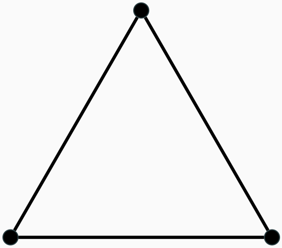
A Strategy for Lowering the Upper Bound of R(5,5)[1]

Thibault Gauthier

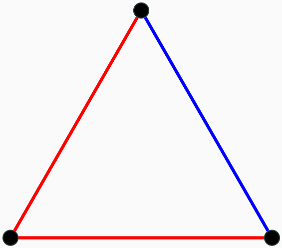
Czech Technical University in Prague

[1] This work was supported by the Czech Science Foundation grant no. 25-17929X, Czech Ministry of Education, Youth and Sports within the ERC CZ program under the project POSTMAN no. LL1902, and by Amazon Research Awards.

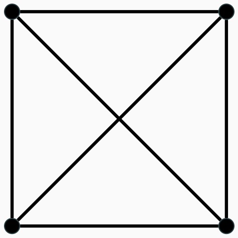
A complete graph of size 3



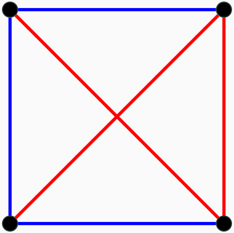
A blue-red coloring avoiding 3-cliques



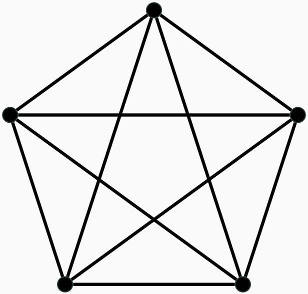
A complete graph of size 4



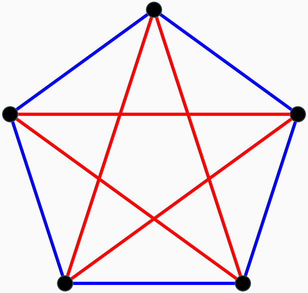
A blue-red coloring avoiding 3-cliques



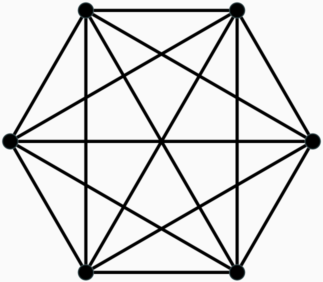
A complete graph of size 5



A blue-red coloring avoiding 3-cliques



A complete graph of size 6



Definition of the Ramsey Number

The Ramsey number $R(n, m)$ is the smallest k such that:

- it is not possible to find a coloring of the complete graph of size k which avoids blue n-cliques and red m-cliques.

Example: $R(3, 3) = 6$

Ramsey Theorem: $R(n, m)$ exists for every $n, m \in \mathbb{N}$.

The set of graphs (modulo isomorphism) of size k which avoid blue n-cliques and red m-cliques is noted $\mathcal{R}(n, m, k)$.
A graph in $\mathcal{R}(n, m, k)$ will be called a $\mathcal{R}(n, m, k)$-graph.

Example: $\mathcal{R}(3, 3, 5) \neq \emptyset$ and $\mathcal{R}(3, 3, 6) = \emptyset$

We rely on the nauty algorithm to normalize graphs.

Why prove that R(5,5) = 25?

"Suppose aliens invade the earth and threaten to obliterate it in a year's time unless human beings can find R(5,5). We could marshal the world's best minds and fastest computers, and within a year we could probably calculate the value. If the aliens demanded R(6,6), however, we would have no choice but to launch a preemptive attack."

– Paul Erdős

How to find the value of $R(5,5)$?

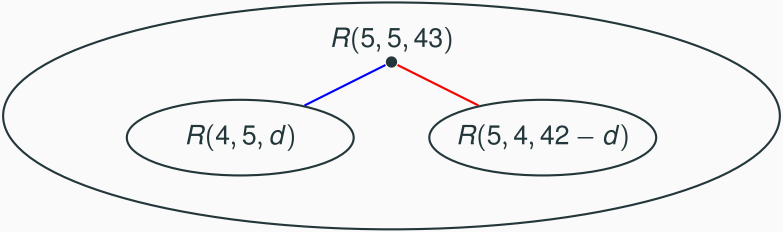
Improve the lower bound and the upper bound.

- 1989: $R(5,5) \geq 43$
- 1995: $R(5,5) \leq 50$
- 2017: $R(5,5) \leq 48$
- 2024: $R(5,5) \leq 46$

Some experts in the field have conjectured that $R(5,5) = 43$.

