Reinforcement learning and pattern finding

Adam Zsolt Wagner

Rutgers Experimental Mathematics Seminar

March 28, 2024

Joint work with Jordan Ellenberg, Marijn Heule, Geordie Williamson

Reinforcement learning

Main idea: RL algorithms have managed to reach superhuman level play in Atari games, Go, Chess, starting from only the rules and learning everything else by themselves.

Can we teach neural networks to reach superhuman level play in the "game" of constructing graphs without 4-cycles, with as many edges as possible?

Can this same algorithm be used to try to learn to disprove any conjecture, by only inputting the statement and letting the algorithm figure out the rest?

Plan

Goal: find counterexamples to open conjectures via RL

- Try to avoid using human insights as much as possible
- Would like a general setup: use the same program for every problem, only change reward function
- Throw this setup at 100 open conjectures and hope for the best.

Example 1

Conjecture

For any graph G, we have $\lambda_1(G) + \mu(G) \geq \sqrt{n-1} + 1$.

Refuted in 2010, but smallest counterexample found has 600 vertices.

Game: offer edges one by one, agent can accept/reject each. A game lasts $\frac{n(n-1)}{2}$ turns.

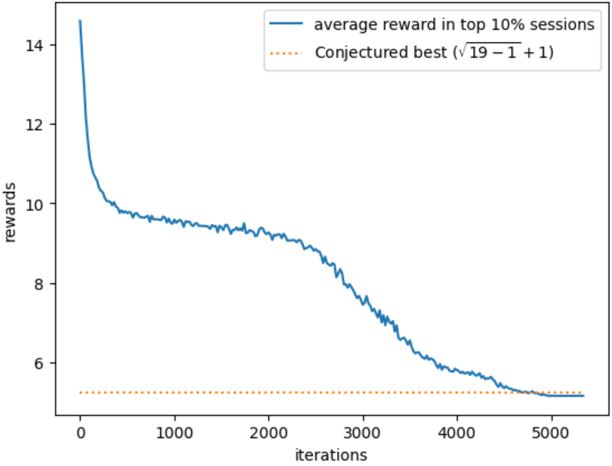
Reward: $\lambda_1 + \mu$ (minimize).

Run a reinforcement learning algorithm for $n = 19$:

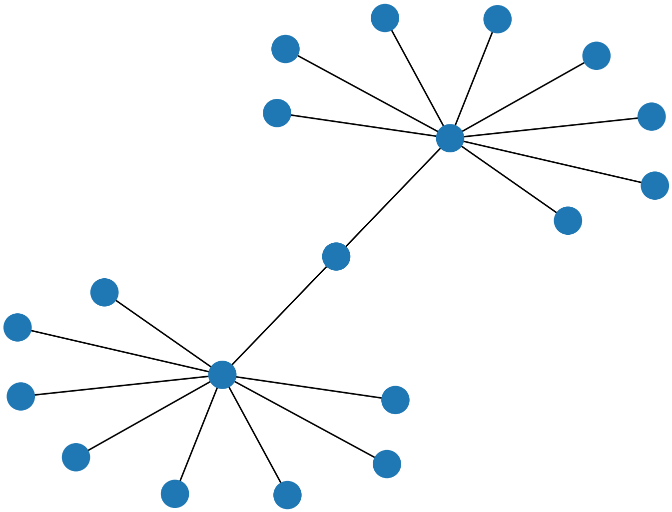
Example 1

Conjecture

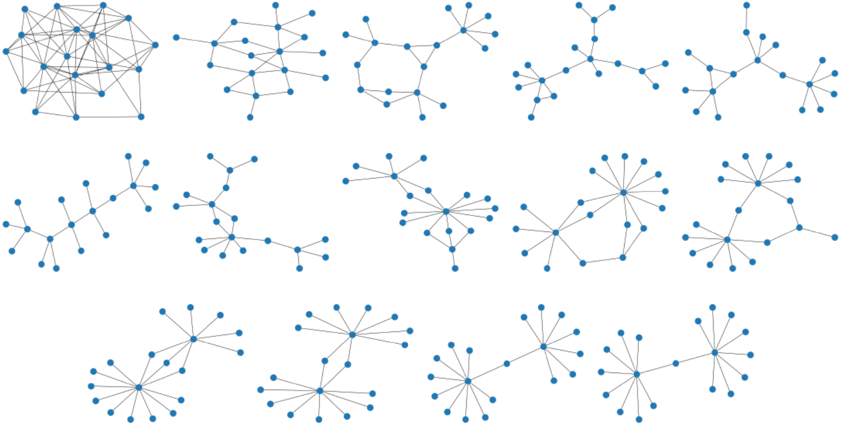
For any graph, $\lambda_1 + \mu \geq \sqrt{n-1} + 1$.



