

Hybrid Quantum-Classical Optimization with the ProvideQ Toolbox

Domenik Eichhorn – domenik.eichhorn@kit.edu



Speaker Portfolio



Domenik Eichhorn

PhD Student at the Chair for
*„Test, Validation and Analysis
of Software-Intensive Systems“*

Research Topics:

Quantum Software and Hybrid Algorithms

Contact:

domenik.eichhorn@kit.edu

Motivation – Quantum Supremacy

- **In theory**, quantum computers can utilize *quantum mechanical effects* to compute specific tasks faster than classical computers.

Use Cases:

- Cryptography
- Material Simulation
- Combinatorial Optimization

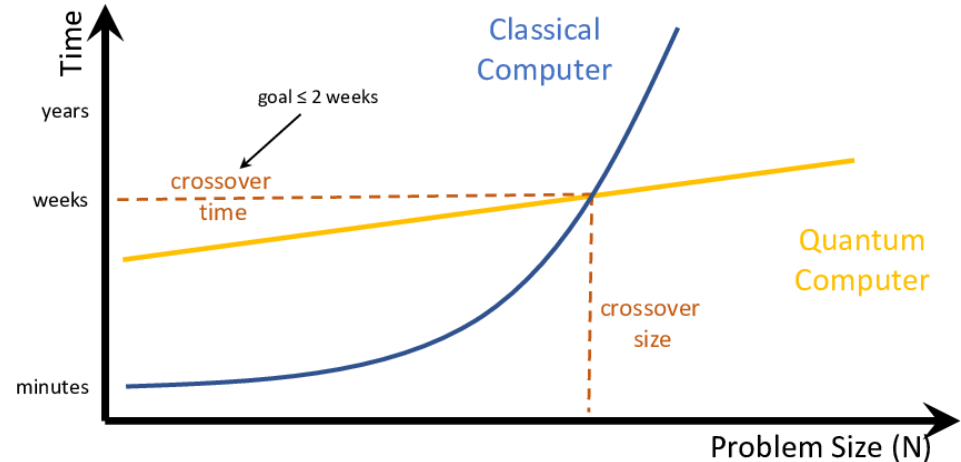


Image Src: T. Hoefler et al., Disentangling Hype from Practicality: On Realistically Achieving Quantum Advantage

Background – NISQ Quantum Hardware

Noisy
Intermediate
Scale
Quantum
(era)

- Noise – a lot ...
- Scale – very small ...
- Quantum – sometimes ...



Background – Quantum Algorithms

„Superpolynomial Speedup“:

- Shor → Prime Factorization
- Deutsch-Josza → Check if Function is Constant

„Quadratic Speedup“:

- Grover → Search in unstructured Database

„Whats a speedup?“:

- QAOA → Approximation Algorithm
- VQE → Approximation Algorithm

Algorithms & Advantages in Combinatorial Optimizaiton

„Superpolynomial Speedup“:

- Shor → Prime Factorization (Cryptography)
- Deutsch-Josza → Check if Function is Constant (Useless)

„Quadratic Speedup“:

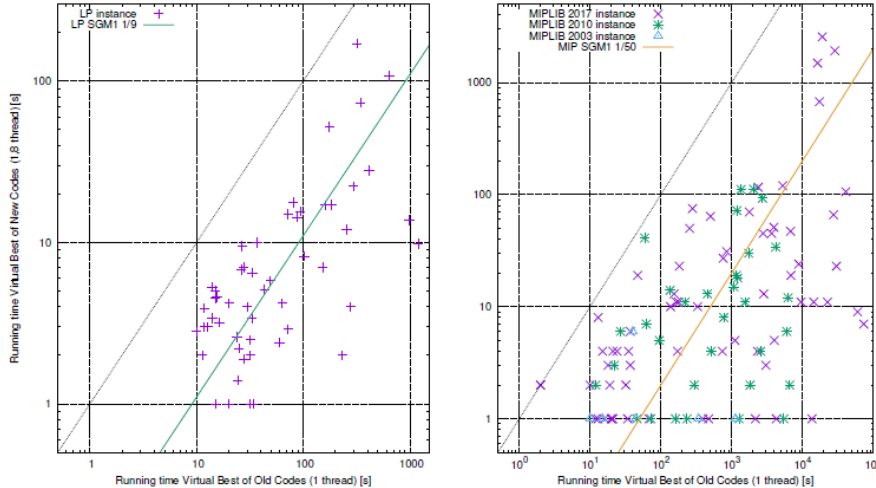
- Grover → Search in unstructured Database

