Can AI help us discover interesting mathematics?

Joint Mathematics Meetings, Washington DC, 2026.

Geordie Williamson

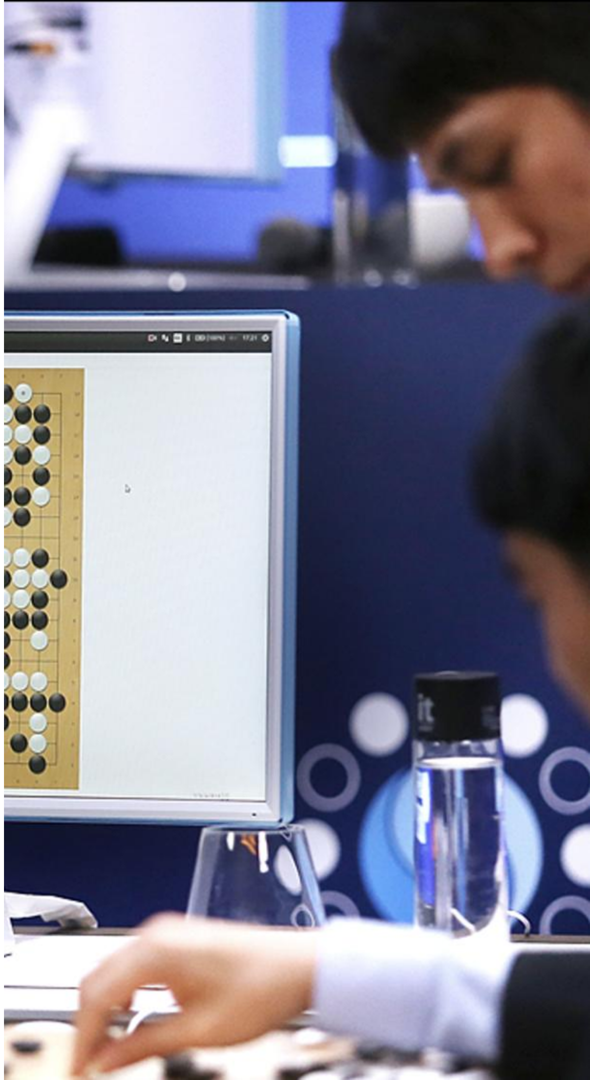
THE UNIVERSITY OF SYDNEY

Mathematical Research Institute

The last decade of AI

- *2016:* AlphaGo (AI system beats world leading Go player)
- *2020:* AlphaFold (protein structure prediction, Nobel prize 2024)
- *2021*: Image generation (Dall-E...)
- *2022:* ChatGPT + LLMs.
  Broad adoption by society, major impacts on education.
- *2024:* IMO Silver (AlphaProof)
- *2025*: IMO Gold (several labs)

What is a *mathematician* to do?

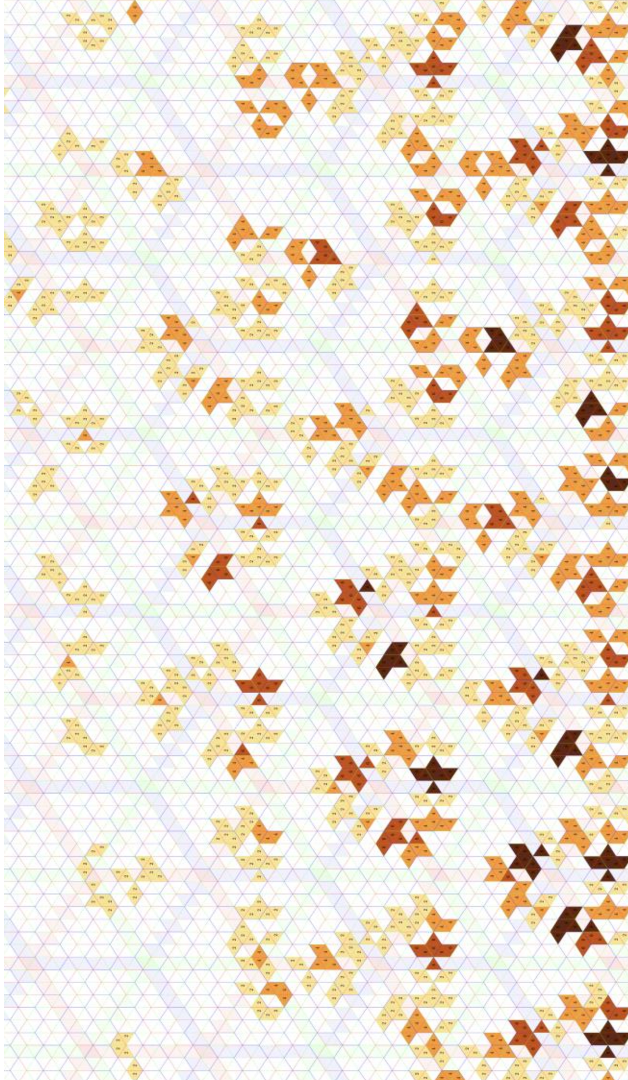


My philosophy

We are making a mistake if our only reaction to these developments is ignorance and fear.

Two millennia of mathematics has been a journey full of surprises.
Many surprises await us!
New tools with great potential deserve our attention.

AI has major implications for math and the world more broadly.
This discussion is of great importance, and needs our engagement.
(I will return to this at the end of this lecture.)



AI and mathematics

Neural networks are a major tool in applied mathematics.

Major recent progress with development and adoption of formal proof assistants (*Lean*, *Isabelle,…*).
Here the computer checks that a proof is correct.

Automated proof *remains a major challenge*.
Major focus of several AI startups.

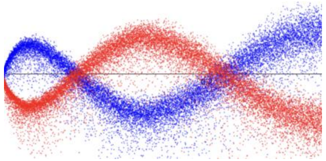
In 2025 we began to see some impacts of LLMs on research level mathematics.

Mathematics and code generation are currently serving as an important benchmark for AI systems.

```
@[simp] lemma id_tensor_comp (f : W ⟶ X) (g : X ⟶ Y) :
  (𝟙 Z) ⊗ (f ≫ g) = (𝟙 Z ⊗ f) ≫ (𝟙 Z ⊗ g) :=
by { rw ←tensor_comp, simp }

@[simp] lemma id_tensor_comp_tensor_id (f : W ⟶ X) (g :
  ((𝟙 Y) ⊗ f) ≫ (g ⊗ (𝟙 X)) = g ⊗ f :=
by { rw [←tensor_comp], simp }

@[simp] lemma tensor_id_comp_id_tensor (f : W ⟶ X) (g :
  (g ⊗ (𝟙 W)) ≫ ((𝟙 Z) ⊗ f) = g ⊗ f :=
by { rw [←tensor_comp], simp }

lemma left_unitor_inv_naturality {X X' : C} (f : X ⟶ X')
  f ≫ (λ_ X').inv = (λ_ X).inv ≫ (𝟙 _ ⊗ f) :=
begin
  apply (cancel_mono (λ_ X').hom).1,
  simp only [assoc, comp_id, iso.inv_hom_id],
  rw [left_unitor_naturality, ←category.assoc, iso.inv_hom
end

lemma right_unitor_inv_naturality {X X' : C} (f : X ⟶ X'
  f ≫ (ρ_ X').inv = (ρ_ X).inv ≫ (f ⊗ 𝟙 _) :=
begin
  apply (cancel_mono (ρ_ X').hom).1,
  simp only [assoc, comp_id, iso.inv_hom_id],
  rw [right_unitor_naturality, ←category.assoc, iso.inv_ho
end

@[simp] lemma tensor_left_iff
  {X Y : C} (f g : X ⟶ Y) :
  ((𝟙 (𝟙_ C)) ⊗ f = (𝟙 (𝟙_ C)) ⊗ g) ↔ (f = g) :=
begin
```

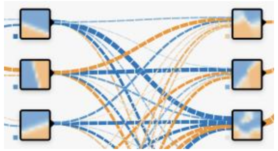
Plan of colloquium lectures

The three lectures are largely independent and (hopefully) broadly accessible.



