Quantum Optimization

Stefan Creemers

November 13, 2024

UCLouvain

Optimization 101

- Goal = maximize (or minimize) some objective function given a number of constraints.
- More formally:

  $\max V(\boldsymbol{x})$

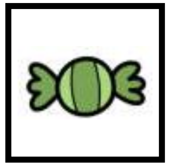
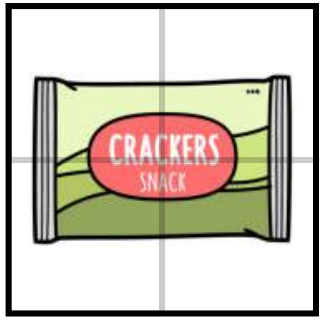
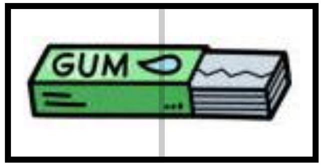
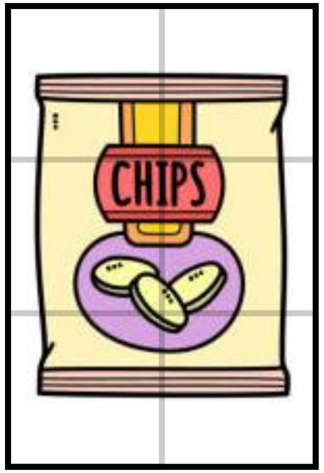
  subject to

  $\boldsymbol{x} \in \Omega$
- Where:
  - $V(\boldsymbol{x})$ is the objective function that returns the value of solution $\boldsymbol{x}$.
  - $\boldsymbol{x} = \{x_1, x_2, \dots, x_n\}$ is a solution that has n decision variables.
  - x_i is the value assigned to decision variable i.
  - Ω is the set of all feasible solutions that do not violate the problem constraints.
- Typical example of an optimization problem: the knapsack problem.

