# Quantum speedups for treewidth

Vladislavs Kļevickis, Krišjānis Prūsis, Jevgēnijs Vihrovs

Joint Estonian-Latvian Theory Days 06.05.2022.

# Exact algorithms for NP-complete problems

- Travelling salesman problem:
  - Check all permutations: $O^*(n!)$ time ☹ slow
- Exact exponential algorithms:
  - Divide & conquer: $O^*(4^n)$ time / $O^*(1)$ space ☺
    [Gurevich & Shelah 87, others]
  - Dynamic programming: $O^*(2^n)$ time / $O^*(2^n)$ space much better!
    [Bellman 62, Held & Karp 62]

# Quantum algorithms

- Grover's algorithm:
  - Finds $\min\{a_1, a_2, \dots, a_N\}$ in time $O(\sqrt{N} \log N)$.

    [Grover 96]
- Quantum speedups:
  - Divide & conquer: $O^*(2^n)$ time / $O^*(1)$ space
  - Dynamic programming: $O^*(1.81^n)$ time / $O^*(1.81^n)$ (QRAM) space

    [Ambainis, Balodis, Iraids, Kokainis, Prūsis, Vihrovs 19]

# Quantum algorithms

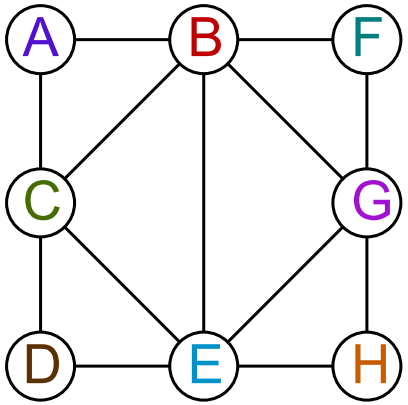
- Quantum speedups!
  - Travelling salesman problem, Minimum set cover, Graph ordering problems, …
  - Classically: $O^*(2^n)$ time & space with known algorithms
  - Quantumly: $O^*(1.81^n)$ time & space
- However, for Treewidth…
  - $O^*(1.75^n)$ time & space classical algorithm

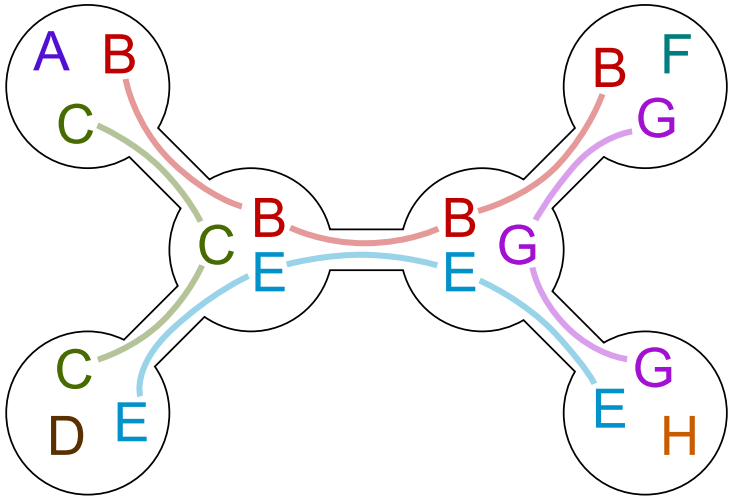
[Fomin & Villanger 12]

For these we can do even better, $O^*(1.72^n)$ quantumly

# Treewidth

- Given a graph $G = (V, E)$, a *tree decomposition* is a tree such that:
  - Vertices (*bags*) are subsets of $V$
  - All vertices $V$ appear in the tree
  - For any vertex $v \in V$, the bags containing $v$ form a connected subtree
  - For any edge $\{u, v\} \in E$, there is a bag with both $u$ and $v$.
- Width = the size of the largest bag – 1
- Treewidth of $G$ = the smallest possible width of a tree decomposition