Example 1



Example 1



Example 2 – What if we don't succeed?

Conjecture (Auchiche–Hansen, 2016)

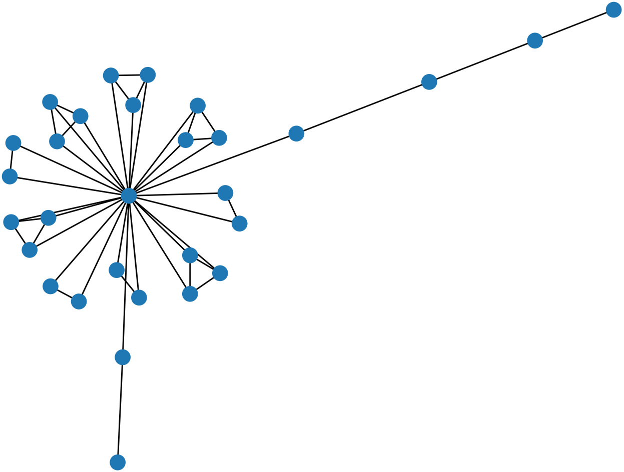
Let G be a connected graph with diameter D, proximity π and distance spectrum $\partial_1 \geq \ldots \geq \partial_n$. Then

$$\pi + \partial_{\lfloor \frac{2D}{3} \rfloor} > 0.$$

Reward: $\pi + \partial_{\lfloor \frac{2D}{3} \rfloor}$ (minimize).

Run it for $n = 30$:

Example 2



This is not quite a counterexample ($\pi + \partial_{\lfloor \frac{2D}{3} \rfloor} \approx 0.4$), but it tells us very clearly what counterexamples could look like.

Example 2

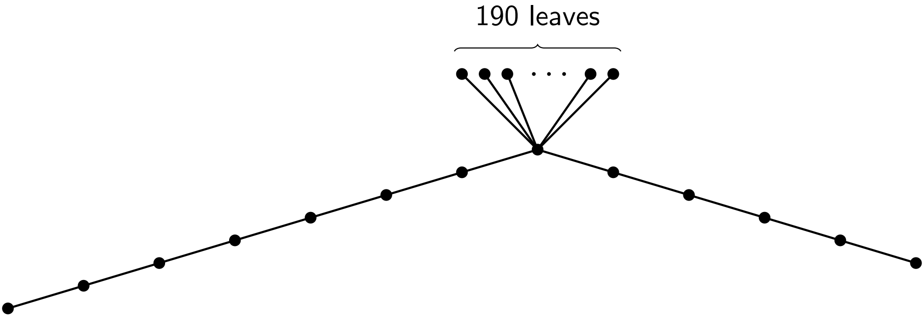


Figure: A counterexample to the conjecture

Example 3 - Not just graphs

Question (Brualdi–Cao)

How large can the permanent of a 312-pattern avoiding 0-1 matrix be?

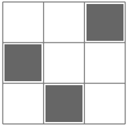


Figure: The pattern 312

$$\mathrm{per}(A) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} A_{i,\sigma(i)}$$

Example 3 - Not just graphs

Question (Brualdi–Cao)

How large can the permanent of a 312-pattern avoiding 0-1 matrix be?

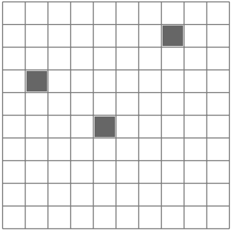


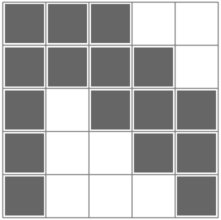
Figure: This is also not allowed

More precisely: we are not allowed to have three ones (dark squares) $(x_i, y_i) : i \in \{1, 2, 3\}$ such that $y_1 < y_2 < y_3$ and $x_2 < x_1 < x_3$.