Knapsack problem

$$\max \sum_{i=1}^{n} v_i x_i$$

$$\text{subject to: } \sum_{i=1}^{n} w_i x_i \leq W$$

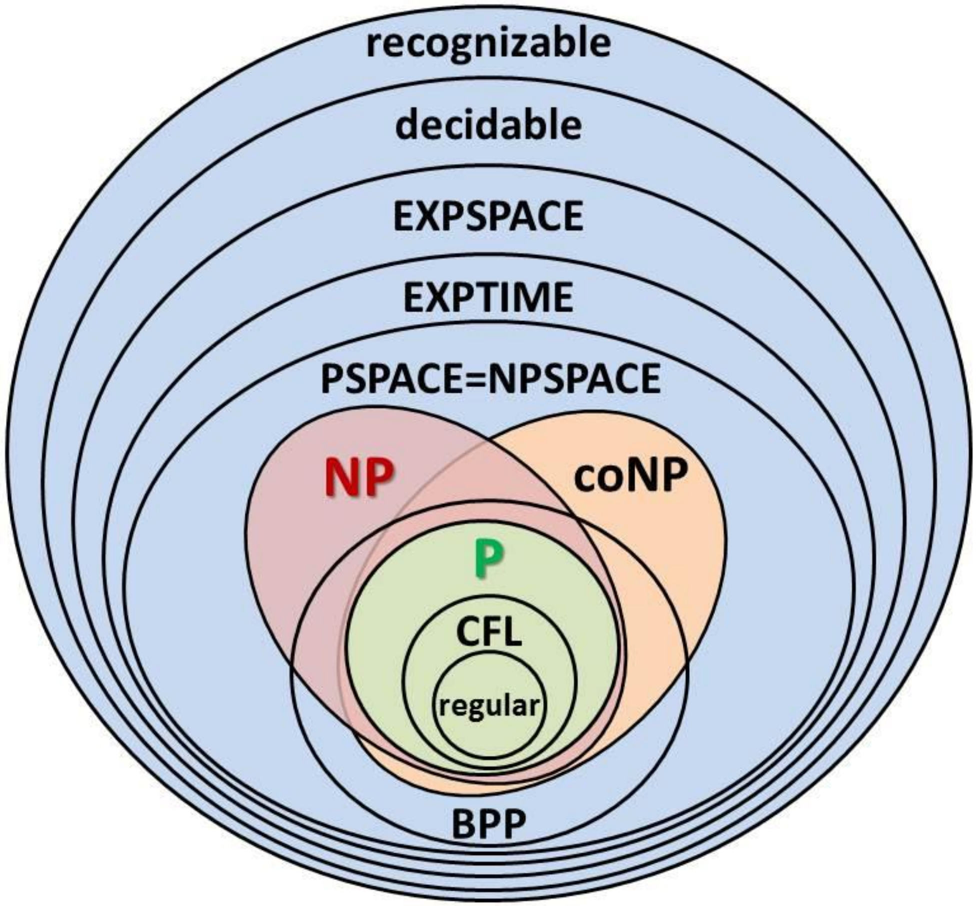
1	1	1	0	1
2€	6€	4€	2€	8€
				
1	4	2	6	4

$$V(\boldsymbol{x}) = 20€$$

Lots of (optimization) problems…

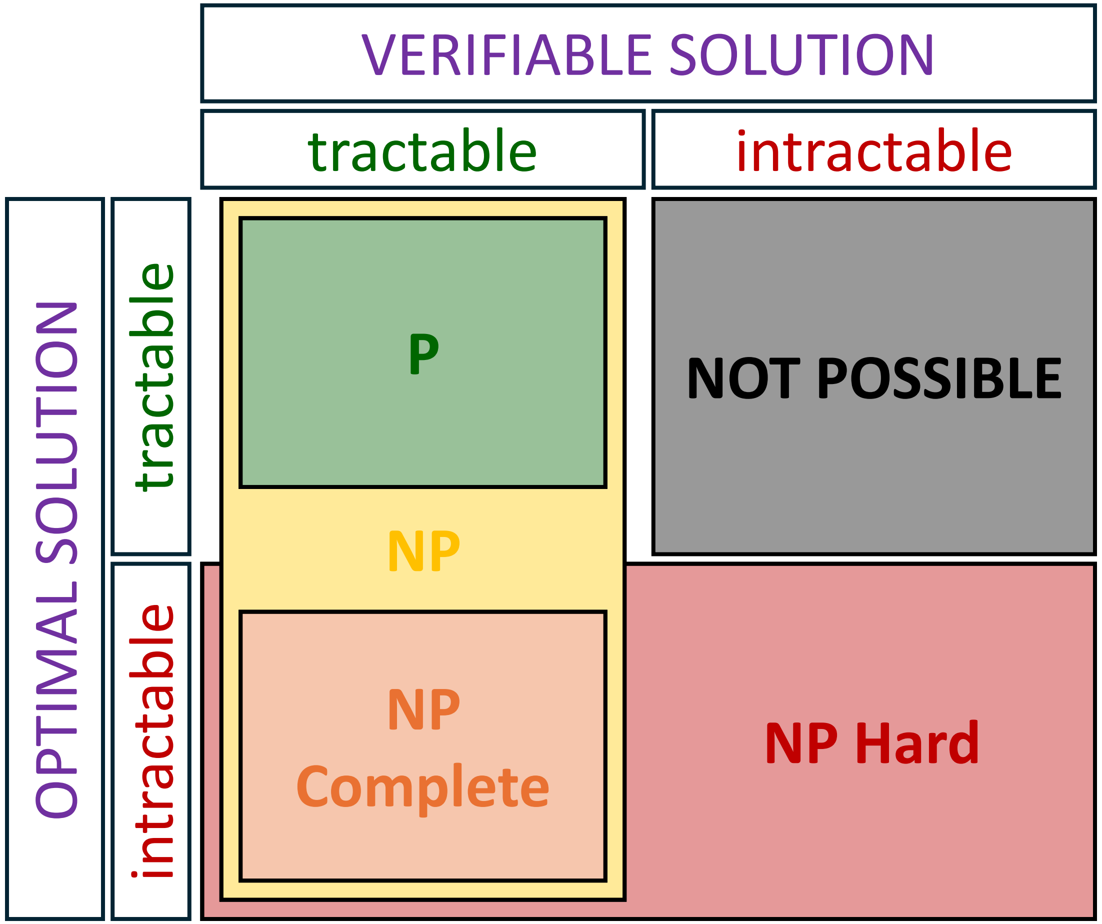
➔ lots of problem classes…

- Problems can be classified based on their « computational complexity ».
- A number of complexity classes are or particular interest:
  - **P**
  - **NP**
  - **NP Complete**
  - **NP Hard**

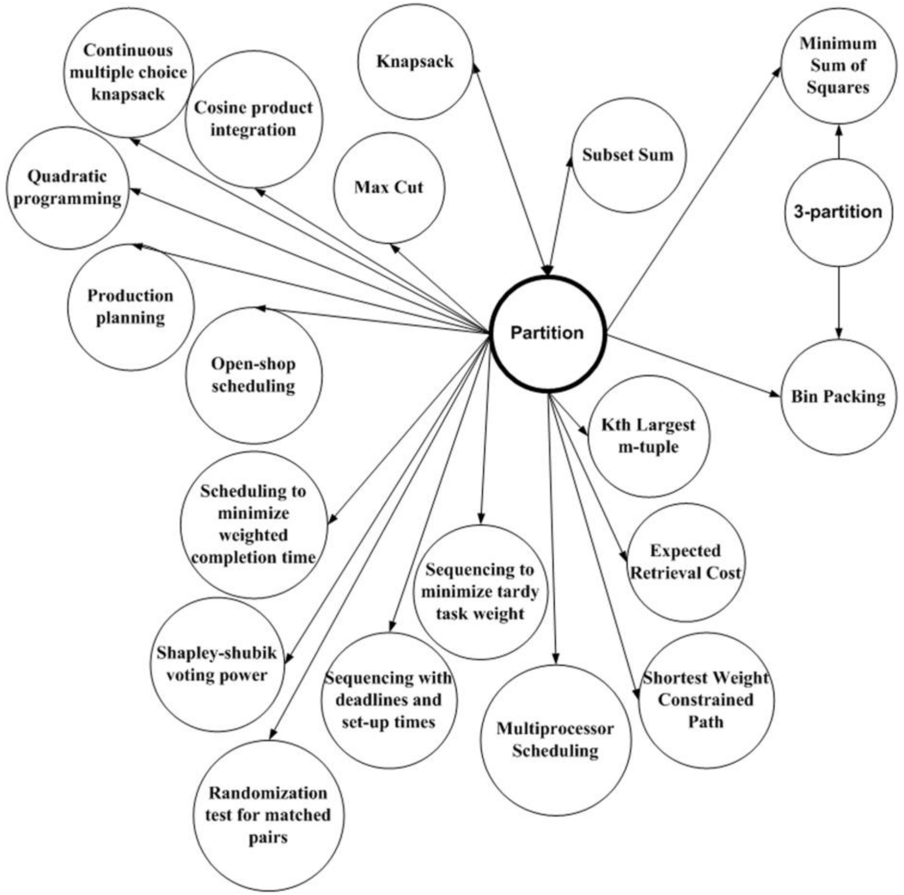


Computational complexity 101

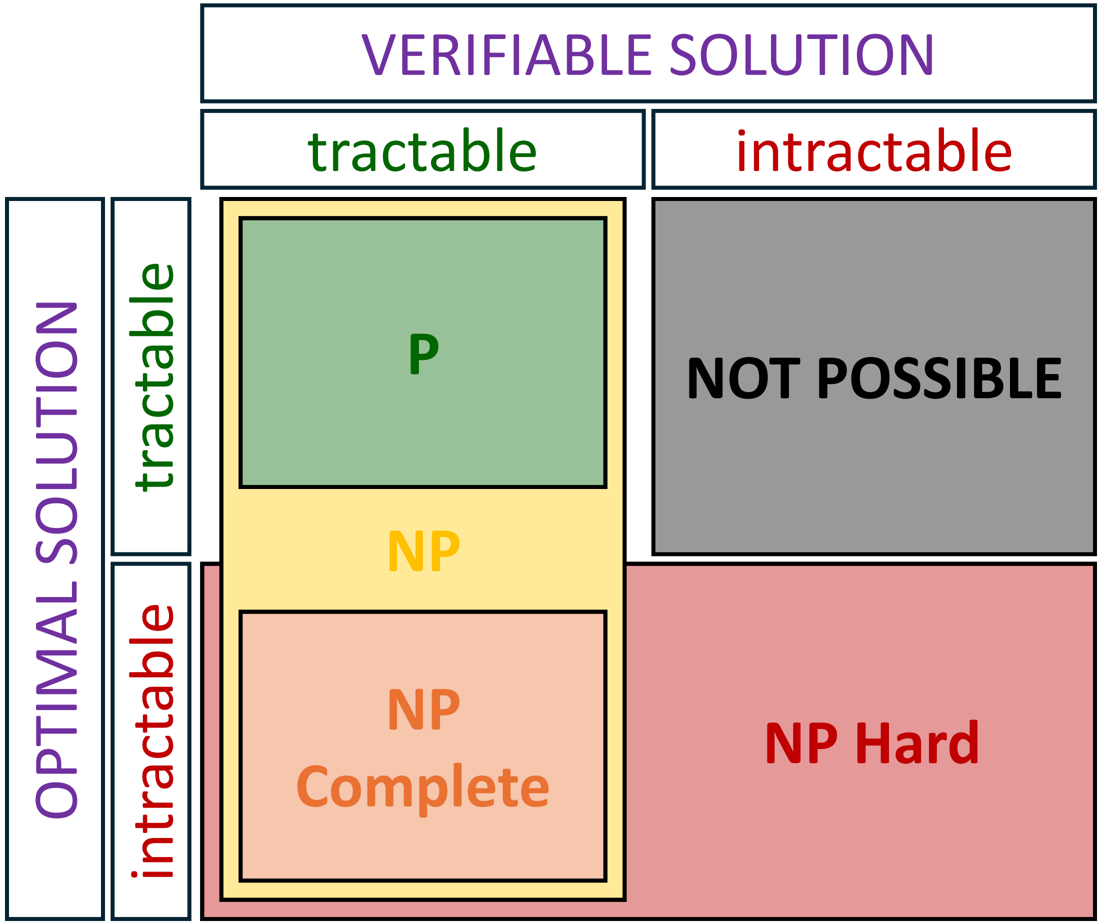
- **P**, **NP**, **NP Complete**, and **NP hard** differ with respect to:
  - Whether a solution is verifiable in polynomial time.
  - Whether an optimal solution can be obtained in polynomial time.
- Polynomial time is considered tractable and requires a less-than exponential running time (e.g., $O(n)$, $O(n^2)$, $O(n^k)$…).
- Many optimization problems are NP Complete/Hard and are considered intractable (i.e., they require running time $O(2^n)$, $O(n!)$, or worse).
- This diagram assumes P ≠ NP.



