains to determine the position of 1 , and the order in each layer.



Populating the layers: According to the structural edge labels.

Positioning 1 on the cycle:

- We can try all possibilities.
- Better: Pick a starting position, keep going around the cycle, collecting the first entry from each structural vertex label in every round. Position **1** where the resulting concatenated structural label is largest. In particular, the indegree at **1** must be maximal on the cycle.

Note: Carefully exploring ties among structural edge labels leads to a description of the automorphism group of $G(t)$.

Computational complexity of the algorithm

- Constructing $G(t)$: $O(n)$.
- Calculating structural labels: $O(n^2)$.
- Determining the location of $\mathbf{1}$ in each connected component: $O(n^2)$ total.
- Ordering the components: $O(n^2)$.

Theorem (Mitchell, Mukherjee and V 2024)

Finding the minimal representative of $t \in T_X$ in its S_X -conjugacy class and finding a suitable conjugating permutation can be done in $O(|X|^2)$.

Note: A better approach could give $O(n \log n)$.

Counting groupoids up to isomorphism

Diagonals

Takeaway: There are too many groupoids. Group actions and invariant domains can help.

In a multiplication table of a groupoid $(X, *)$, the **diagonal** can be seen as an endofunction

$$d_* : X \rightarrow X, \quad d_*(x) = x * x.$$

If $f : (X, *) \rightarrow (X, \circ)$ is an isomorphism, we have

$$x \circ y = f(f^{-1}(x) * f^{-1}(y))$$

and thus $d_\circ(x) = x \circ x = f(d_* f^{-1}(x))$, so $d_\circ = f d_* f^{-1}$.

Diagonals first in a lexmin model

Traditionally, the **lexicographically minimal multiplication table** is understood as the lexicographically smallest vector obtained by concatenating the rows. But this is only a convention.

We can reorder the cells so that the diagonal cells are considered first. Then, searching for a **lexmin** copy of $(X, *)$, we must first find an isomorphic copy of $(X, *)$ with the smallest diagonal. This is precisely the problem we just solved.

Janota, Chow, Araújo, Codish and V 2024 implemented a SAT solver-based algorithm for finding lexmin groupoids.

Basic setup:

$f : (X, \cdot) \rightarrow (X, *)$, $x_{i \rightarrow j}$ is a variable meaning $f(i) = j$

Constraints:

- $B(X) = \{\sum_{j \in X} x_{j \rightarrow i} = \sum_{j \in X} x_{i \rightarrow j} = 1 : i \in X\}$
- For all $r, c, v \in X$

$$E(r * c = v) = \{(x_{i \rightarrow r} \wedge x_{j \rightarrow c}) \Rightarrow x_{i \cdot j \rightarrow v} : i, j \in X\}.$$

Basic algorithm:

```
A := empty set
for r,c in [1..n] do
  v := 1
  while not SAT( B(X) and E(r*c=v) ) do
    v := v+1
  A := Append(A, r* c=v)
```

With many improvements (such as focusing on the diagonal first), the algorithm performs quite well and is in some situations competitive with nauty when used as an isomorphism test.

Mapping types

For a set X of a fixed cardinality n , the minimal representatives of S_X -conjugacy classes are known as **mapping types**.

Theorem (de Bruijn 1972)

The number of mapping types on a set of cardinality n is

$$M_n = \sum \prod_{i=1}^{\infty} \left(\sum_{j|i} (jn_j) \right)^{n_i} i^{-n_i} / (n_i!),$$

where the first summation runs over all (n_1, n_2, \dots) such that $n_1 + 2n_2 + 3n_3 + \dots = n$. (Good luck!)

We get $M_1 = 1$, $M_2 = 3$, $M_3 = 7$, \dots , $M_{10} = 7318$. Values up to M_{1000} have been calculated.

Right self-distributive groupoids

In 1997, Ježek used the idea of mapping types to enumerate **right self-distributive groupoids**, that is, groupoids $(X, *)$ satisfying

$$(x * y) * z = (x * z) * (y * z).$$

Theorem (Ježek 1997)

The number of right self-distributive groupoids of order n is

n	2	3	4	5	6
<i>absolutely</i>	9	224	14067	3717524	?
<i>up to isomorphism</i>	6	48	720	33425	35527077*

* = corrected by us, Ježek reported 35527485

From the master himself ...

```
int Verify(I){ int i,j,k,a,b,c,d,e,p,q,Z=0;
for(i=0;i<n;i++)for(j=0;j<n;j++){
    c=A[n*i+j];if(c>=0)for(k=0;k<n;k++){
        d=A[n*i+k];a=A[n*j+k];if(a>=0&&d>=0){
            p=n*i+a;q=n*c+d;b=A[p];e=A[q];
            if(b>=0&&e<0){A[q]=b;B[q]=I;Z=1;}
            else if(b<0&&e>=0){A[p]=e;B[p]=I;Z=1;}
            else if(b>=0&&b!=e)return -1;}}
return Z;}
```

Right self-distributive groupoids revisited

A common phenomenon in enumerative combinatorics: **Anybody can calculate a_n , it takes a lot of effort to do a_{n+1} , and a_{n+2} is presently impossible.**

Mukherjee and V managed to count the number of right self-distributive groupoids of order **6** and **7** absolutely:

$$a_6 = 25488943921 \text{ (30 min)}, \quad a_7 = 3021268037534480 \text{ (1 month)}$$

We are double-checking a_7 .