Sunday

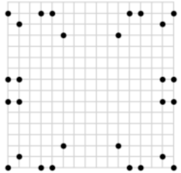
Can AI help us discover interesting mathematics?



Monday

Mental models for neural networks

(non expert introduction)



Tuesday

Searching for interesting mathematical objects with neural networks

(my current obsession)

Plan of today’s talk

- A few slides about machine learning and neural networks.
- Three examples of AI assisted mathematical discovery:

  i) Bruhat graphs and combinatorial invariance

  ii) Elliptic curves and murmurations

  iii) Bruhat graphs and hypercubes via AlphaEvolve

  Note: i) and ii) are “low resource AI”, the last uses LLMs.
- Some reflections on mathematics and AI.

Machine learning and neural networks

Machine learning in nutshell

Perceptual functions (vision, speech,...) involve *very large dimensions*.

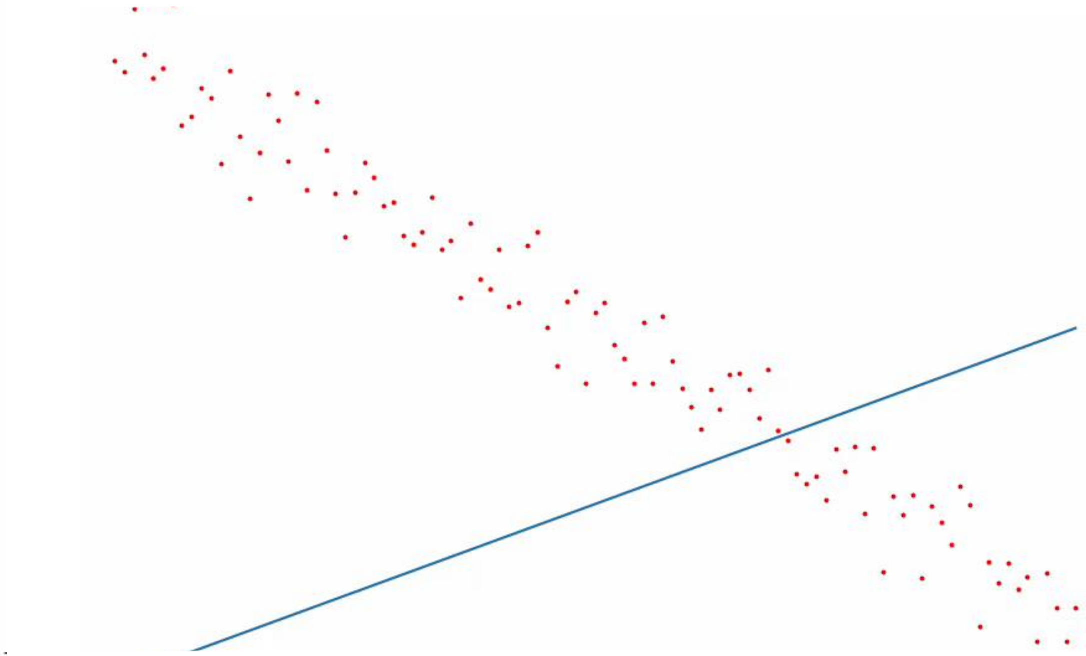
Example: The image of tiger is a vector in $\mathbb{R}^n$, where $n = 2 \times 10^6$.

There are powerful simple ways of dealing with high-dimensional data (regression, PCA,...).

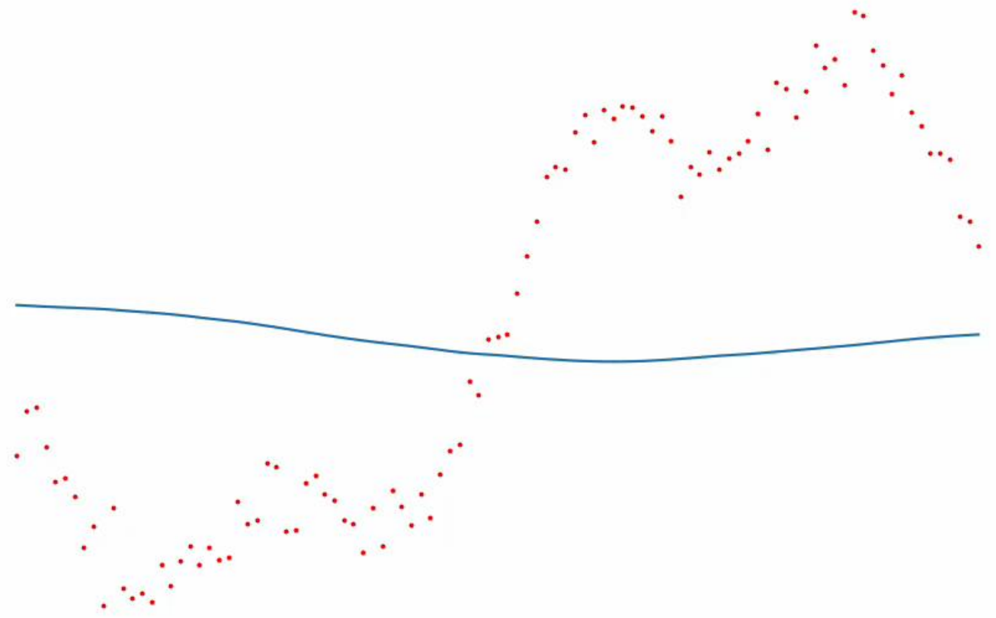
Neural networks provide powerful non-linear function approximators in high dimension.


Amélie Rec/Pexels

The line of best fit



Neural network: a powerful approximator



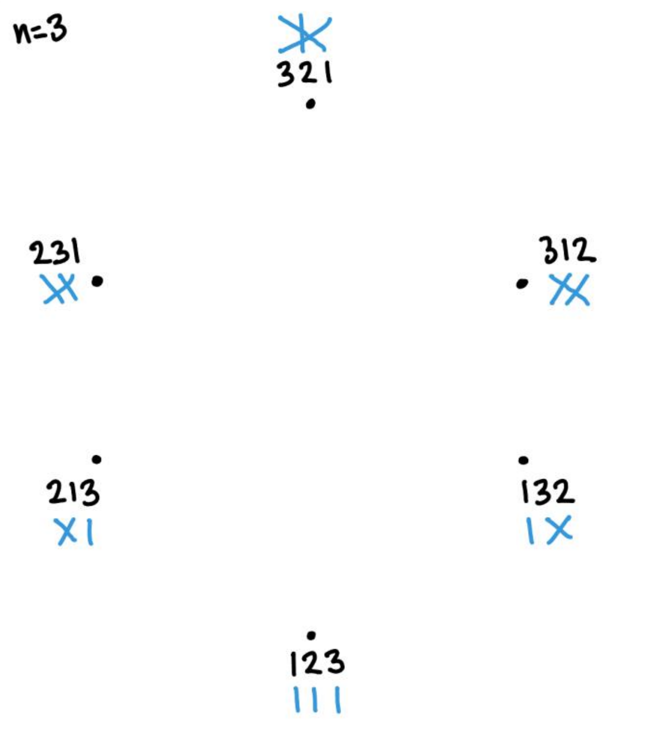
Example 1:

Bruhat graphs, Kazhdan-Lusztig polynomials and combinatorial invariance

The Bruhat graph

Given a permutation $x \in S_n$ its *length* is

$$\ell(x) = \#\{\, i < j \mid x(i) > x(j) \,\}.$$

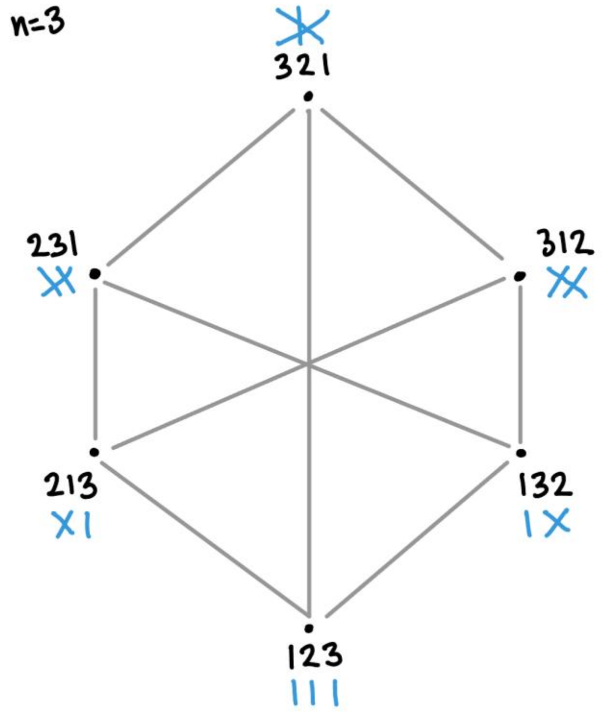


The Bruhat graph

Given a permutation $x \in S_n$ its *length* is

$$\ell(x) = \#\{ i < j \mid x(i) > x(j) \}.$$

The Bruhat graph is the Cayley graph of S_n with generators the transpositions (i, j).