Example NP-complete optimization problems

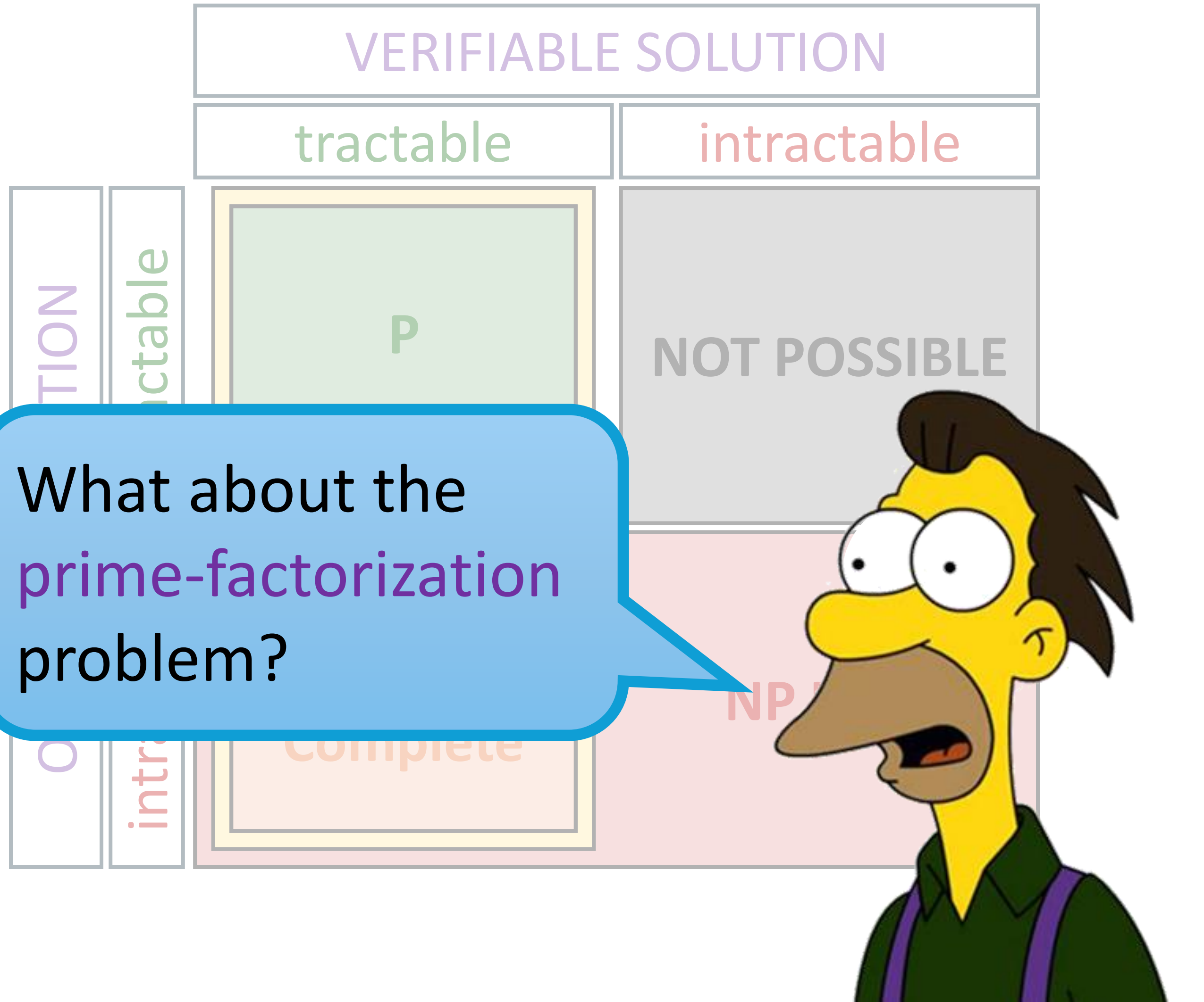


Ruiz-Vanoye et al. (2011). *Survey of polynomial transformations between NP-complete problems*. Journal of Computational and Applied Mathematics.



Example NP-complete optimization problems

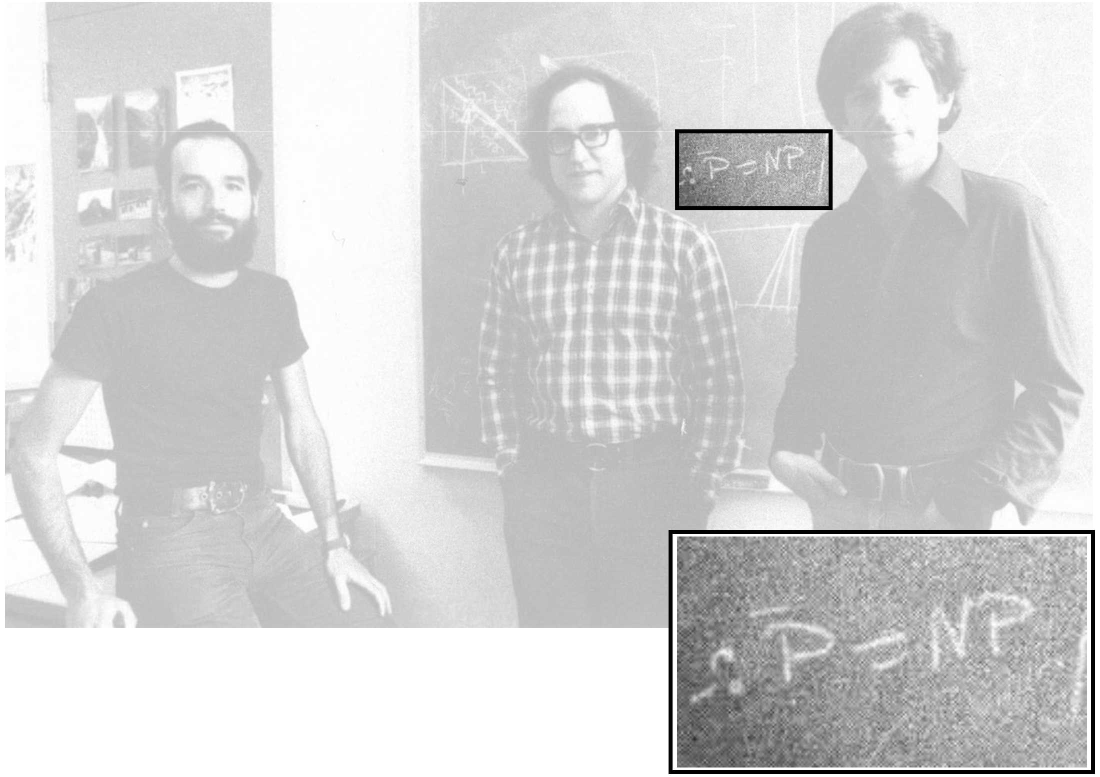
Ruiz-Vanoye et al. (2011). *Survey of polynomial transformations between NP-complete problems*. Journal of Computational and Applied Mathematics.