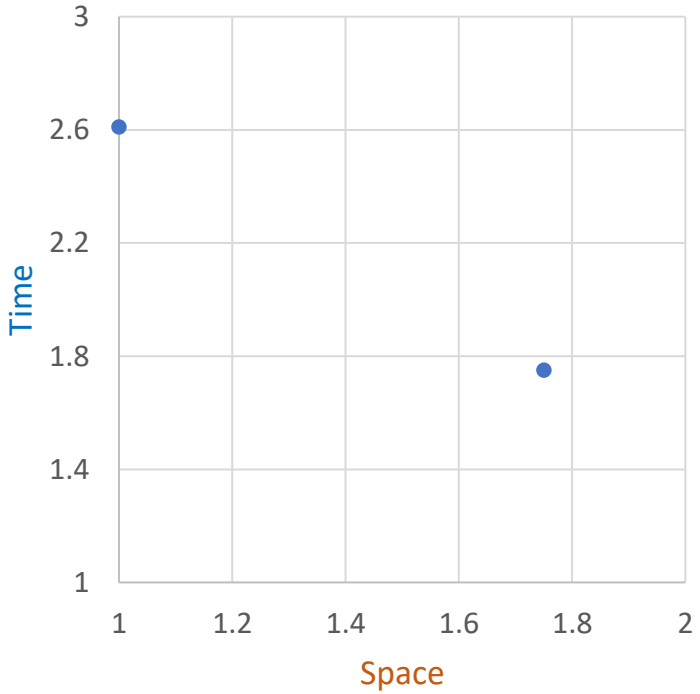



# Treewidth

- Prominent application in parametrized algorithms: many computational problems become tractable if the treewidth is small
- Classical algorithms:
  - $O^*(1.75^n)$ time / $O^*(1.75^n)$ space
  - $O^*(2.61^n)$ time / $O^*(1)$ space

[Fomin & Villanger 12]

- What can we do quantumly?...



# Fomin's and Villanger's algorithm

1. Examines candidates $\chi \subseteq V$
2. Finds the best tree decomposition with $\chi$ as a fixed bag

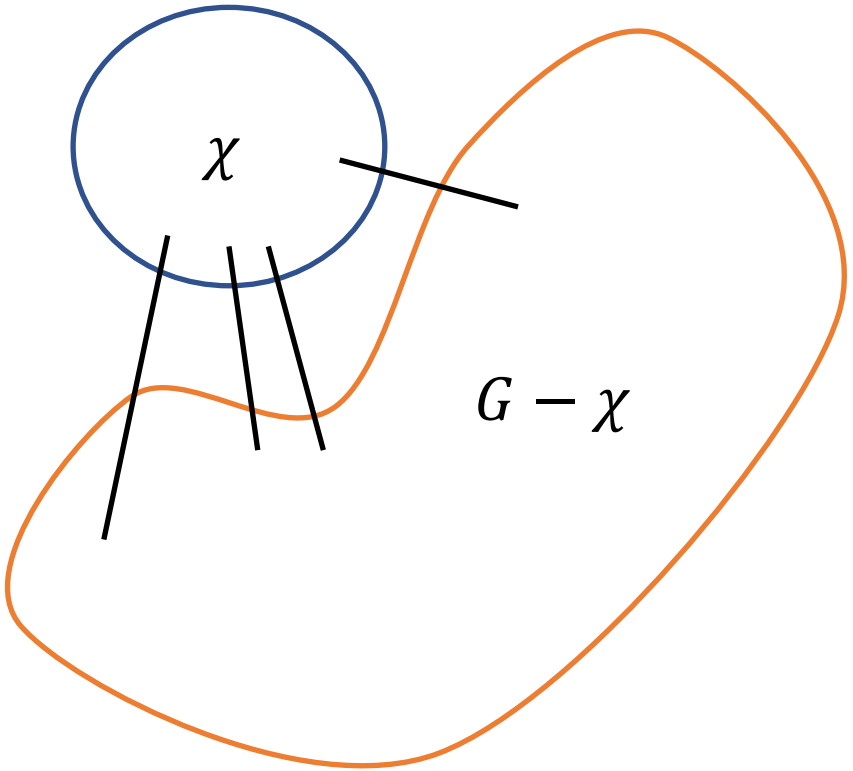
Time complexity:

- Part 1 – $O^*(\#\chi)$ (combinatorial lemma)
- Part 2 – $O^*(4^{n-|\chi|})$ (divide & conquer)

By optimizing parameters, time is

$$O^*\left(\sum_\chi 4^{n-|\chi|}\right) = O^*(2.61^n)$$

Space is polynomial, $O^*(1)$.