The Bruhat graph

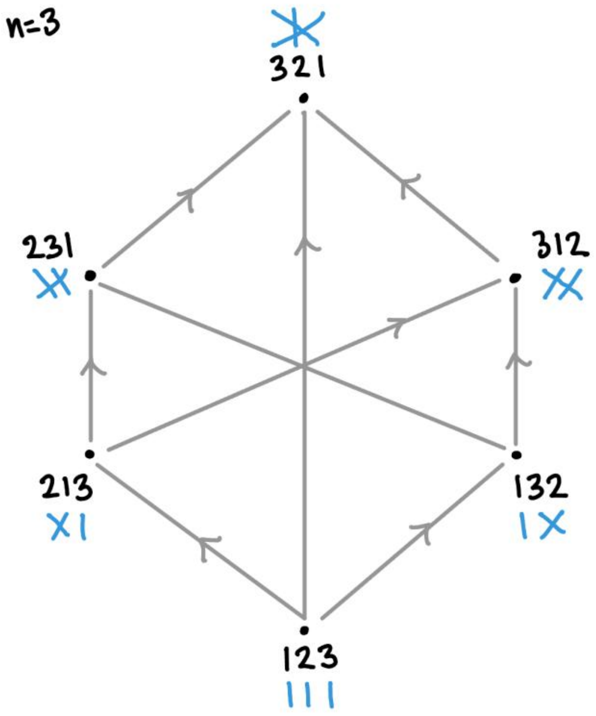
Given a permutation $x \in S_n$ its *length* is

$$\ell(x) = \#\{\, i < j \mid x(i) > x(j) \,\}.$$

The Bruhat graph is the Cayley graph of S_n with generators the transpositions (i, j).

We direct its edges using the length function.

The *Bruhat order* is $x \leq y$ if there exists a path from x to y in the Bruhat graph.



The Bruhat graph

Given a permutation $x \in S_n$ its *length* is

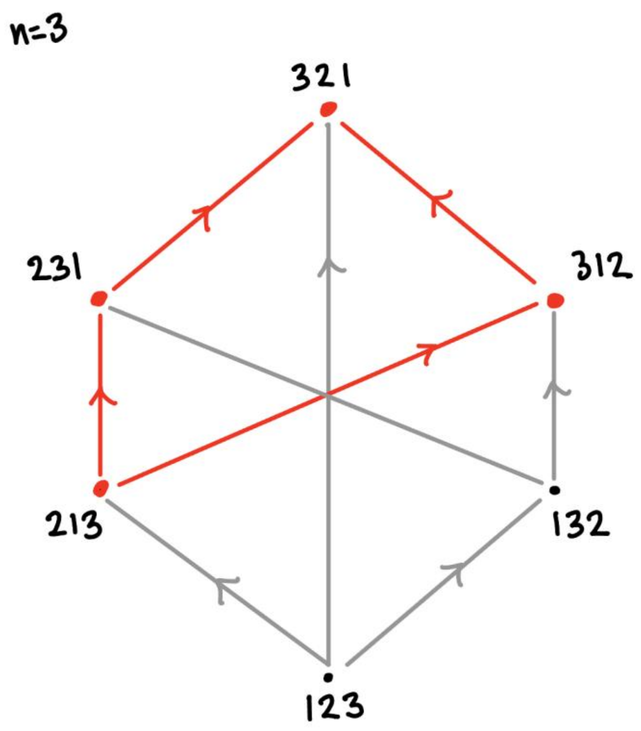
$$\ell(x) = \#\{\, i < j \mid x(i) > x(j) \,\}.$$

The Bruhat graph is the Cayley graph of S_n with generators the transpositions (i, j).

We direct its edges using the length function.

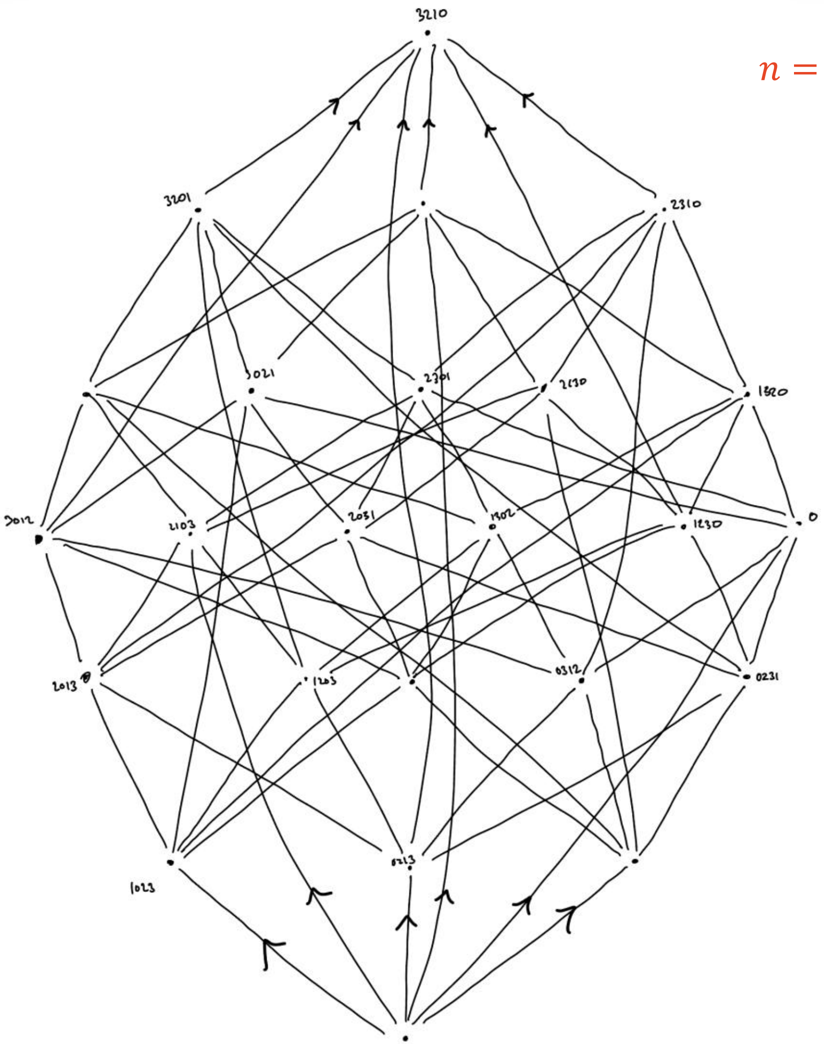
The *Bruhat order* is $x \le y$ if there exists a path from x to y in the Bruhat graph.

For $x \le y$ get Bruhat graph of interval $[x, y]$.