Prime-factorization problem

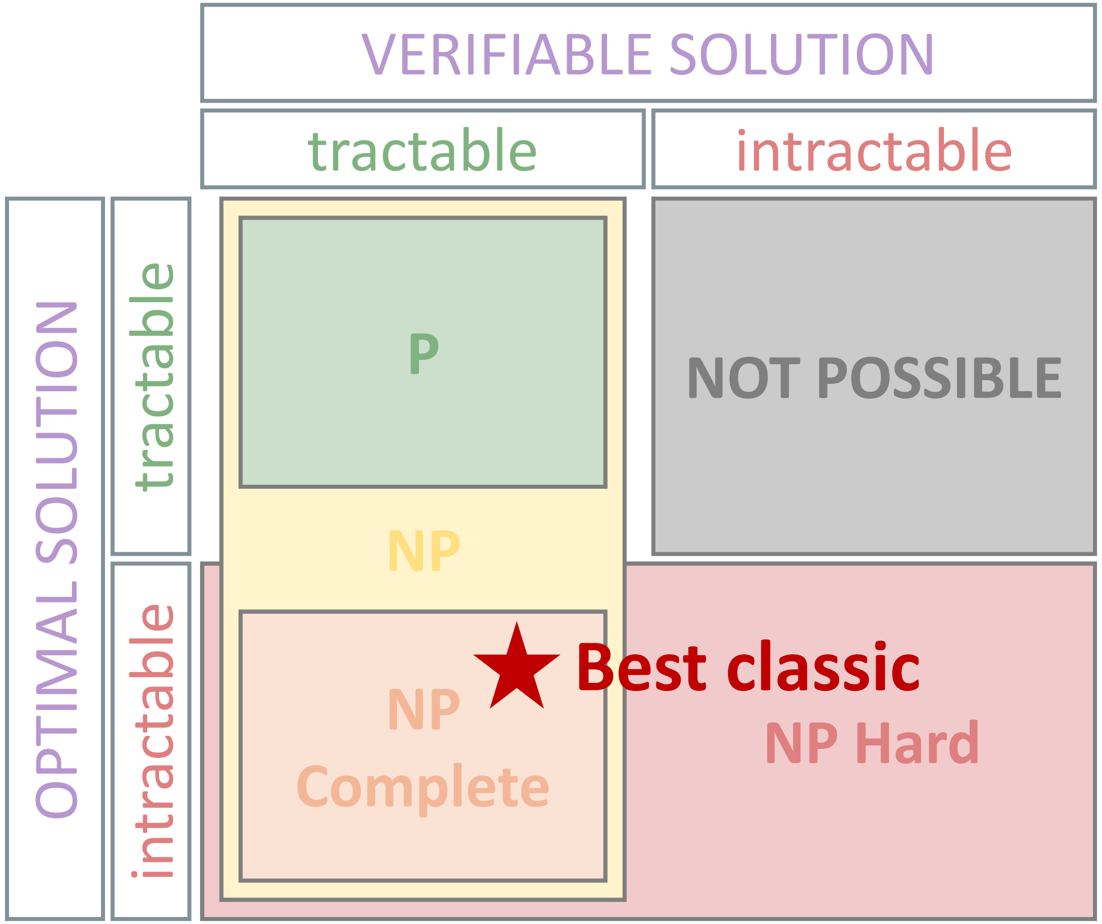
- Most encryption schemes that are used today rely on large primes (e.g., RSA).
- If primes can be factorized efficiently, these encryption schemes can be broken in polynomial time.
- The best-known classical algorithm to factorize primes runs in (sub-)exponential time…

- Rivest, Shamir, & Adleman (RSA)



Prime-factorization problem

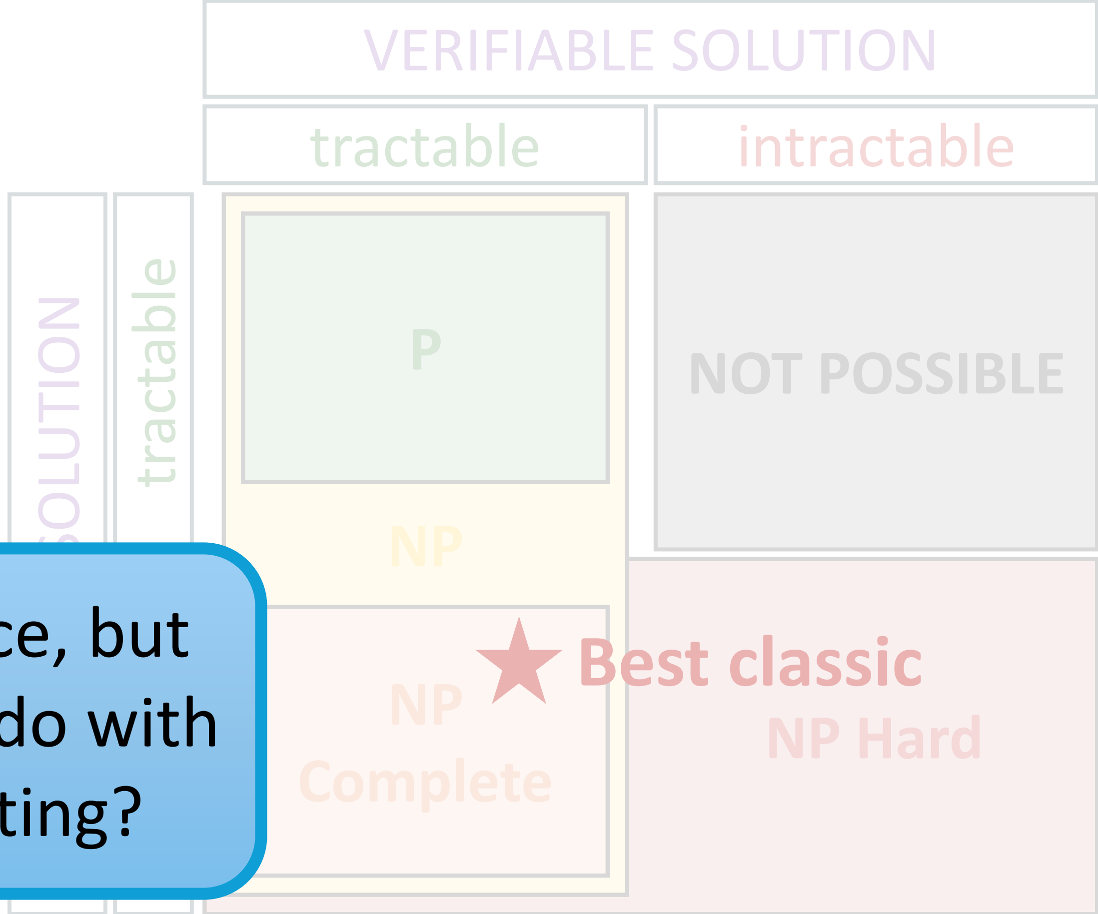
- Most encryption schemes that are used today rely on large primes (e.g., RSA).
- If primes can be factorized efficiently, these encryption schemes can be broken in polynomial time.
- The best-known classical algorithm to factorize primes runs in (sub-)exponential time…



Prime-factorization problem

- Most encryption schemes that are used today rely on large primes (e.g., RSA).
- If primes can be factorized efficiently, these encryption schemes can be broken in ...nomial time
- ...est-know... algori... o fact... ns ...o-)exp...