Main ideas:

- S_n acts on the n^{n^2} space of all multiplication tables.
- Let it act on partial tables (partial domains).
- Subdivide the table: the diagonal, the rest of the first row, etc.
- Apply the orbit-stabilizer theorem iteratively.
- Finish some cases by hand, e.g., the diagonal **1111223**.

Sets of mutually right distributive groupoids

We were led into this due to our work in **knot theory**.

The third Reidemeister move naturally corresponds to the right self-distributive law.

There are homology theories (due to Przytycki) based on a set S of mutually right distributive groupoids, that is, $(x * y) \circ z = (x \circ z) * (y \circ z)$ for any two operations in the set.

How big can S get?

Sets of mutually right distributive groupoids: A construction

Fix $c \leq k \leq n$ and consider any groupoid such that





$$x * y = \begin{cases} c, & \text{if } x \leq k, \\ \in \{1, \dots, k\}, & \text{otherwise.} \end{cases}$$





There are $k^{n(n-k)}$ such groupoids.






Example: $c = 2$, $k = 3$, $n = 5$:




2	2	2	2	2
2	2	2	2	2
2	2	2	2	2
1	3	2	3	1
1	1	1	3	1

Any two such groupoids right distribute over each other.

-  L. Babai, *Graph isomorphism in quasipolynomial time [extended abstract]*, STOC'16–Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, 684–697, ACM, New York, 2016.
-  C.J. Colbourn and K.S. Booth, *Linear time automorphism algorithms for trees, interval graphs, and planar graphs*, SIAM J. Comput. **10** (1981), no. **1**, 203–225.
-  J. Crawford, M. Ginsberg, E. Luks and A. Roy, *Symmetry-breaking predicates for search problems*, Proceedings 5th International Conference on Principles of Knowledge Representation and Reasoning, 1996, 148–159.
-  N.G. de Bruijn, *Enumeration of mapping patterns*, J. Combinatorial Theory Ser. A **12** (1972), 14–20.

-  The GAP Group, *GAP – Groups, Algorithms, and Programming, Version 4.13.1*; 2024, <https://www.gap-system.org>.
-  C. Jefferson, E. Jonauskyste, M. Pfeiffer and R. Waldecker, *Minimal and canonical images*, *J. Algebra* **521** (2019), 481–506.
-  C. Jefferson, M. Pfeiffer, R. Waldecker and E. Jonauskyste, *images*, *Minimal and Canonical Images, Version 1.3.2* (2024), GAP package, <https://gap-packages.github.io/images/>
-  M. Janota, C. Chow, J. Araújo, M. Codish and P. Vojtěchovský, *SAT-based techniques for lexicographically smallest finite models*, *Proceedings of the AAI Conference on Artificial Intelligence* **38(8)** (2024), 8048–8056.

-  J. Ježek, *Enumerating left distributive groupoids*, Czechoslovak Math. J. **47(122)** (1997), no. **4**, 717–727.
-  E. Klarreich, *Algorithmic Advance: the Group Isomorphism Problem*, Communications of the ACM **67** (February 2024), no. **2**, 9–11.
-  S. Linton, *Finding the smallest image of a set*, in ISSAC '04: Proceedings of the 2004 international symposium on symbolic and algebraic computation, July 2004, 229–234.
-  B.D. McKay and A. Piperno, *Practical graph isomorphism, II*, J. Symbolic Comput. **60** (2014), 94–112.
-  B.D. McKay and A. Piperno, nautyTraces, software distribution webpage <http://cs.anu.edu.au/~bdm/nauty> and <http://pallini.di.uniroma1.it>.

-  J.D. Mitchell, S. Mukherjee and P. Vojtěchovský, *Minimal representatives of endofunctions*, to appear in Semigroup Forum.
-  L.H. Soicher, GRAPE, graph algorithms using permutation groups, version 4.9.0,
<https://gap-packages.github.io/grape>, December 2022, refereed GAP package.
-  Sun, X., *Faster isomorphism for p -groups of class 2 and exponent p* , <https://arxiv.org/abs/2303.15412>