Example 3

Conjecture (Brualdi–Cao, 2020)

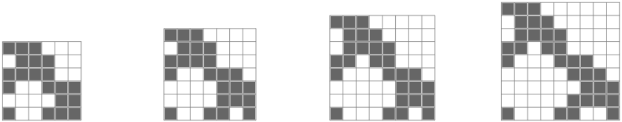
The best you can do is $Fib_{n+2} - 1$.



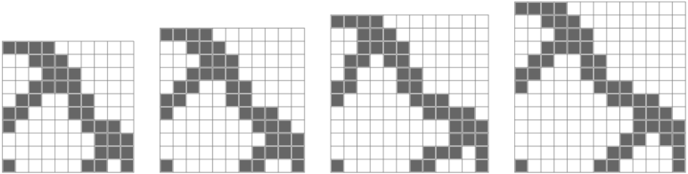
Reward: $\operatorname{per}(A) - \operatorname{penalty}(\#\text{ of 312-s})$

Example 3

$\mathrm{per}(A_1) = 1$ $\quad \mathrm{per}(A_2) = 2$ $\quad \mathrm{per}(A_3) = 4$ $\quad \mathrm{per}(A_4) = 8$ $\quad \mathrm{per}(A_5) = 16$



$\mathrm{per}(A_6) = 32$ $\quad \mathrm{per}(A_7) = 64$ $\quad \mathrm{per}(A_8) = 120$ $\quad \mathrm{per}(A_9) = 225$



$\mathrm{per}(A_{10}) = 424$ $\quad \mathrm{per}(A_{11}) = 795$ $\quad \mathrm{per}(A_{12}) = 1484$ $\quad \mathrm{per}(A_{13}) = 2809$

These are best possible for $n \leq 8$ (computer proof). So the sequence starts with $1, 2, 4, 8, 16, 32, 64, \mathbf{120}$.

Example 4 - non-obvious reward function

Question (Hogben–Reinhart)

Do there exist two graphs G and H such that they have the same $\mathcal{D}^L$-eigenvalues, but G is transmission regular and H is not?

RL has constructed two graphs. What should the reward function be?

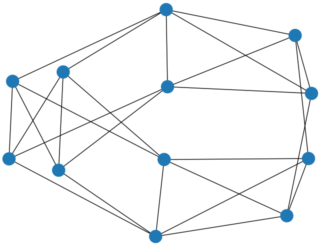
Idea:

$$\text{score}(G, H) = f_1(G, H) + f_2(G) + f_3(H),$$

where

- f_1 measures how close the $\mathcal{D}^L$-spectrum of G and H is,
- f_2 measures how close G is to being transmission regular, and
- f_3 gives a penalty if H is transmission regular.

Example 4



The graph on the left is transmission regular, whereas the graph on the right is not. The characteristic polynomials of their distance Laplacians are the same $\left(x^{12} - 216x^{11} + 21188x^{10} - 1245904x^{9} + 48797440x^{8} - 1336652544x^{7} + 26129121472x^{6} - 364516883456x^{5} + 3556516628224x^{4} - 23113129559040x^{3} + 90045806284800x^{2} - 159318669312000x\right)$, so they are $\mathcal{D}^{L}$-cospectral.

Example 5 - Infinite problems?

Many interesting problems can not have finite counterexamples.

Conjecture (Erdős, 1962)

The function

$$K_4(G) + K_4(\bar{G})$$

is asymptotically minimized by random graphs.

Thomason (1989): This is false!

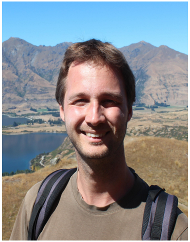


Figure: Gwenaël Joret

Example 5 - Infinite problems?

How can we refute such conjectures using RL?