This is all very nice, but what has this to do with quantum computing?

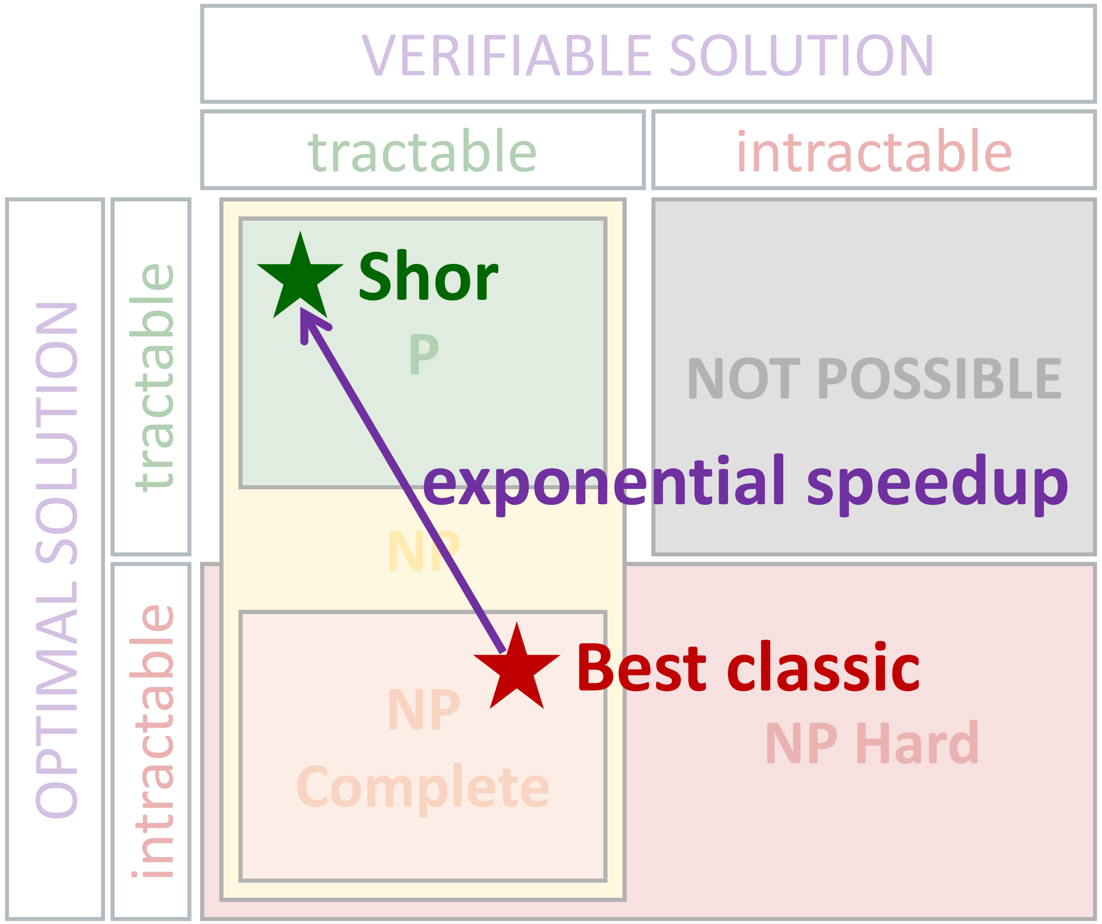


Prime-factorization problem

- In 1994 Shor published a quantum algorithm that can factorize primes in polynomial time, resulting in an exponential speedup!