# Improvements

Classically:

- Part 1 – $O^*(\#\chi)$ (combinatorial lemma)
- Part 2 – $O^*(2^{n-|\chi|})$ (dynamic programming)

By optimizing parameters, time is

$$O^*\left(\sum_\chi 2^{n-|\chi|}\right) = O^*(2^n)$$

and space

$$O^*(\max_\chi 2^{n-|\chi|}) = O^*(\sqrt{2^n})$$

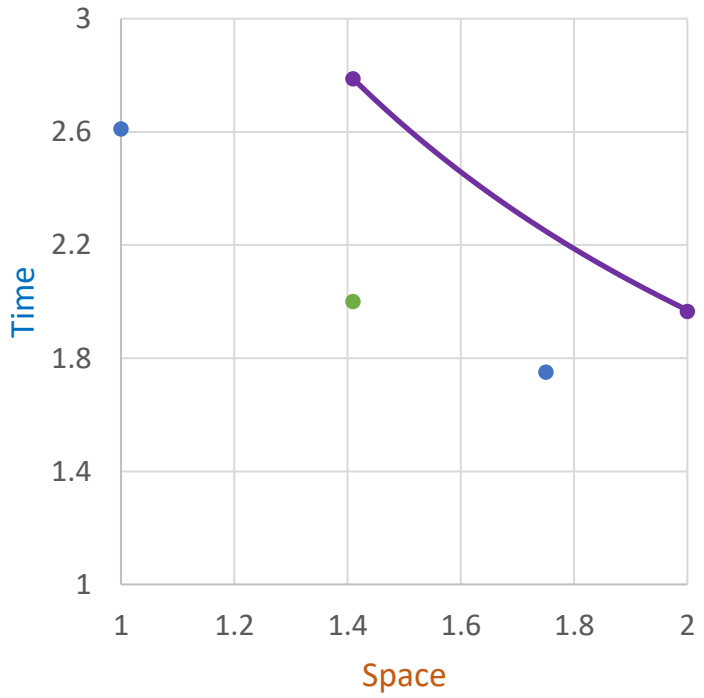
Improves previous best results for treewidth by Koivisto and Parviainen, which give a tradeoff

$$\text{Time} \cdot \text{Space} \approx 3.93^n,$$

where $T \geq 2^n$ and $\sqrt{2^n} \leq S \leq 2$.

[Koivisto & Parviainen 10]

Not specific for treewidth!



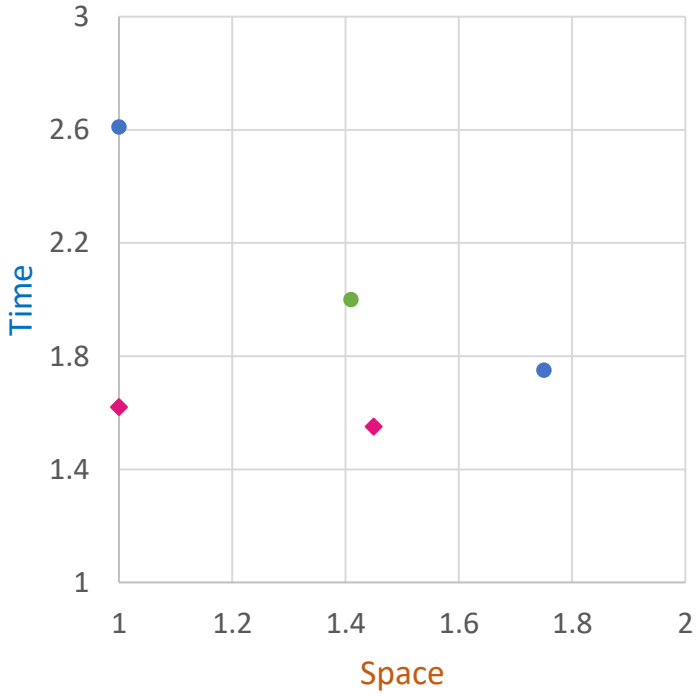
# Improvements

Quantumly:

- Part 1 – $O^*(\sqrt{\#\chi})$ (combinatorial lemma + Grover's search)
- Part 2
  - $O^*(2^{n-|\chi|})$ time / $O^*(1)$ space (divide & conquer + Grover's search)

  Result: $O^*(1.62^n)$ time / $O^*(1)$ space
  - $O^*(1.81^{n-|\chi|})$ time / $O^*(1.81^{n-|\chi|})$ space (quantum dynamic programming)

  Result: $O^*(1.55^n)$ time / $O^*(1.45^n)$ space

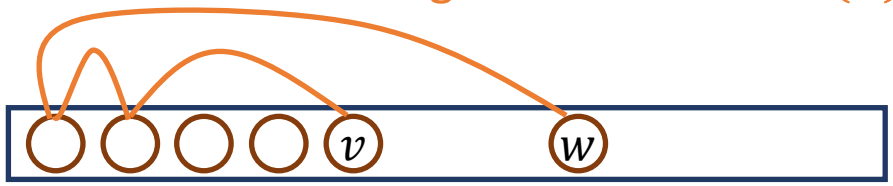


# Improved quantum algorithm

Result: $O^*(1.55^n)$ time / $O^*(1.45^n)$ space

Space and time are not balanced! Can we trade space for time?