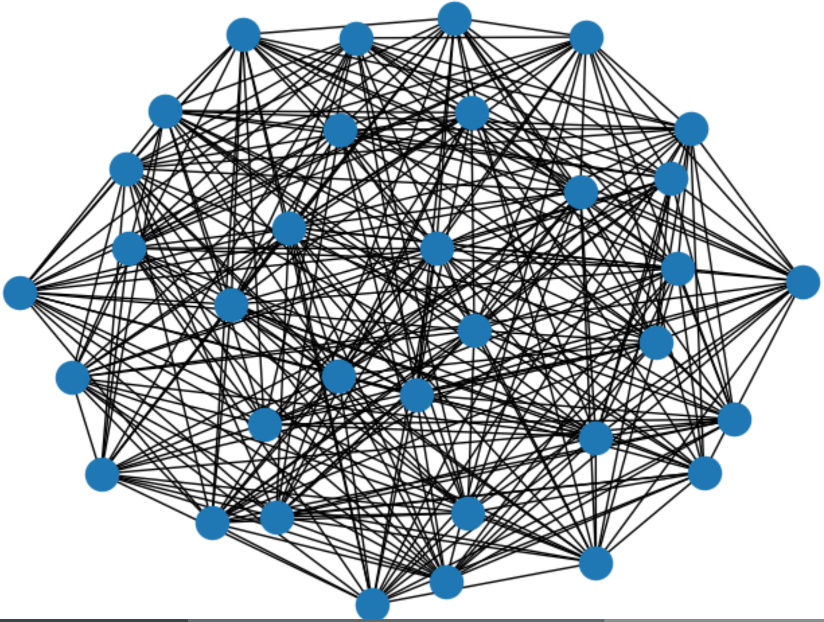
- Find the best construction for $n = 50$, then generalize "by hand".
- Somehow reduce to a finite conjecture.

Solution: "blowing up"! Construct a finite graph G, so that $G \times K_m$ is a counterexample as $m \to \infty$.

$\lim_{m\to\infty} \frac{K_4(G\times K_m)+K_4(\overline{G\times K_m})}{m^4}$ depends only on G, and there is an easy formula for it. This will be our reward function.

Run RL for $n = 34 \longrightarrow$ find a counterexample.

Example 5



Pros and cons

How useful is this simple RL setup in pure maths research?

Pros:

- Simple and fun baseline method that can be thrown at a large class of problems
- Occasionally it works...

Cons:

- ...but most of the times it doesn't.
- Very slow, doesn't scale well
- Often doesn't perform better than simple greedy searches
- It is **not** the case that we can throw AlphaZero at any conjecture, and expect it to work better than any other algorithm. (Mehrabian et al, 2023)

Generative methods

Joint work with Jordan Ellenberg, Marijn Heule, Geordie Williamson.

How can we use generative methods in maths research?

Let's pick a problem in maths where there is a mysterious pattern that we don't understand, and see if transformers can understand it better and help us generalize it.

Question

How many integers can we choose between 1 and N, without choosing 3 numbers that form an arithmetic progression?

Central problem in pure mathematics, current best bounds (Behrend 1946, Bloom–Sisask 2023):

$$e^{-c\sqrt{\log N}} \cdot N \leq r_3(n) \leq e^{-c \log^{1/9} N} \cdot N.$$

What about the 2D variant of this problem?

The isosceles triangle problem

A 3-AP in one dimension is three distinct numbers a, b, c with $d(a, b) = d(b, c)$.

Question (Ellenberg)

How many points can we choose in the $N \times N$ grid, without choosing three points that satisfy $d(a, b) = d(b, c)$, i.e. without creating any isosceles triangles?

Let this maximum be $f(N)$. What bounds can we prove on $f(N)$?

The isosceles triangle problem – bounds

Lower bounds:

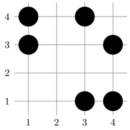
- $f(N) \geq \frac{N}{\sqrt{\log N}}$ – simple alteration argument
- $f(N) \geq cN$ – Random greedy independent set process

Upper bounds:

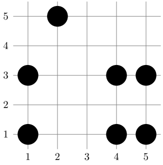
- $f(N) \leq e^{-c \log^{1/9} N} \cdot N^2$ – trivial bound
- $f(N) \leq N^{1.99}$ – open

It would be nice to have a guess about the asymptotics of $f(N)$.