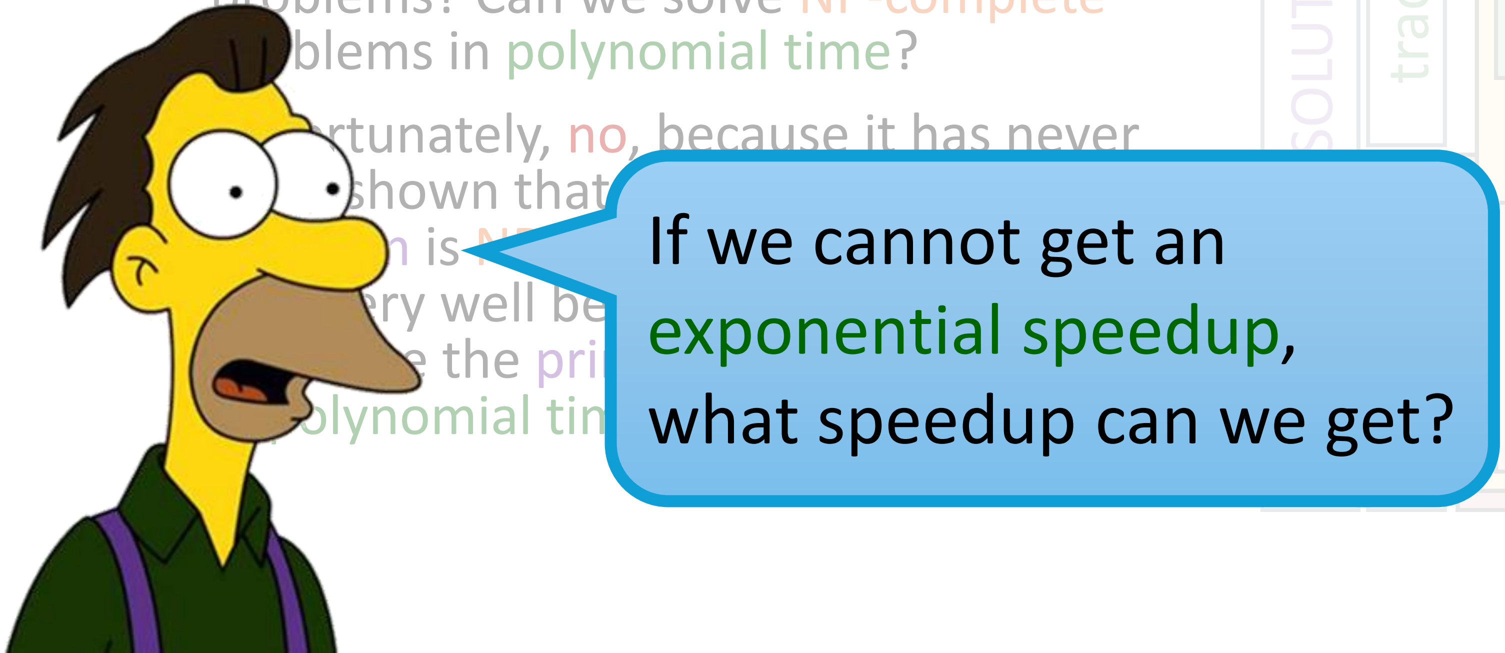
Prime-factorization problem

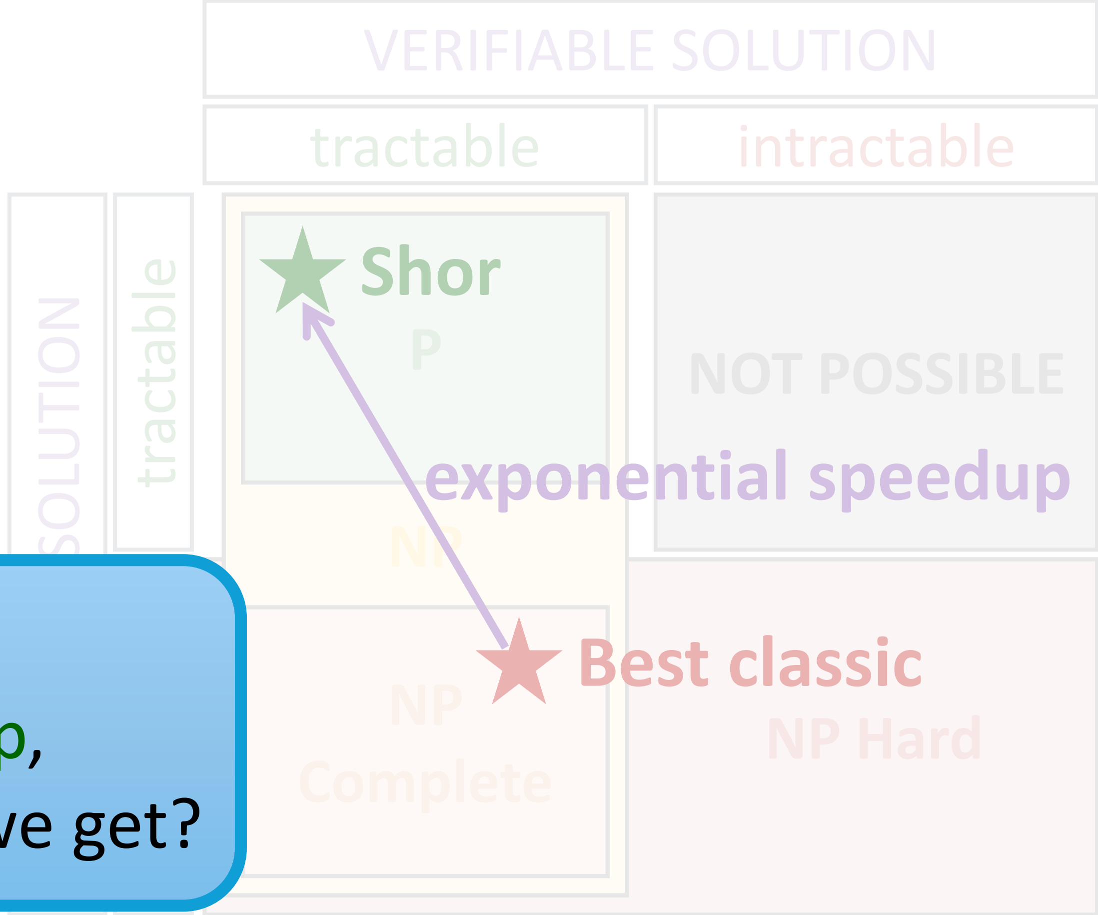
- In 1994 Shor published a quantum algorithm that can factorize primes in polynomial time, resulting in an exponential speedup!
- Does this imply we can get an exponential speedup also when solving NP-complete problems? Can we solve NP-complete problems in polynomial time?
- Unfortunately, no, because it has never been shown that the prime-factorization problem is NP-complete… In fact, there may very well be a classical algorithm that can solve the prime-factorization problem in polynomial time.



Prime-factorization problem

- In 1994 Shor published a quantum algorithm that can factorize primes in polynomial time, resulting in an exponential speedup!
- Does this imply we can get an exponential speedup also when solving NP-complete problems? Can we solve NP-complete ...blems in polynomial time?
- ...rtunately, no, because it has never ... shown that ... n is N... ...ry well be ... e the pri... ...olynomial tin...





A true Lov story

- Imagine yesterday Carl went out yesterday and he met the girl of his dreams!
- It gets even better: he got her phone number!
- Unfortunately, however, Carl completely forgot the name of his dream girl…

A true Lov story

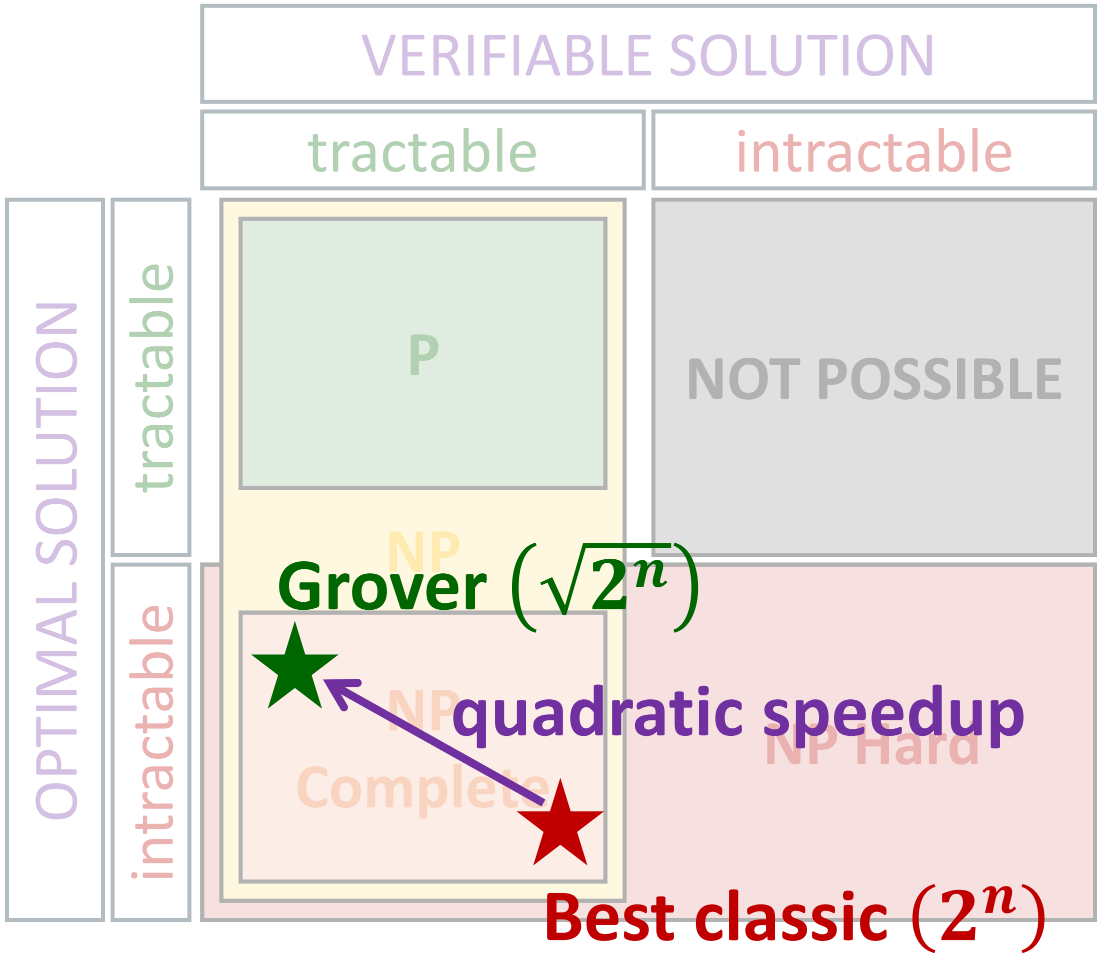
- Imagine yesterday Carl went out yesterday and he met the girl of his dreams!
- It gets even better: he got her phone number!
- Unfortunately, however, Carl completely forgot the name of his dream girl…
- Luckily, Carl has a phone book!
- The phone book, however, is ordered by name, not by phone number.
- As Carl is determined to find the name of his dream girl, he starts browsing the phone book.
- As there are 2^n entries in the phone book, in the worst-case, Carl will have to check 2^n phone numbers. This might take a while…