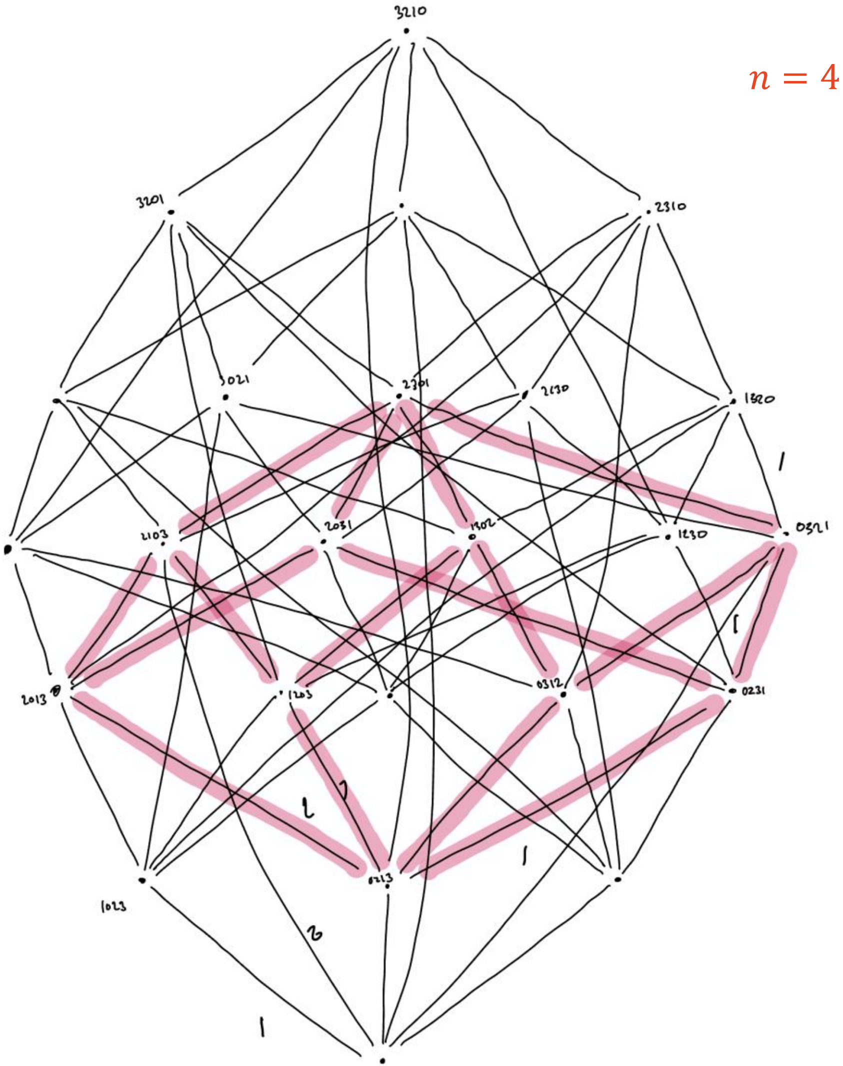
The Bruhat graph

$n = 4$



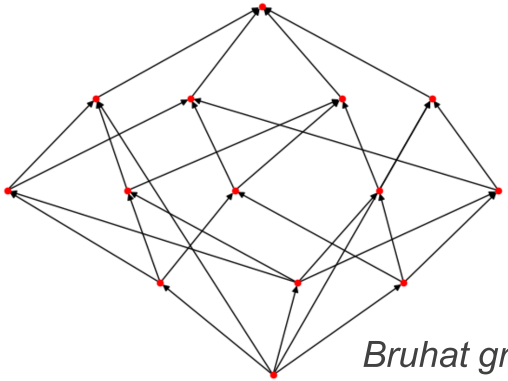
The Bruhat graph

$n = 4$



The combinatorial invariance conjecture (Dyer, Lusztig 1980s)

x, y (pair of permutations, $x \leq y$)



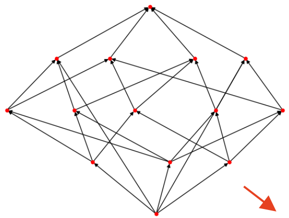
Bruhat graph (1950s)

Strong combinatorial invariance conjecture

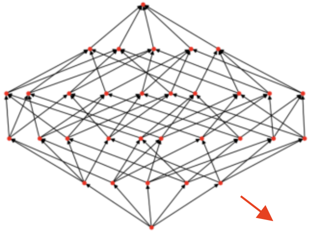
$$1+3q+q^2$$

Kazhdan-Lusztig polynomial (1979)

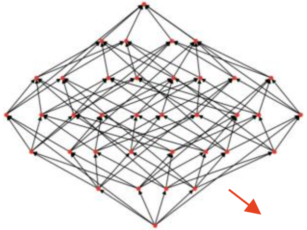
How to get an idea from *many* examples?



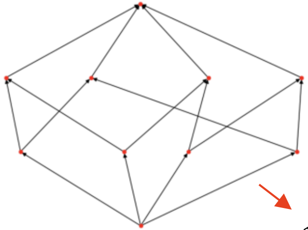
$1 + 2q$



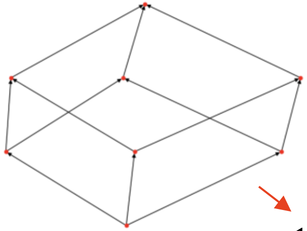
$1 + 2q$



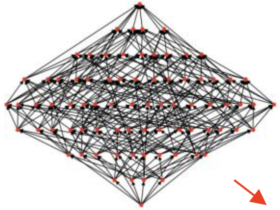
$1 + q + 2q^2$



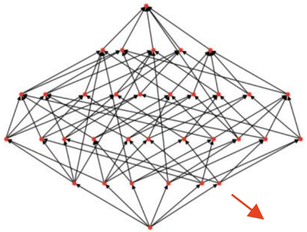
$1 + q$



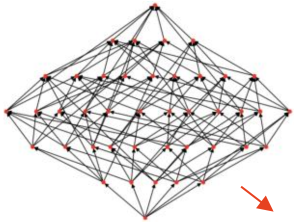
$1 + 2q$



$1 + 2q^2 + q^3$

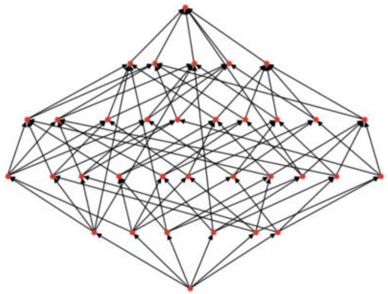


$1 + q + 2q^2$



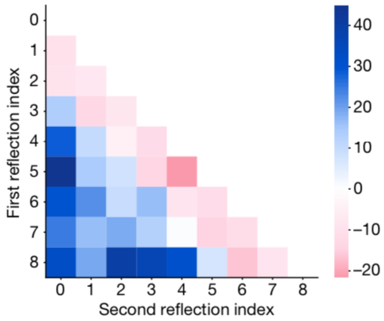
$1 + 3q$

How to get an idea from *many* examples?



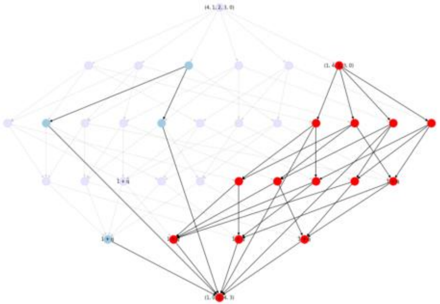
We train a *graph neural network* to predict Kazhdan-Lusztig polynomials from the Bruhat graph.

~ *two weeks*



We inspect the *gradients* of the neural network. Certain edges are much more important than others for prediction.

~ *two months*



This led to a *new formula* for Kazhdan-Lusztig polynomials, and a new approach to the combinatorial invariance conjecture.

~ *seven months*

Much activity!