A standard strategy applied to $R(5,5) \leq 43$

Proof by contradiction: suppose there exists a $R(5,5,43)$-graph.



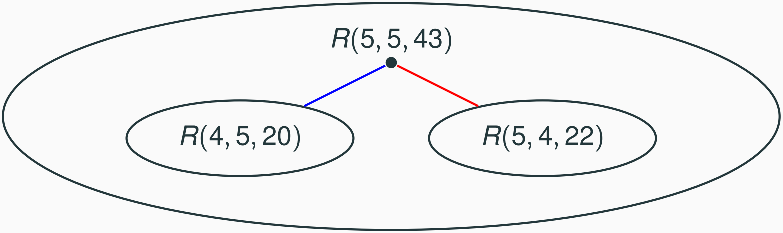
How many gluing problems are they?

The worst case by far is $d = 20$ (we can ignore all other cases).

$$|R(4,5,20)| \times |R(4,5,22)| = \text{number of problems}$$
$$(8.5 \times 10^{18}) \times (1.9 \times 10^{15}) = 1.6 \times 10^{34}$$

Without isomorphism checking: $20! \times 22!$ times more problems.

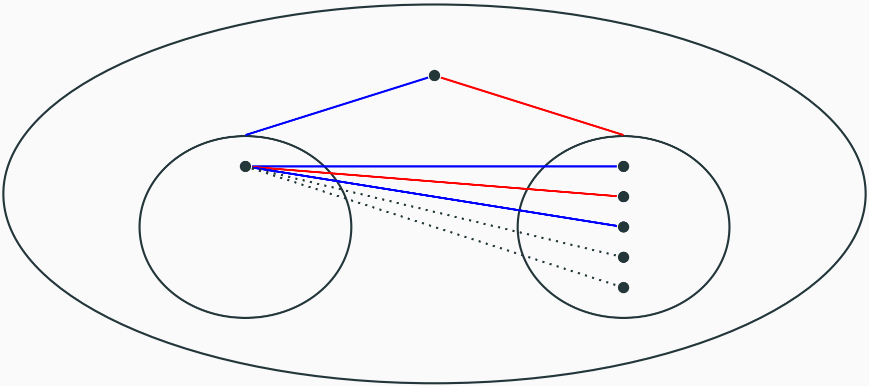
Solving one gluing problem



Solving one gluing problem P_0

```
# o o o o o o o o o o o o o o o o o o o o - - - - - - - - - - - - - - - - - - - - - -
o # o o o o o o o o o - - - - - - - - - -
o o # - - o - - - - - o - o o o o - - - o
o o - # - - - - o - o - o o o o - - o - -
o o - - # - - o - o - - o - - - o o o - -
o o o - - # o - - - - o o o o - - - - o -
o o - - - o # - - o o o - o - - o o - o -
o o - - o - - # o - o - o - - - o - - o o
o o - o - - - o # o - - - o o - - - - o o
o o - - o - o - o # - o o - - o - o - o o
o o - o - - o o - - # o - - - o - o o o o
o - o - - o o - - o o # o - - - - - - - o
o - - o o o - o - o - o # - - - - - o o -
o - o o - o o - o - - - - # - o - o o - o
o - o o - o - - o - - - - - # o o o - o -
o - o o - - - - - o o - - o o # - o - o -
o - o - o - o o - - - - - - o - # o o o o
o - - - o - o - - o o - - o o o o # - - o
o - - o o - - - - - o - o o - - o - # o o
o - - - - o o o o o o - o - o o o - o # -
o - o - - - - o o o o o - o - - o o o - #
-                                         # o o o o o o o o o o - - - - - - - - - - -
-                                         o # o o o - o o - - o - o - - o o - - o - o
-                                         o o # - o o o o o - - - o o o - o - o - - -
-                                         o o - # o o o - o o - - - o - - - o - - o o
-                                         o o o o # o - - o - - - o o - o - - - - o o
-                                         o - o o o # - o - o - - o - o o - o - o o -
-                                         o o o o - - # - - o - o - o o - o - - o - o
-                                         o o o - - o - # - - o o o - - - - o o o - -
-                                         o - o o o - - - # - o o - o - o o - o - o -
-                                         o - - o - o o - - # - - - - o - o o - o o o
-                                         o o - - - - - o o - # o - - - o - o o o - o
-                                         - - - - - - o o o - o # o o o o - - o o o o
-                                         - o o - o o - o - - - o # o o - o o o o o -
-                                         - - o o o - o - o - - o o # - o o o o o - o
-                                         - - o - - o o - - o - o o - # o o o o - o o
-                                         - o - - o o - - o - o o - o o # o o - o o o
-                                         - o o - - - o - o o - - o o o o # o - o o -
-                                         - - - o - o - o - o o - o o o o o # o - - o
-                                         - - o - - - - o o - o o o o o - - o # - - o
-                                         - o - - - o o o - o o o o o - o o - - # o -
-                                         - - - o o o - - o o - o o - o o o - - o # -
-                                         - o - o o - o - - o o o - o o o - o o - - #
```