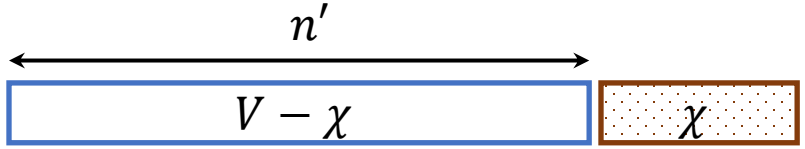
- Treewidth as a vertex ordering problem:
  - $R_v(\pi) = |\{w \mid \pi(w) > \pi(v)$ & there is a path from $w$ to $v$ only through vertices $u$ s.t. $\pi(u) < \pi(v)\}|$




  - Treewidth$(G) = \min_{\pi} \max_{v} R_v(\pi)$

# $O^*(1.81^n)$ quantum dynamic programming

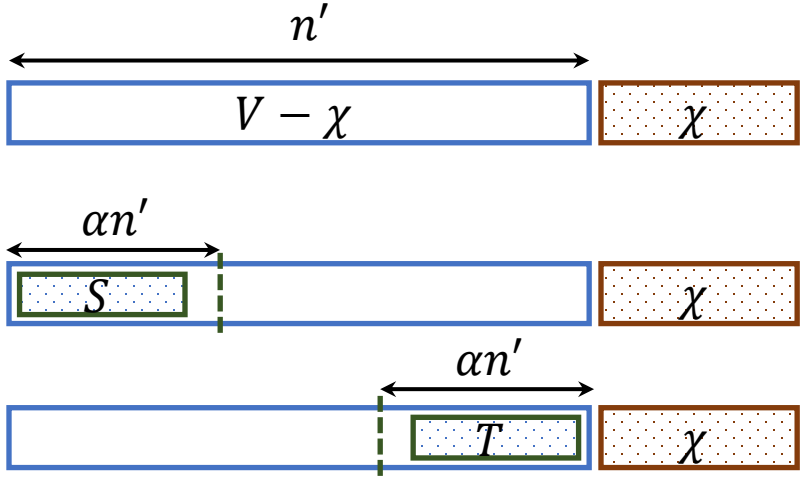
Calculate best ordering for fixed suffix $\chi$:



# $O^*(1.81^n)$ quantum dynamic programming

Calculate best ordering for fixed suffix $\chi$:

1. Precalculation:
   - Calculate best ordering for all $\{S \subset V - \chi \mid |S| \leq \alpha n'\}$ as a prefix using DP
   - Calculate best ordering for all $\{T \subset V - \chi \mid |T| \leq \alpha n'\}$ as a suffix before $\chi$ using DP