Lov Grover to the rescue!

- In 1996, Grover published an algorithm that can find a target entry in an unstructured database that has 2^n entries by looking at only $\sqrt{2^n}$ entries.
- Compared to a classical approach (that requires to look at all 2^n entries), this results in a quadratic speedup!

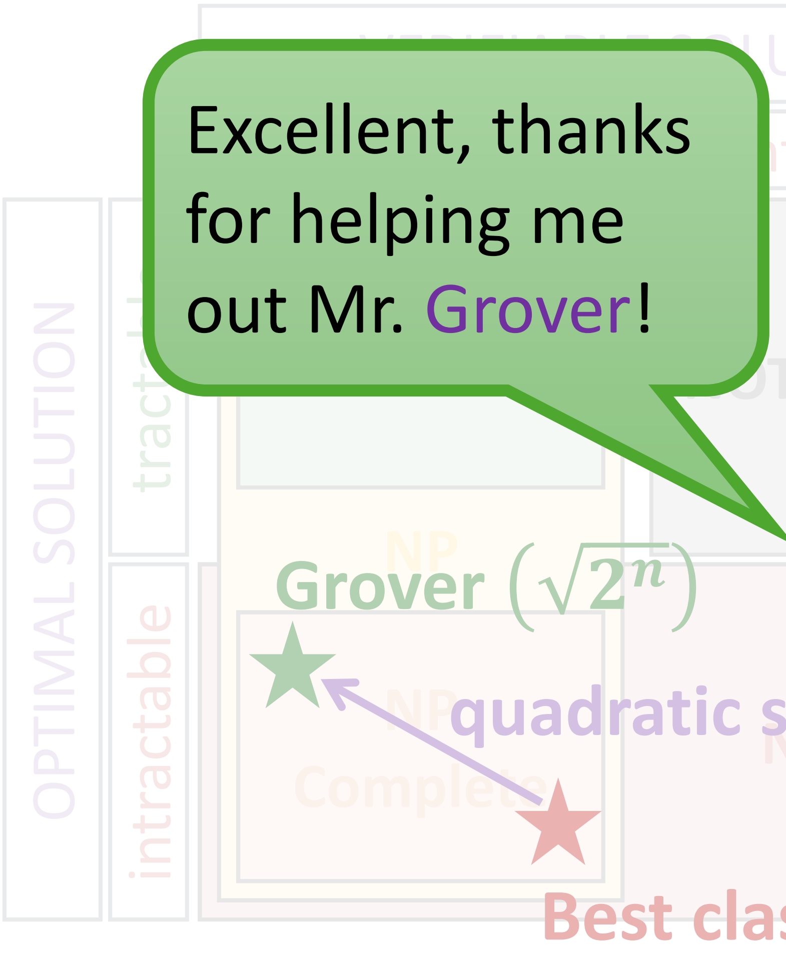
Lov Grover to the rescue!

- In 1996, Grover published an algorithm that can find a target entry in an unstructured database that has 2^n entries by looking at only $\sqrt{2^n}$ entries.
- Compared to a classical approach (that requires to look at all 2^n entries), this results in a quadratic speedup!
- It has been shown that this speedup is optimal.



Lov Grover to the rescue!

- In 1996, Grover published an algorithm that can find a target entry in an unstructured database that has 2^n entries by looking at only $\sqrt{2^n}$ entries.
- Compared to a classical ... proach (that requires to look ... 2^n entrie... ...ratic s... ... been sh... ...eedup is op...



Lov Grover to the rescue!

The key takeaway here is that we may not get an exponential speedup (as was the case with Shor), but we can perhaps hope for a quadratic speedup when solving NP-Complete/Hard optimization problems.

Ok, that's nice, however, even with a quadratic speedup, an NP-Complete problem is still NP Complete… Also, these results are almost 30 years old! Where do we stand now?

The NISQ era (Noisy Intermediate-Scale Quantum)

- Two big approaches for quantum computing:
  - Adiabatic quantum computing. Used by machines (quantum annealers) that have a single purpose: optimization! Example: D-Wave.
  - Gate-based universal quantum computing. Example: IBM. Note that a universal quantum computers is superior to a classical computer as any classical operation can be performed on a quantum computer with a polynomial overhead.
- Limitations shared by NISQ-era quantum computers:
  - Small number of qubits ➔ limited problem size.
  - Decoherence ➔ limited calculation time.
  - Noise ➔ limited accuracy/precision.

The NISQ era (Noisy Intermediate-Scale Quantum)

- Two big approaches for quantum computing:
  - Adiabatic quantum computing. Used by machines (quantum annealers) that have a single purpose: optimization! Example: D-Wave.
  - Gate-based universal quantum computing. Example: IBM. Note that a universal quantum computers is superior to a classical computer as any classical operation can be performed on a quantum computer with a polynomial overhead.
- Limitations shared by NISQ-era quantum computers:
  - Small number of qubits ➔ limited problem size.
  - Decoherence ➔ limited calculation time.
  - Noise ➔ limited accuracy/precision.

Adiabatic quantum computing

- Limitations of current quantum annealers:
  - Original optimization problem needs to be transformed to a problem that is understood by the quantum annealer (e.g., a QUBO).
  - QPU topology requires embedding.
  - Limited qubit connectivity.
  - Chains are required to connect qubits.
- Selected research:
  - Pérez Armas, Creemers, & Deleplanque. (2024). *Solving the resource constrained project scheduling problem with quantum annealing*, Nature Scientific Reports.
  - Deleplanque, Creemers, & Pérez Armas. (2024). *solQHealer: quantum procedures for rendering infeasible solutions feasible*.
  - Pérez Armas, Deleplanque, Aggoune, & Creemers (2024). *A quantum hybrid column-generation heuristic*.