Solving one gluing sub-problem



Solving one gluing sub-problem Q_0

```
# ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ − − − − − − − − − − − − − − − − − − − − − −
○ # ○ ○ ○ ○ ○ ○ ○ ○ ○ − − − − − − − − − − − − − − − − − − − − − ○ ○ − − − ○ − − ○ − −
○ ○ # − − ○ − − − − − ○ − ○ ○ ○ ○ − − − ○
○ ○ − # − − − − ○ − ○ − ○ ○ ○ ○ − − ○ − −
○ ○ − − # − − ○ − ○ − − ○ − − − ○ ○ ○ − −
○ ○ ○ − − # ○ − − − − ○ ○ ○ ○ − − − − ○ −
○ ○ − − − ○ # − − ○ ○ ○ − ○ − − ○ ○ − ○ −
○ ○ − − ○ − − # ○ − ○ − ○ − − − ○ − − ○ ○
○ ○ − ○ − − − ○ # ○ − − − ○ ○ − − − − ○ ○
○ ○ − − ○ − ○ − ○ # − ○ ○ − − ○ − ○ − ○ ○
○ ○ − ○ − − ○ ○ − − # ○ − − − ○ − ○ ○ ○ ○
○ − ○ − − ○ ○ − − ○ ○ # ○ − − − − − − − ○
○ − − ○ ○ ○ − ○ − ○ − ○ # − − − − − ○ ○ −
○ − ○ ○ − ○ ○ − ○ − − − − # − ○ − ○ ○ − ○
○ − ○ ○ − ○ − − ○ − − − − − # ○ ○ ○ − ○ −
○ − ○ ○ − − − − − ○ ○ − − ○ ○ # − ○ − ○ −
○ − ○ − ○ − ○ ○ − − − − − − ○ − # ○ ○ ○ ○
○ − − − ○ − ○ − − ○ ○ − − ○ ○ ○ ○ # − − ○
○ − − ○ ○ − − − − − ○ − ○ ○ − − ○ − # ○ ○
○ − − − − ○ ○ ○ ○ ○ ○ − ○ − ○ ○ ○ − ○ # −
○ − ○ − − − − ○ ○ ○ ○ ○ − ○ − − ○ ○ ○ − #
− −                                       # ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ − − − − − − − − − − −
− −                                       ○ # ○ ○ ○ − ○ ○ − − ○ − ○ − − ○ ○ − − ○ − ○
− −                                       ○ ○ # − ○ ○ ○ ○ ○ − − − ○ ○ ○ − ○ − ○ − − −
− −                                       ○ ○ − # ○ ○ ○ − ○ ○ − − − ○ − − − ○ − − ○ ○
− −                                       ○ ○ ○ ○ # ○ − − ○ − − − ○ ○ − ○ − − − − ○ ○
− −                                       ○ − ○ ○ ○ # − ○ − ○ − − ○ − ○ ○ − ○ − ○ ○ −
− −                                       ○ ○ ○ ○ − − # − − ○ − ○ − ○ ○ − ○ − − ○ − ○
− −                                       ○ ○ ○ − − ○ − # − − ○ ○ ○ − − − − ○ ○ ○ − −
− −                                       ○ − ○ ○ ○ − − − # − ○ ○ − ○ − ○ ○ − ○ − ○ −
− −                                       ○ − − ○ − ○ ○ − − # − − − − ○ − ○ ○ − ○ ○ ○
− −                                       ○ ○ − − − − − ○ ○ − # ○ − − − ○ − ○ ○ ○ − ○
− ○                                       − − − − − − ○ ○ ○ − ○ # ○ ○ ○ ○ − − ○ ○ ○ ○
− ○                                       − ○ ○ − ○ ○ − ○ − − − ○ # ○ ○ − ○ ○ ○ ○ ○ −
− −                                       − − ○ ○ ○ − ○ − ○ − − ○ ○ # − ○ ○ ○ ○ ○ − ○
− −                                       − − ○ − − ○ ○ − − ○ − ○ ○ − # ○ ○ ○ ○ − ○ ○
− −                                       − ○ − − ○ ○ − − ○ − ○ ○ − ○ ○ # ○ ○ − ○ ○ ○
− ○                                       − ○ ○ − − − ○ − ○ ○ − − ○ ○ ○ ○ # ○ − ○ ○ −
− −                                       − − − ○ − ○ − ○ − ○ ○ − ○ ○ ○ ○ ○ # ○ − − ○
− −                                       − − ○ − − − − ○ ○ − ○ ○ ○ ○ ○ − − ○ # − − ○
− ○                                       − ○ − − − ○ ○ ○ − ○ ○ ○ ○ ○ − ○ ○ − − # ○ −
− −                                       − − − ○ ○ ○ − − ○ ○ − ○ ○ − ○ ○ ○ − − ○ # −
− −                                       − ○ − ○ ○ − ○ − − ○ ○ ○ − ○ ○ ○ − ○ ○ − − #
```