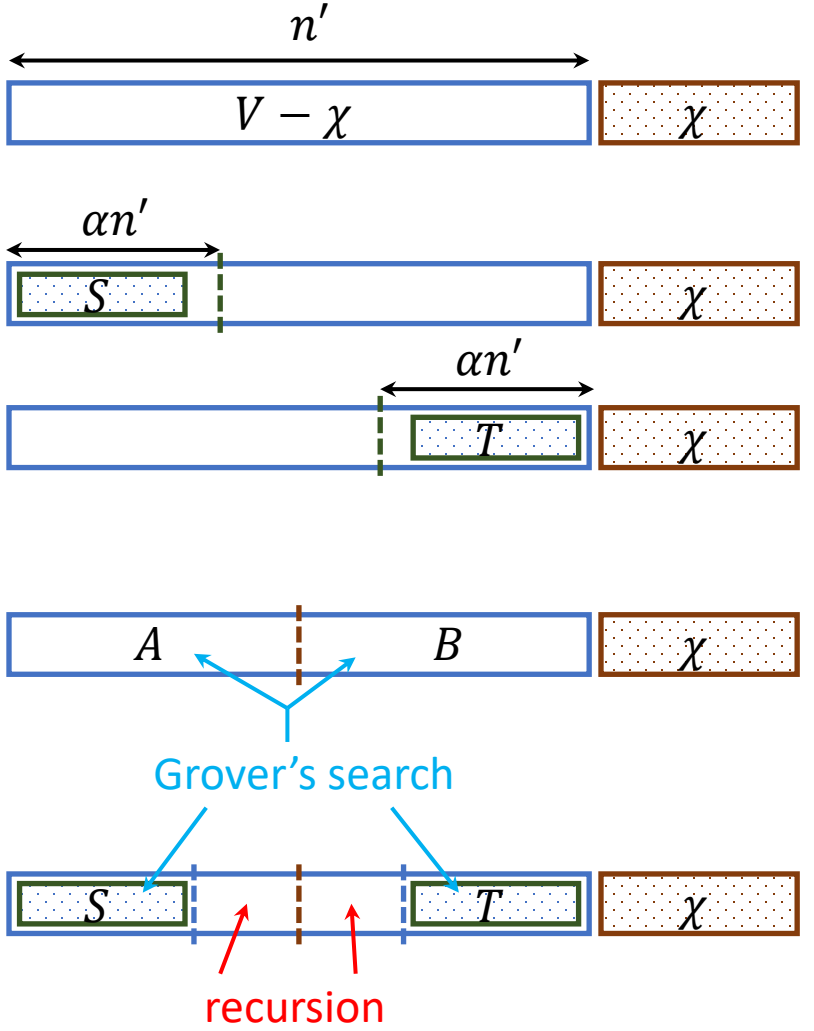


# $O^*(1.81^n)$ quantum dynamic programming

Calculate best ordering for fixed suffix $\chi$:

1. Precalculation:
   - Calculate best ordering for all $\{S \subset V - \chi \mid |S| \leq \alpha n'\}$ as a prefix using DP
   - Calculate best ordering for all $\{T \subset V - \chi \mid |T| \leq \alpha n'\}$ as a suffix before $\chi$ using DP

2. Use Grover's search to examine all partitions $A \cup B = V - \chi$ such that $|A| = |B| = \frac{n'}{2}$.
   - Use Grover's search to examine all possible candidates for prefix $S$ and suffix $T$
   - Solve remaining ordering recursively

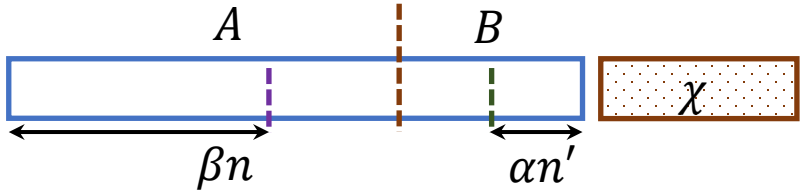


# Asymmetric quantum dynamic programming

- Observation: the ordering of $S$ as a prefix does not depend on $\chi$
- Therefore, we can perform a global precalculation for prefixes $\{S \subset V \| |S| \leq \beta n\}$:


- For those $\chi$ such that $\beta n > \alpha n'$, we have precalculated more!


- Therefore, the new running time must be less than $O^*(1.81^{n'})$!
- The partition $A \cup B = V - \chi$ is now chosen asymmetrically, as well as the suffix parameter $\alpha$:



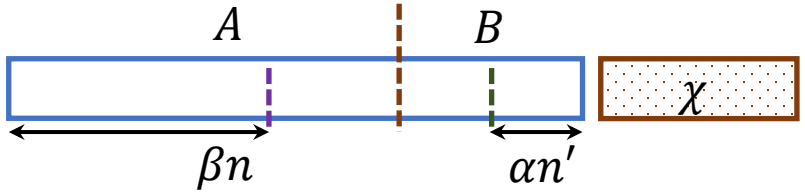
# Asymmetric quantum dynamic programming

- Observation: the ordering of $S$ as a prefix does not depend on $\chi$
- Therefore, we can perform a global precalculation for prefixes $\{S \subset V \| |S| \leq \beta n\}$:


- For those $\chi$ such that $\beta n > \alpha n'$, we have precalculated more!

Not true for suffixes T!