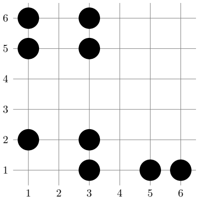
For small grids, we can do it by hand:



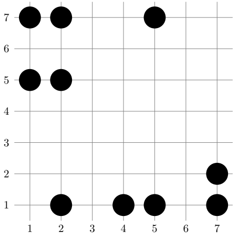
(a) $f(4) = 6$



(b) $f(5) = 7$

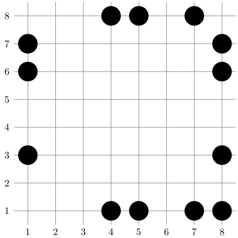


(c) $f(6) = 9$

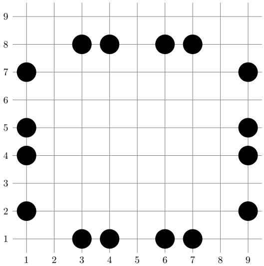


(d) $f(7) = 10$

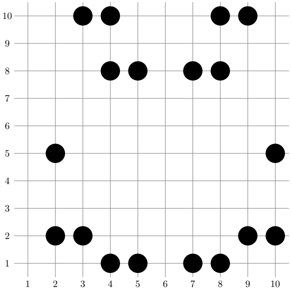
For slightly larger grids, we can use LP solvers:



(a) $f(8) = 13$



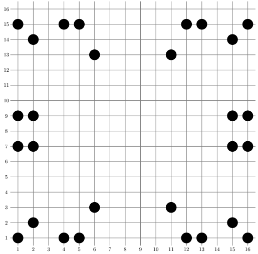
(b) $f(9) = 16$



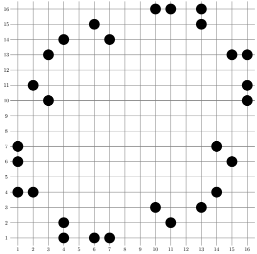
(c) $f(10) = 18$

The isosceles triangle problem

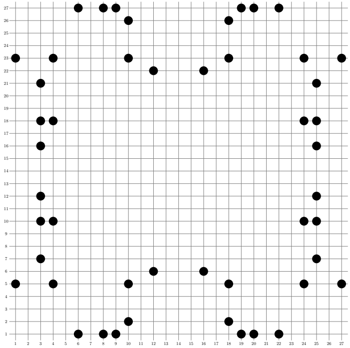
SAT solvers work up to $n \approx 30$.