2021

(Rep theory + knot theory results)

PARABOLIC RECURSIONS FOR KAZHDAN-LUSZTIG POLYNOMIALS AND THE HYPERCUBE DECOMPOSITION

MAXIM GUREVICH AND CHUIJIA WANG

2023

Kazhdan–Lusztig R-polynomials, combinatorial invariance, and hypercube decompositions

Francesco Brenti[1] · Mario Marietti[2]

2024

THE BBDVW CONJECTURE FOR KAZHDAN–LUSZTIG POLYNOMIALS OF LOWER INTERVALS

GRANT T. BARKLEY AND CHRISTIAN GAETZ

2025

COMBINATORIAL INVARIANCE FOR KAZHDAN–LUSZTIG R-POLYNOMIALS OF ELEMENTARY INTERVALS

GRANT T. BARKLEY AND CHRISTIAN GAETZ

2025

Example 2:
Elliptic curves and murmurations

The rank of an elliptic curve

Consider an elliptic curve over $\mathbb{Q}$:

$$E:\ y^2 = x^3 + Ax + B, \qquad\qquad A, B \in \mathbb{Z}$$

The rational solutions $E(\mathbb{Q})$ of this equation form a finitely-generated abelian group (Mordell-Weil). Thus

$$E(\mathbb{Q}) = E(\mathbb{Q})_{tors} \oplus \mathbb{Z}^{rank(E)}.$$

- Determination of $E(\mathbb{Q})_{tors}$ is “easy”.
- Determination of $rank(E)$ is a *major problem* in Diophantine geometry.

The rank of an elliptic curve

For any prime p we can reduce the equation

$$E\colon\ y^2 = x^3 + Ax + B, \qquad A, B \in \mathbb{Z}$$

modulo p and count solutions to get a number $N_p(E)$.

This is “easy”.

$$(1) \qquad \Gamma: y^2 = x^3 - Ax - B$$

where A, B are rational. In particular, one hopes that if for most p the curve (1) has unusually many points in the finite field with p elements, then it will have a lot of rational points.

Birch and Swinnerton-Dyer, *Notes on elliptic curves I, Crelle’s journal, 1963.*

Elliptic curves via their features

It is tempting to think of E as represented in a large vector space by the “Frobenius traces”:

$$a_p(E) = (p + 1) - N_p(E)$$

Thus, an elliptic curve becomes encoded by an infinite sequence of integers indexed by the primes:

$$[-1, 0, 0, -2, -3, 0, 0, -2, 3, 6, 4, -2, 0, 8, 9, 6, -9, 2, -8, 0, -5, 14, 15, \dots]$$

By considering the large-scale structure of this dataset, He, Lee, Oliver and Pozdnyakov were led to a remarkable observation.

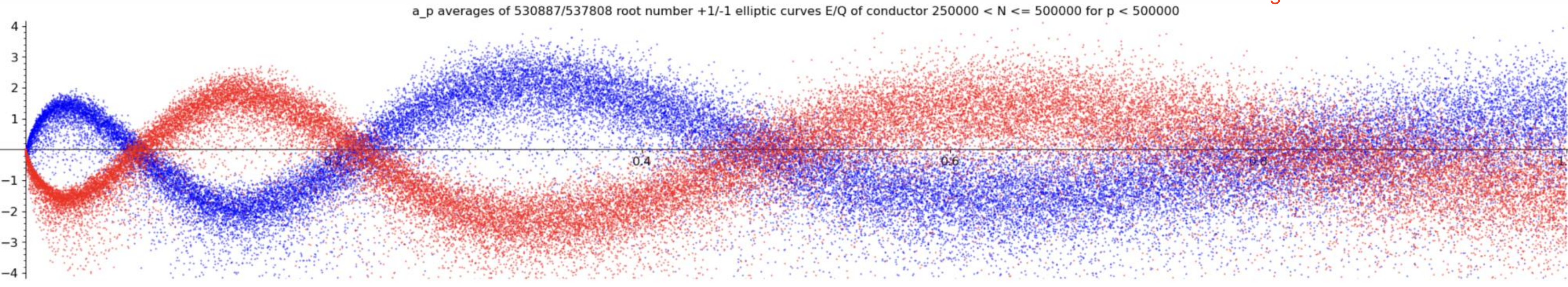
Murmurations

Let $\mathcal{E}_r(N_1, N_2)$ denote the set of (isogeny classes of) elliptic curves of rank r and *conductor* between N_1 and N_2.

They observed striking oscillations in the average values of the $a_p(E)$:

$$f_r(p) = \frac{1}{\#\mathcal{E}_r(N_1, N_2)} \sum_{E \in \mathcal{E}_r(N_1,N_2)} a_p(E)$$

Image Credit: Andrew Sutherland



Much activity!

LETTER TO DREW SUTHERLAND AND NINA ZUBRILINA ON MURMURATIONS AND ROOT NUMBERS. AUGUST 2023 ①

Sarnak's letter, 2023

Murmurations of Elliptic Curves 2024

Yang-Hui He, Kyu-Hwan Lee, Thomas Oliver, Alexey Pozdnyakov

August 30, 2022

Dear Mike and Peter,

It was great to see you both at ANTS XV. I apologize for the delay in getting this letter to you. I started writing it the weekend after ANTS but then realized I wanted to include computations that could not be completed until I returned to Cambridge, which happened just last week.

Sutherland's letter, 2022

MURMURATIONS OF DIRICHLET CHARACTERS

2024

KYU-HWAN LEE*, THOMAS OLIVER, AND ALEXEY POZDNYAKOV

MURMURATIONS OF MAASS FORMS

ANDREW R. BOOKER, MIN LEE, DAVID LOWRY-DUDA, ANDREI SEYMOUR-HOWELL, AND NINA ZUBRILINA

2024

MURMURATIONS

NINA ZUBRILINA

2025

MURMURATIONS FOR ELLIPTIC CURVES ORDERED BY HEIGHT

WILL SAWIN AND ANDREW V. SUTHERLAND

2025

Nuts and bolts

- A genuinely new phenomenon in number theory,
  that could have been noticed decades ago!

- Arose from attempting to understand earlier observations by He, Lee and Oliver that ranks of elliptic curves can be learned by linear regression.

- Ordering by conductor is critical.

- The *L-functions and modular forms database* (LMFDB)
  provides easy access to enormous amounts of well-organized arithmetic data, enabling such experiments.

Example 3:

Bruhat graphs and hypercubes via AlphaEvolve

funsearch and AlphaEvolve

Suppose we are looking for some concrete mathematical object (e.g. a graph, a subset of $\mathbb{F}_p^n$, ...).

We may describe the object *explicitly* (e.g. via an adjacency matrix, or list of points, ...) or may provide a *recipe* for building the object.

Sometimes a recipe can be given by a python program.

LLMs like writing python! We can search in program space.

This is the fundamental idea behind funsearch and AlphaEvolve.