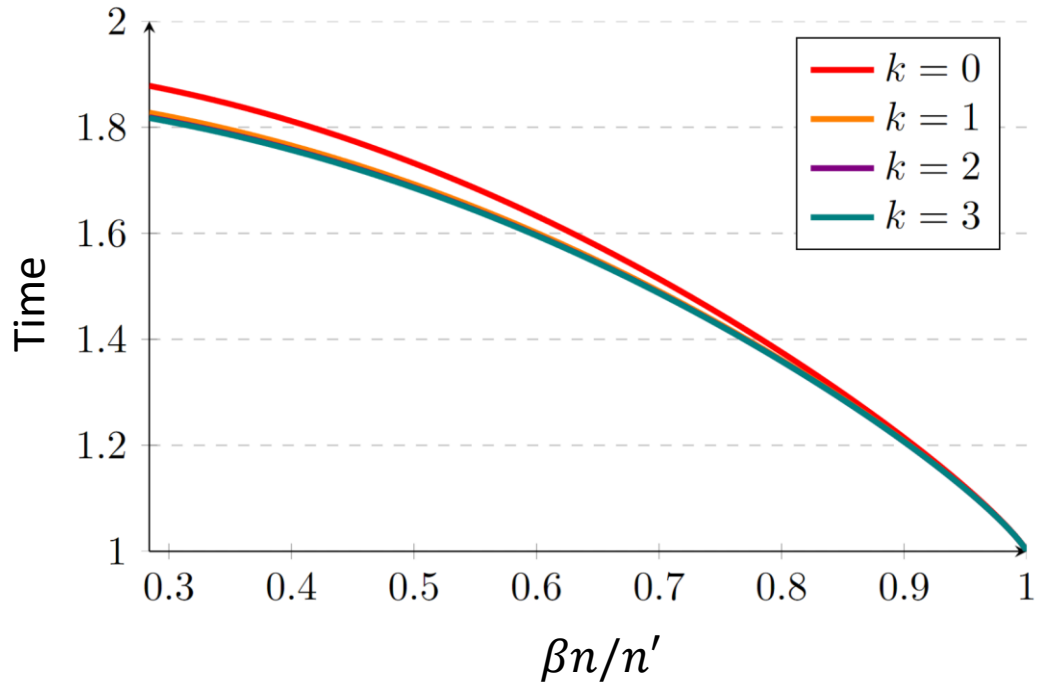

- Therefore, the new running time must be less than $O^*(1.81^{n'})$!
- The partition $A \cup B = V - \chi$ is now chosen asymmetrically, as well as the suffix parameter $\alpha$:



# Asymmetric quantum dynamic programming

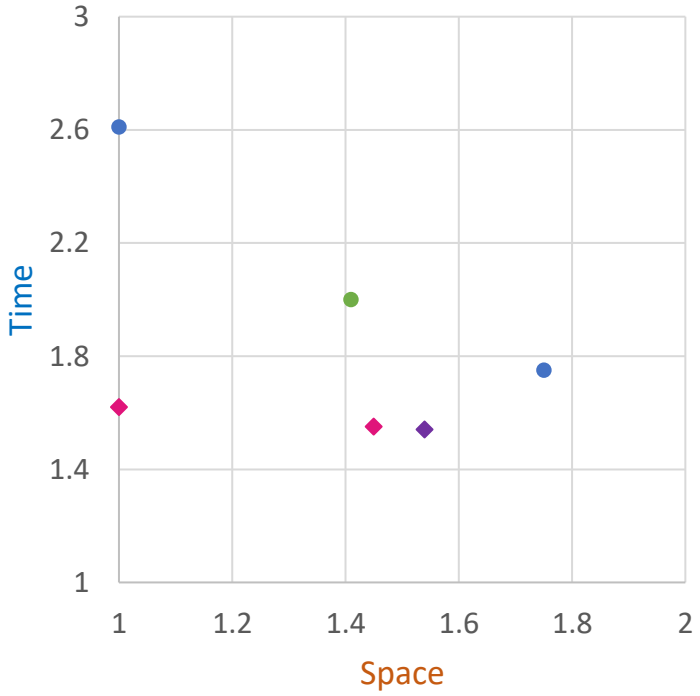
- Complexity depending on $n'$:
- Since $\alpha \approx 0.28$ for the symmetric quantum DP, the improvement is achieved only for $\frac{\beta n}{n'} > \alpha \approx 0.28$

- Main result: a quantum algorithm for treewidth with $O^*(1.54^n)$ time & space



# Conclusions

- Treewidth can be computed more quickly (quantumly, in the QRAM model)
- The quantum dynamic programming of Ambainis et al. is a flexible technique
- The global precalculation does not give anything interesting classically – useful only quantumly!



Thank you! Aitäh! Paldies!

¿Questions?