(a) $f(16) = 28$

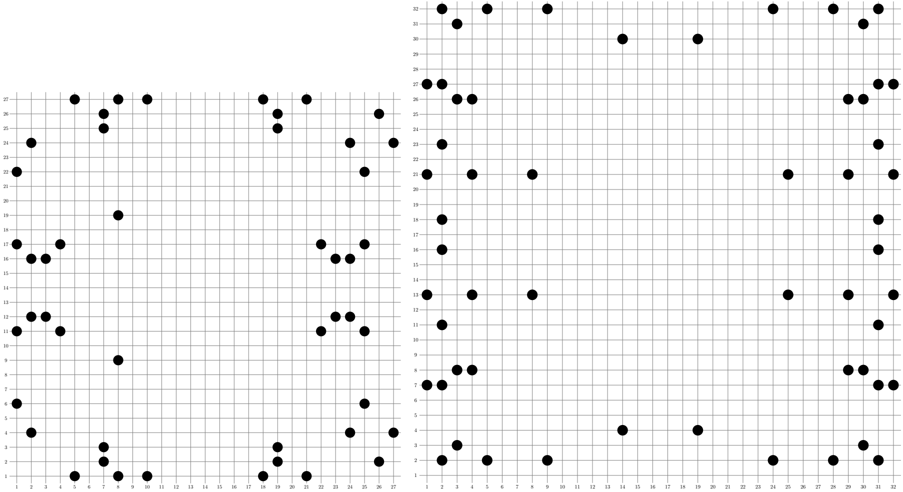


(b) $f(16) = 28$



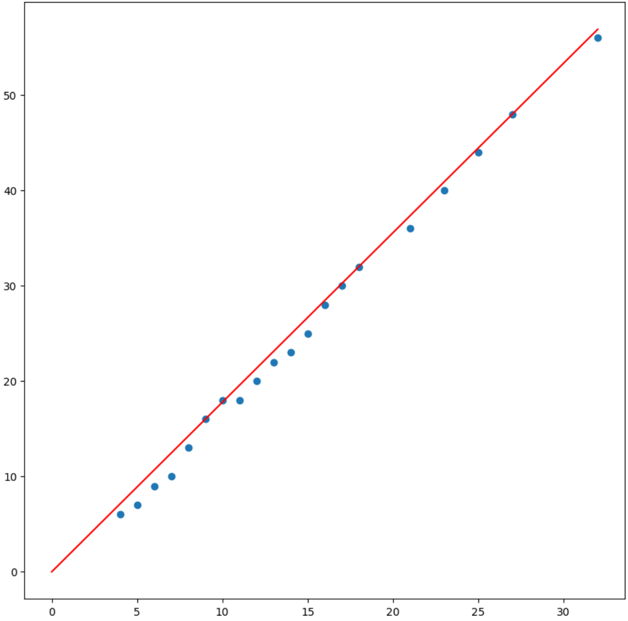
(c) $f(27) = 48$

The isosceles triangle problem



(a) $f(27) = 48$ (b) $f(32) \geq 56$

The isosceles triangle problem



The isosceles triangle problem

As we suspected, $f(n)$ seems to be a linear function, with $f(n) \approx \frac{16}{9}n$.

As $f(16) = 28$ and $f(32) \approx 56$, we expect $f(64) \approx 112$. Can we find it?

With LP solvers, SAT solvers, local search methods, after a few months we managed to find a 108.

The isosceles triangle problem

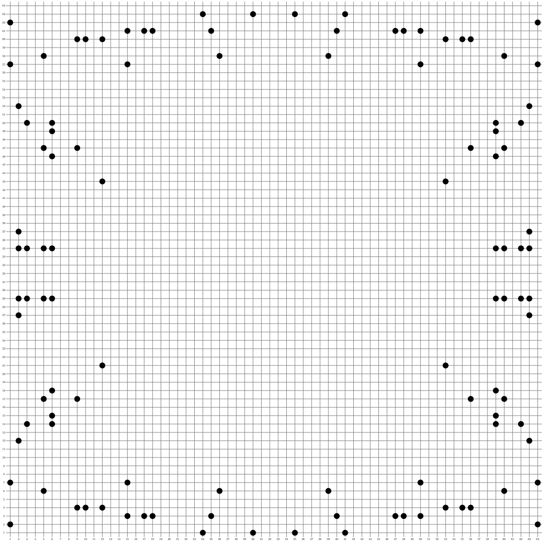


Figure: $f(64) \geq 108$

The isosceles triangle problem

We create a database of many thousands of good 64×64 constructions.

We train a transformer on these, and then generate more constructions like those in the dataset. We used a simple transformer implementation by Andrej Karpathy, called Makemore.

How nice would it be if the transformer generated some 109s and 110s as well, if we let it generate new constructions for a long time?

Transformer generated a bunch of 104-108s, but nothing better.

But local search is really struggling at such high scores: it takes days to find a single new 108. The transformer finds new good constructions much more frequently!

Idea: run a quick local search from all new constructions given by the transformer. By pure numbers, maybe 1% of them will not be a local maximum.

The isosceles triangle problem

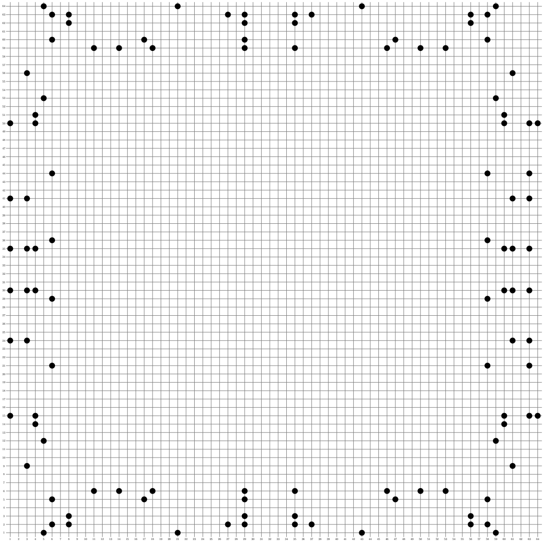


Figure: $f(64) \geq 110$

The isosceles triangle problem

This worked, but maybe it was pure luck. Let's repeat this experiment on 100×100 grids.

We ran several different local search methods for 3 weeks. The best they found was 154 (the optimum should likely be around 176-ish).

We trained a transformer on the best 1 million constructions, then generated new grids with it, launching local searches from all new grids.

This led to a grid with score 160!