Classical local search

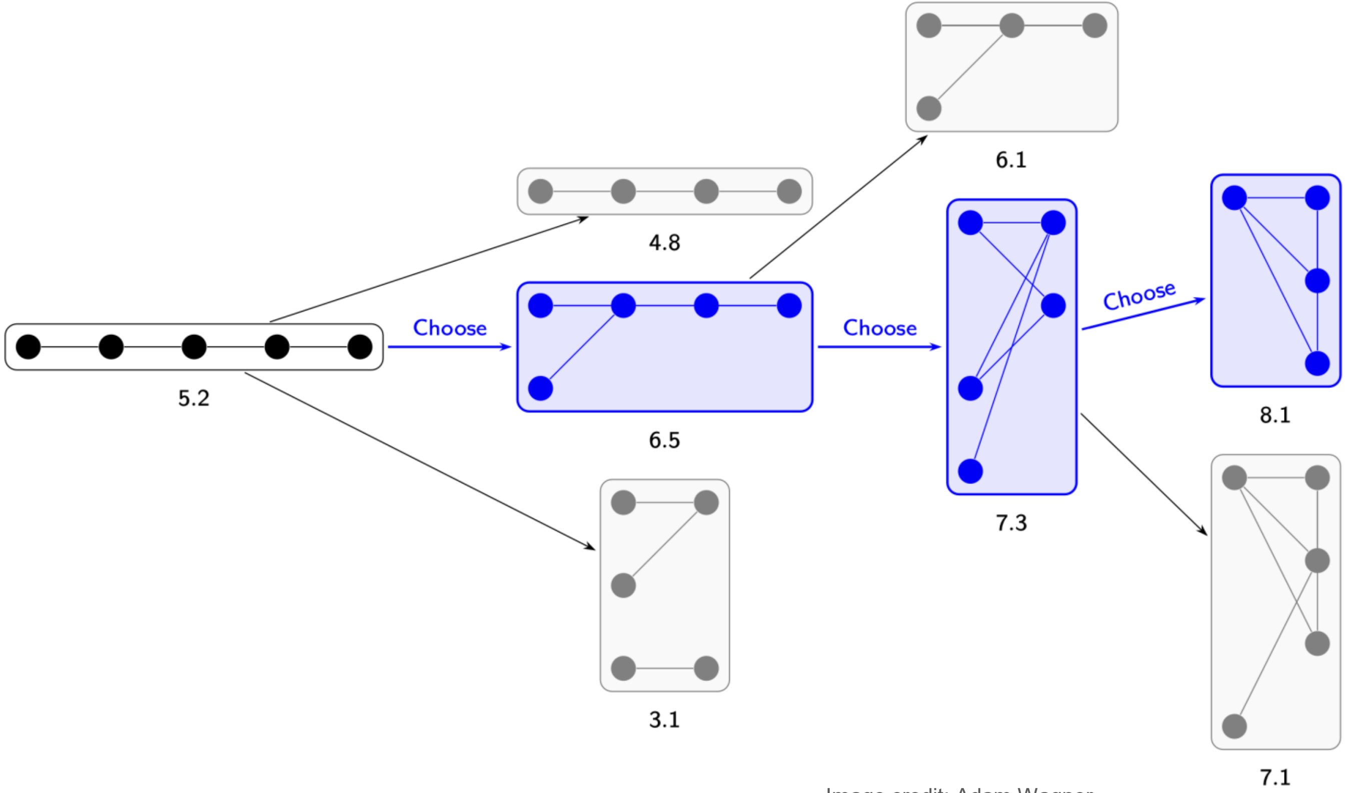


Image credit: Adam Wagner

Local search in python programs

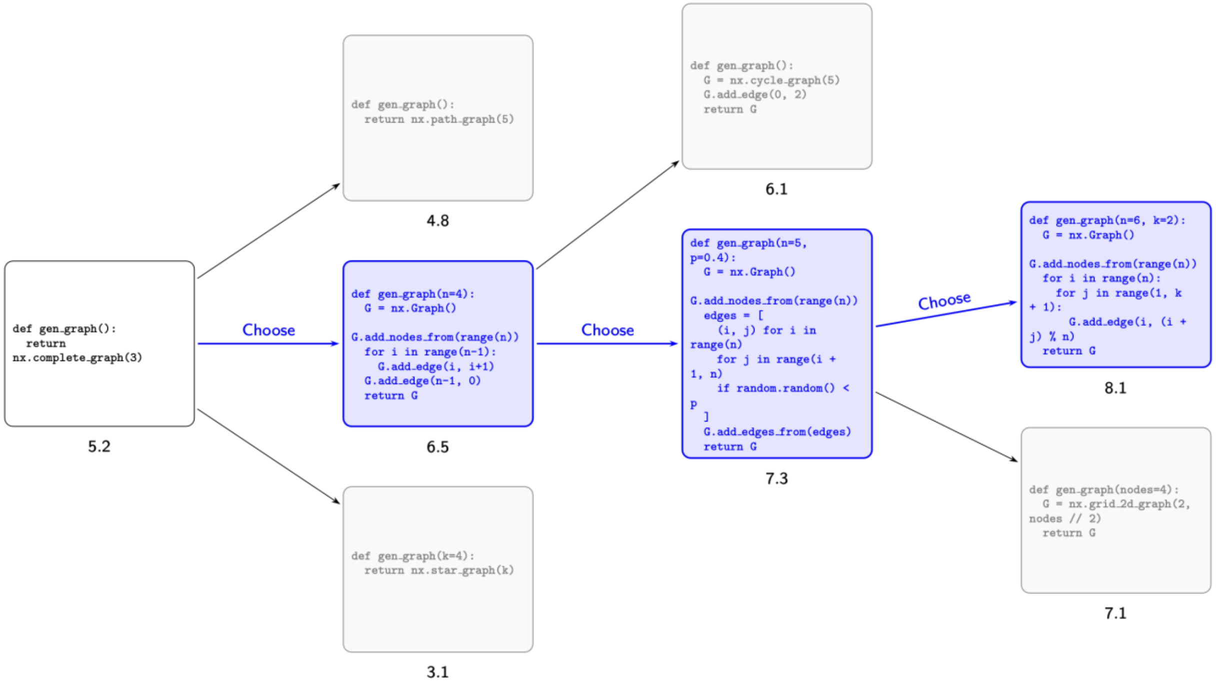
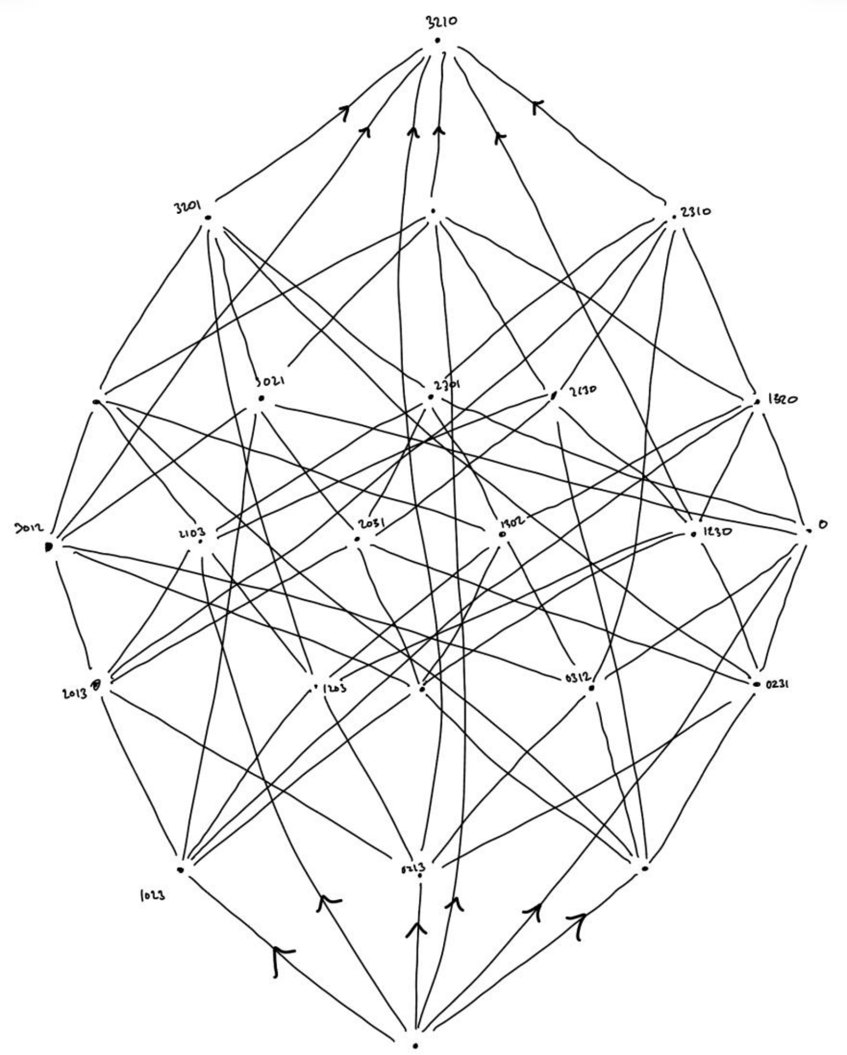