Solving one gluing problem

Result for P_0:

- 2^{22} subproblems $\rightarrow$ 622,746 non-trivial
- Most solved by `CaDiCaL` in < 10s
- Hardest case: 8 hours
- Total: 200 CPU-days

Estimate for all gluing problems:
$(1.62 \times 10^{34}) \times 200$ CPU-days $= 3.2 \times 10^{36}$ CPU-days

Generalization strategy

Construct $Q_0, Q_1, Q_2, \ldots, Q_n$ by forgetting the color of edges (one at a time).

Advantage:

- Solving Q_i solves many subproblems simultaneously.
- Covered subproblems may be from different gluing problems.

Disadvantage:

- Generalized problems are harder.
- Requires to check if a subproblem has already been covered.

Edge selection

Run `CaDiCaL` on all potential generalizations from Q_i.
(one for each colored edges in Q_i)

Select edge with the lowest amortized solving time:

$$\frac{\text{solving time}}{\text{number of subproblems covered}}$$

The number of subproblems covered is computed by a model counter.

Generalization algorithm

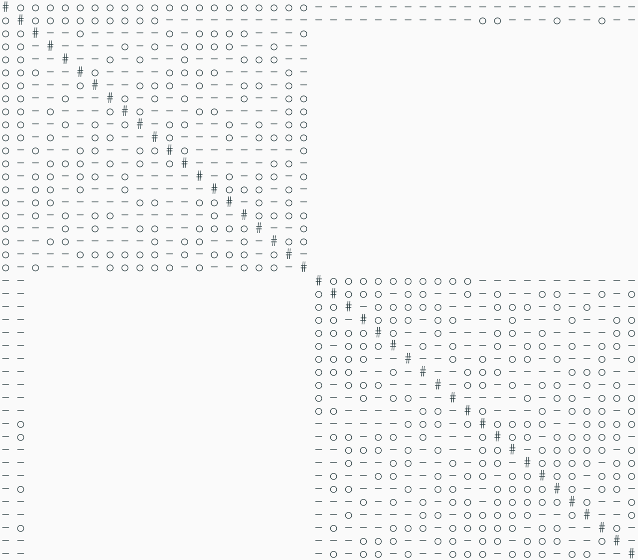
Result for Q_0:

- Generalization sequence: $Q_0, Q_1, Q_2, \ldots, Q_{298}$.
- Stopped when CaDiCaL solving time exceeded 20 seconds.
- Took 10 hours to compute (rounded up to 1 day in our estimate).
- Solves an estimated 2.4×10^{27} non-isomorphic subproblems

Estimate for all gluing subproblems (thus all gluing problems):

$$\frac{(1.62 \times 10^{34}) \times 622,746 \times 1 \text{ CPU-day}}{2.4 \times 10^{27}} = 4.2 \times 10^{12} \text{ CPU-days}$$

The subproblem Q_0



The generalized subproblem Q_{298}

Conclusion

Summary:

- Solved one of the gluing problem P_0 in 200 CPU-days.
  Estimate for all gluing problems: 3.2×10^{36} CPU-days
- Generalization of 298 vertices in a subproblem Q_0.
  Estimate for all gluing problems: 4.2×10^{12} CPU-days

Key ideas:

- **symmetry-breaking** (splitting vertex, isomorphism checking)
- **generalization** (don't-care edges)

Future ideas:

- deeper splitting
- simultaneous edge generalization
- faster edge selection: heuristics, graph neural networks.