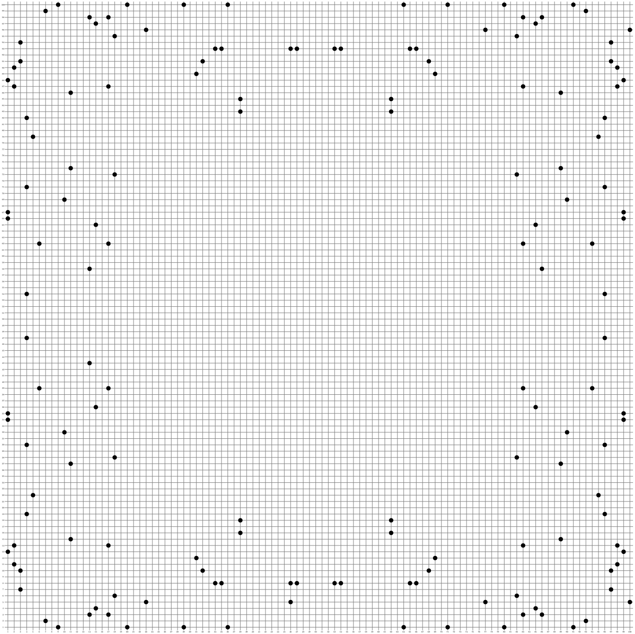


Figure: $f(100) \geq 154$

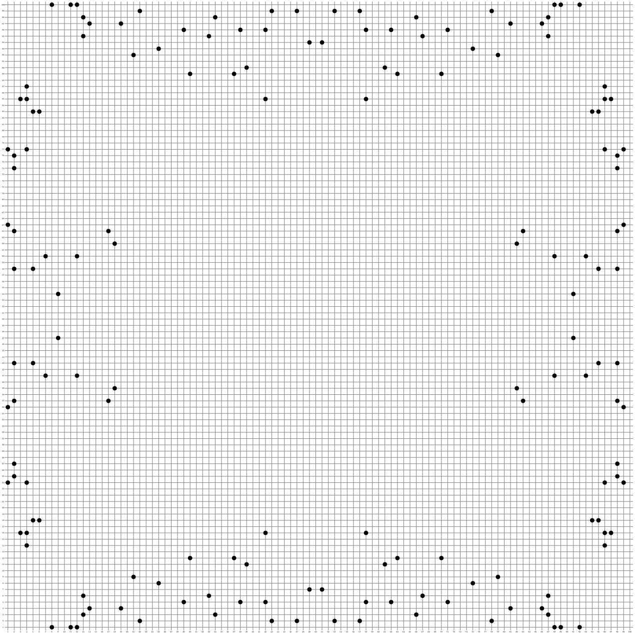


Figure: $f(100) \geq 160$

Conclusion

This method repeatedly worked over multiple experiments. The method seems to be:

1. Run local search until it reaches its limit, where you reach a high enough score so that it takes days to find new constructions with this score.
2. At this point, train a transformer on all (1M+) best constructions
3. Ask the transformer to generate new constructions, and launch local searches from them
4. The scores of the local searches from these new seeds will be better than the scores of the original local searches, and it leads to new, higher scores.

Challenge: what would the optimal construction for $f(10,000)$ look like?

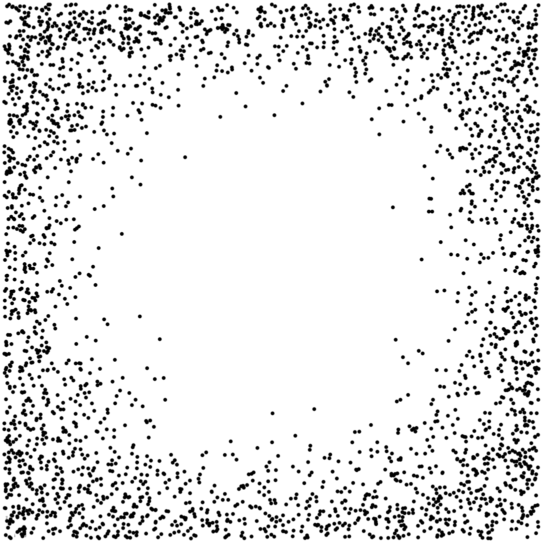


Figure: $f(10,000) \geq 3000$

Conclusion

Even though most mathematicians don't use machine learning in their research yet, there exist already some interesting applications.

There are still many low hanging fruits and things are happening very quickly.

I can't wait to hear all the new ideas you will come up with in the next years.