Image credit: Adam Wagner

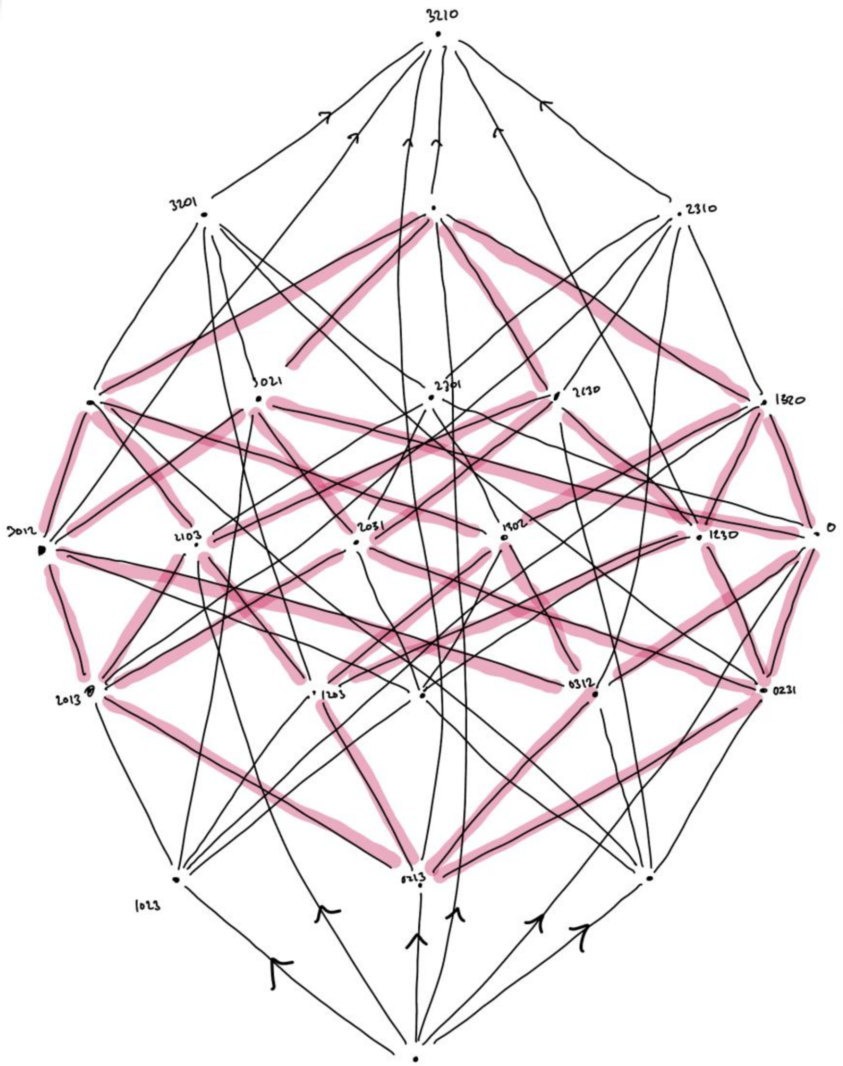
Hypercubes in Bruhat order

What is the largest hypercube inside Bruhat order in the symmetric group?



Hypercubes in Bruhat order

What is the largest hypercube inside Bruhat order in the symmetric group?



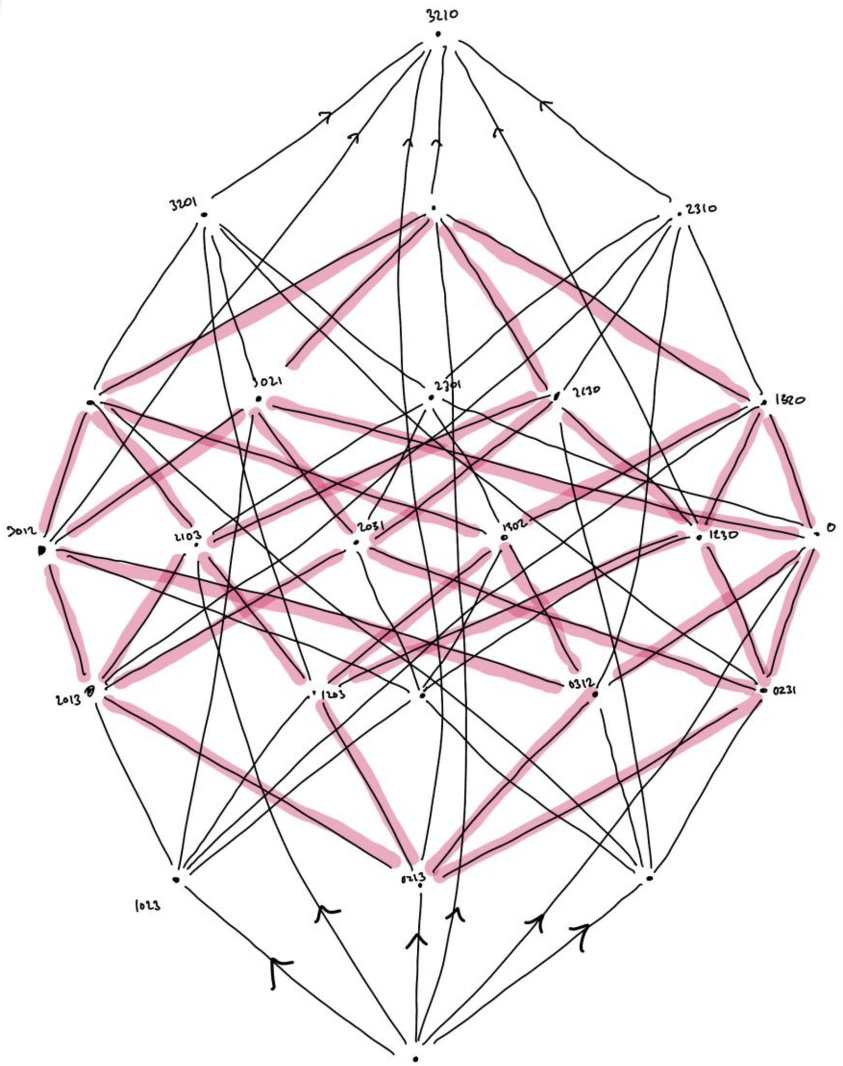
Hypercubes in Bruhat order

What is the largest hypercube inside Bruhat order in the symmetric group?

We have a big graph (with *n!* vertices). We are looking for a rare subgraph.*

We ask for python programs for the permutations at the bottom and top of the interval.

*) This is not the original problem (which involved the "*d* invariant") but ended up being equivalent to it.



Prompting AlphaEvolve

```
Act as an expert in computational algebra and combinatorial optimization. Your
task is to find pairs of permutations `x` and `y` in the symmetric group S_n
that have the largest possible "d-invariant", denoted d(x,y). This invariant is
a coefficient of a Kazhdan-Lusztig polynomial.

...


**Hint:** It is known that the maximum value of d(x,y) for large `n` is at
least `2n-5` (for odd `n`) and `2n-4` (for even `n`). Try to find pairs that
meet or exceed this bound!

...

You got this! I believe in you!!!
```

Over night AlphaEvolve found something very interesting...

```python
# --- Fractal/Recursive Constructions ---
# This "crazy" idea builds permutations recursively. We start with {1..n}
# and recursively partition the set, building up permutations of the subsets
# and combining them. This creates self-similar or "fractal" structures.

@lru_cache(maxsize=None)
def _build_recursive(numbers: Tuple[int, ...], split_type: str, combine_type: str):
    n_local = len(numbers)
    if n_local <= 1:
        return (numbers, numbers)

    if split_type == 'contiguous':
        m = n_local // 2
        part1 = numbers[:m]
        part2 = numbers[m:]
    else:  # 'interleaved'
        part1 = numbers[::2]
        part2 = numbers[1::2]

    # Recursive calls
    x1, y1 = _build_recursive(part1, split_type, combine_type)
    x2, y2 = _build_recursive(part2, split_type, combine_type)

    # Combine results
    x = x1 + x2
    if combine_type == 'swap':
        y = y2 + y1
```

```python
    # Recursive calls
    x1, y1 = _build_recursive(part1, split_type, combine_
    x2, y2 = _build_recursive(part2, split_type, combine_

    # Combine results
    x = x1 + x2
    if combine_type == 'swap':
        y = y2 + y1
    elif combine_type == 'swap_reverse':
        y = tuple(reversed(y2)) + tuple(reversed(y1))
    elif combine_type == 'alternating_merge': # New combi
        # Helper for alternating merge, specific to this
        def _merge_alternating_local(p1: Tuple[int, ...],
            res = []
            i, j = 0, 0
            while i < len(p1) or j < len(p2):
                if i < len(p1):
                    res.append(p1[i])
                    i += 1
                if j < len(p2):
                    res.append(p2[j])
                    j += 1
            return tuple(res)
```

Decompiling into a theorem

Jordan Ellenberg noticed that if $n = 2^m$ the subgraph has a simple description.