Adiabatic quantum computing

(selected research 1)



- Pérez Armas, Creemers, & Deleplanque. (2024). *Solving the Resource Constrained Project Scheduling Problem (RCPSP) with quantum annealing*, Nature Scientific Reports.
- What: we use quantum annealing to compare 12 well-known classical formulations for solving the RCPSP.
- Key take-away: formulations that work well on classical computers do not necessarily work well on quantum computers. In fact, on a quantum computer, the oldest formulation (which required the least number of qubits) had the best performance.

Adiabatic quantum computing

(selected research 2)

- Deleplanque, Creemers, & Pérez Armas. (2024). *solQHealer: quantum procedures for rendering infeasible solutions feasible*.
- What: we use (reverse) quantum annealing to solve the Maximum Independent Set (MIS) problem as well as 3-SAT.
- Key take-away: (reverse) quantum annealing may be used to quickly repair infeasible solutions/solutions that have become infeasible due to new constraints that have surfaced. This is particularly useful in a setting where fast, online optimization is required (e.g., train scheduling).

Adiabatic quantum computing
(selected research 3)




- Pérez Armas, Deleplanque, Aggoune, & Creemers (2024). *A quantum hybrid column-generation procedure*.
- What: we use a hybrid column-generation procedure to solve the parallel machine scheduling problem as well as the 2-dimensional cutting stock problem.
- Key take-away: quantum annealers excel in rapidly generating many (good) solutions. These solutions may, for instance, be introduced as new columns in a hybrid column generation procedure (where the master problem is solved by a classical computer).

The NISQ era (Noisy Intermediate-Scale Quantum)

- Two big approaches for quantum computing:
  - Adiabatic quantum computing. Used by machines (quantum annealers) that have a single purpose: optimization! Example: D-Wave.
  - Gate-based universal quantum computing. Example: IBM. Note that a
    …mputer as any
    …puter with a

Lessons learned:

1. Don’t expect that what works in the classical world also works in a quantum world.
2. Current machines may not yet be advanced enough to find optimal solutions for big, real-life problems. However, they can already be used as fast heuristics or in hybrid procedures that need fast (but good) solutions.

The NISQ era (Noisy Intermediate-Scale Quantum)

- Two big approaches for quantum computing:
  - Adiabatic quantum computing. Used by machines (quantum annealers) that have a single purpose: optimization! Example: D-Wave.
  - Gate-based universal quantum computing. Example: IBM. Note that a universal quantum computers is superior to a classical computer as any classical operation can be performed on a quantum computer with a polynomial overhead.
- …ations shared by NISQ era quantum computers:
  - …all n… size.
  - …oherer…
  - Noise ➔ li…

What would happen if we have a noiseless universal quantum computer?

Universal quantum computing

- A noiseless universal quantum computer does not exist (yet)…
- However, we can simulate one!
- We coded an ideal simulator to assess the performance of future quantum optimization algorithms.
- Selected research:
  - Creemers & Pérez Armas. (2024). *Discrete optimization: a quantum revolution*?
  - Creemers & Pérez Armas. (2023). *Limitations of existing quantum algorithms*.
  - Creemers. (2024). *Speeding up Grover's algorithm*.

Universal quantum computing

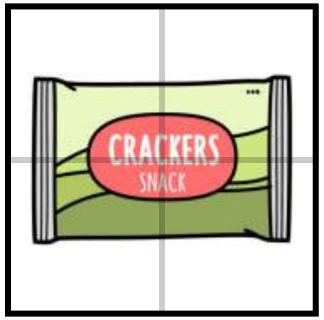
(selected research 1)

- Creemers & Pérez Armas. (2024). *Discrete optimization: a quantum revolution*?
- What: we develop several Grover-based quantum procedures for solving discrete optimization problems.
- Key takeaways:
  - We can solve any discrete optimization problem using $O\left(\mu\sqrt{2^{bn}}\right)$ operations, where μ is the number of operations required to verify a solution & 2^b is the number of discrete values that can be assigned to any of the n decision variables.
  - Our procedures can be used as general-purpose solvers (similar to CPLEX and Gurobi) but also as heuristics.
  - We present a hybrid Branch-and-Bound (B&B) procedure that expects to visit $O\left(\sqrt{2^n}\right)$ nodes. In contrast, in the worst case, a classical B&B visits $O(2^n)$ nodes ➔ we achieve a quadratic speedup!
  - We demonstrate that our procedures can match the worst-case performance of the best classical algorithms that solve the knapsack problem.
  - For quadratic knapsack problems we outperform the best classical algorithms.

Quadratic knapsack

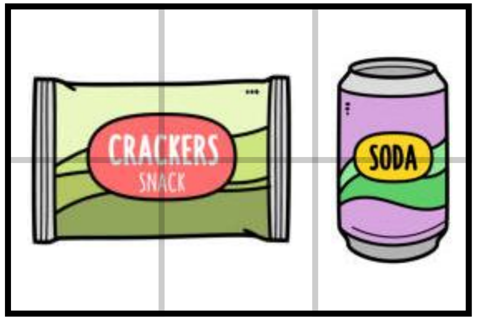
$$\max \sum_{i=1}^{n} \left(v_i x_i + \sum_{j=1}^{n} v_{ij} x_i x_j \right)$$

$v_2 = 6€$



$v_6 = 2€$

$v_{26} = 4€$



Universal quantum computing

(selected research 2)

- Creemers & Pérez Armas. (2023). *Limitations of existing quantum algorithms*.
- What: we investigate whether we can use quantum counting, nested quantum search, and amplitude amplification for solving optimization problems.
- Key takeaways:
  - When effectively implementing these quantum algorithms, several challenges need to be overcome.
  - They do not allow to outperform Vanilla Grover; they do not allow a better-than quadratic speedup.

Universal quantum computing

(selected research 3)

- Creemers. (2024). *Speeding up Grover's algorithm*.
- What: we assess the (expected) speedup when running Grover's algorithm in series, parallel, and in series/parallel.
- Key takeaways:
  - The expected runtime can be reduced by almost 13% if we run Grover's algorithm in series.
  - The parallel execution of Grover's algorithm yields only a square-root speedup in the number of QPUs (e.g., if we use 4 QPUs, we do not obtain a linear speedup of factor 4, but expect only a square-root speedup of factor $\sqrt{4} = 2$). The reason being that we deal with a probabilistic process (as opposed to classical procedures that are often deterministic).

Some conclusions…

- As quantum algorithms often have probabilistic outcomes, it may be more challenging to obtain a linear speedup from parallel computing.
- Quantum algorithms may be used as general-purpose solvers, exact (hybrid) procedures, or heuristics.
- Compared to classical computing we can obtain up to a quadratic speedup when solving NP-Complete/Hard optimization problems. Even though these problems remain NP-Complete/Hard, a quadratic speedup is still very interesting from a practical point of view.
- Quantum computers excel at solving complex problems that have non-linear objective functions and/or constraints (in contrast to classical computers). Take for instance the quadratic knapsack problem.
- Once we have universal quantum computing, quantum algorithms will revolutionize the field of optimization!

However, we may need to be patient for a bit longer…

Email: sc@cromso.com