JE Jordan Ellenberg <ellenber@math.wisc.edu>   Wednesday, 15 October 2025 at 2:16 pm

To: Geordie Williamson; José Simental Rodríguez; Cc: David Plaza; nlibedinsky@gmail.com

Hi all,

When n is 2^m, the permutations generated by this code have a very nice description:

Let the set to be permuted be the set 0,...2^{m-1}, which is identified with length-m bitstrings by binary expansion.

x = reverse the bitstrings
y = reverse the bitstrings and apply NOT (i.e. x -> 1-x) to each bit

Note that both of these are involutions, and their product is evidently "NOT each bit" which is the longest permutation w_0.

Jordan

Theorem: (Ellenberg, Libedinsky, Plaza, Simental Rodriguez, Wagner, W.)
Dyadically well-distributed permutations provide a hypercube of size $\Omega(n \log n)$.

This experience was unusual

Typically, AlphaEvolve does a reasonable job at providing fixed *n* solutions.

This is the only case I know of that provided a general *n* solution.

How well does this work?

MATHEMATICAL EXPLORATION AND DISCOVERY AT SCALE

BOGDAN GEORGIEV, JAVIER GÓMEZ-SERRANO, TERENCE TAO, AND ADAM ZSOLT WAGNER

ABSTRACT. `AlphaEvolve` [223] is a generic evolutionary coding agent that combines the generative capabilities of LLMs with automated evaluation in an iterative evolutionary framework that proposes, tests, and refines algorithmic solutions to challenging scientific and practical problems. In this paper we showcase `AlphaEvolve` as a tool for autonomously discovering novel mathematical constructions and advancing our understanding of long-standing open problems.

To demonstrate its breadth, we considered a list of 67 problems spanning mathematical analysis, combinatorics, geometry, and number theory. The system rediscovered the best known solutions in most of the cases and discovered improved solutions in several. In some instances, `AlphaEvolve` is also able to *generalize* results for a finite number of input values into a formula valid for all input values. Furthermore, we are able to combine this methodology with `Deep Think` [148] and `AlphaProof` [147] in a broader framework where the additional proof-assistants and reasoning systems provide automated proof generation and further mathematical insights.

These results demonstrate that large language model-guided evolutionary search can autonomously discover mathematical constructions that complement human intuition, at times matching or even improving the best known results, highlighting the potential for significant new ways of interaction between mathematicians and AI systems. We present `AlphaEvolve` as a powerful new tool for mathematical discovery, capable of exploring vast search spaces to solve complex optimization problems at scale, often with significantly reduced requirements on preparation and computation time.

Arxiv:2511.02864v1, Nov 2025

From Terry Tao's blog

Watch this space!

NEW NIKODYM SET CONSTRUCTIONS OVER FINITE FIELDS

TERENCE TAO

2025

SUM-DIFFERENCE EXPONENTS FOR BOUNDEDLY MANY SLOPES, AND RATIONAL COMPLEXITY

TERENCE TAO

2025

BRUHAT INTERVALS THAT ARE LARGE HYPERCUBES

JORDAN ELLENBERG, NICOLAS LIBEDINSKY, DAVID PLAZA, JOSÉ SIMENTAL, AND GEORDIE WILLIAMSON

Tomorrow!

Broader context of AI for math

Immediate challenges

- We don’t yet have a culture around acknowledging AI use, versioning, certificates, benchmarks, documenting resource usage, ...
  See: https://ai-math.zulipchat.com/

- AI for math requires a specific skillset (computational math, software engineering and GPU basics, databases, ...).

- New NSF institute *ICARM* at CMU. Part of their mission is to fill this gap:

Institute for Computer-Aided Reasoning in Mathematics

Broader context

- Industry AI labs (and government) are not primarily motivated by intellectual curiosity.

- Some members of the community are concerned about the negative effects of industry involvement.
  See upcoming Leiden declaration:
  https://siliconreckoner.substack.com/p/tulips-and-turbulence

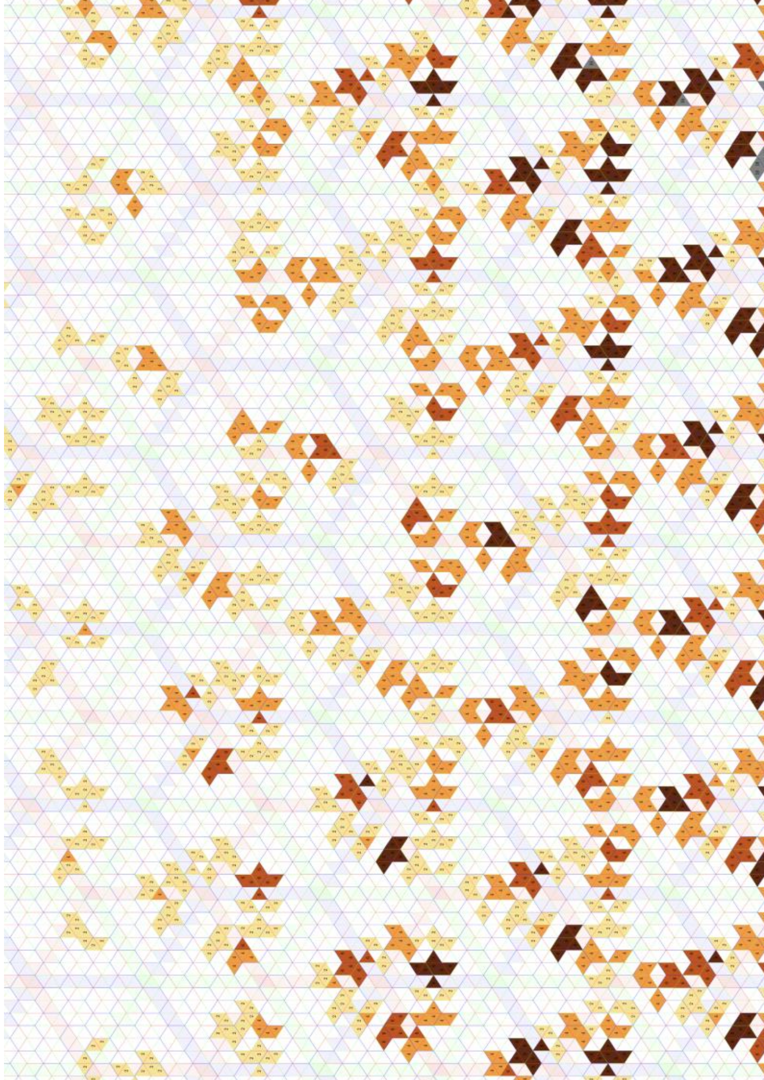
- The integrity of the mathematical community is one of our strongest assets. What we do and say matters.

Summary

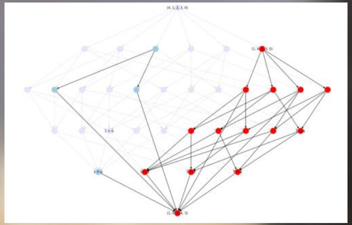
We should engage with the opportunities and challenges raised by AI.

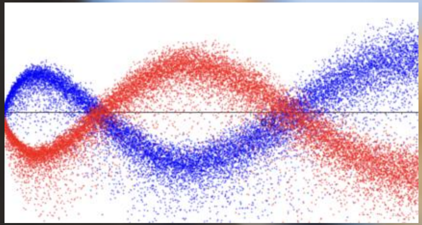
Three examples were presented, of AI assisted mathematical discovery.

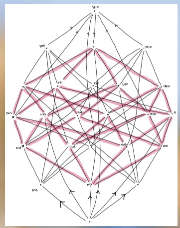
There are several other such examples, and no doubt the list will grow. AI for math is very much in its infancy.



THE UNIVERSITY OF SYDNEY | Mathematical Research Institute







Thank you