„Whats a speedup?“:

- QAOA → Approximation Algorithm
- VQE → Approximation Algorithm

What do we compete against?

- Classical Solvers have improved drastically in the recent years:



Source: Koch, T., Berthold, T., Pedersen, J., Vanaret, C.: Progress in mathematical programming solvers from 2001 to 2020. EURO Journal on Computational Optimization 10, 100031 (2022)

Example:

Speedup for (Mixed-Integer) Linear Programming Problems:

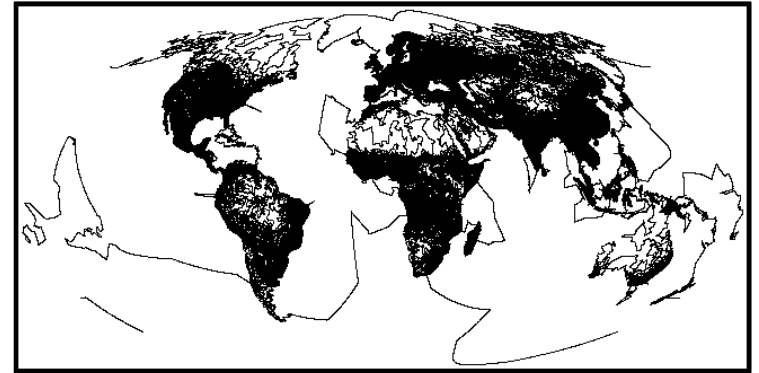
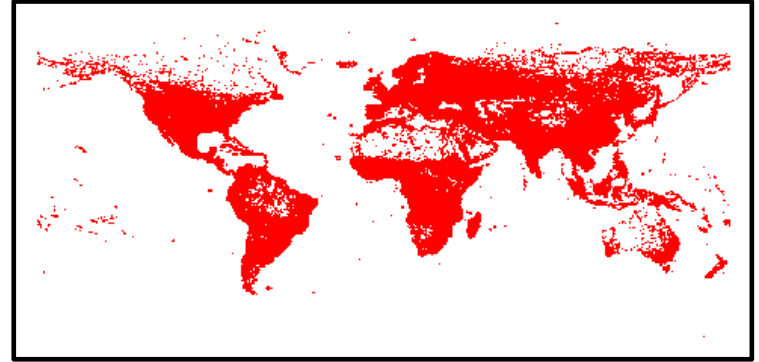
- LP: 9x Speedup
- MIP: 50x Speedup
- Hardware: 20x Speedup
- Combined LP: 180x
- Combined MIP: 1000x

SotA: Traveling Sales Person

Example – World TSP:

- 1,907,711 cities
- Lower Bound: 7,512,218,268
- Current best Solution: 7,515,755,956
(only 0.0471% greater than LB)

Source:
<https://www.math.uwaterloo.ca/tsp/world/>



Low Autocorrelation Binary Sequences (LABS)

- Also an Optimization problems, however, it „breaks“ much earlier.
- Even problems with only 60 variables are very hard to solve.

Consider a sequence $S = (s_1, \dots, s_N)$ with $s_i = \pm 1$. The autocorrelations of S are defined as

$$C_k(S) = \sum_{i=1}^{N-k} s_i s_{i+k} \quad (1)$$

for $k = 0, 1, \dots, N-1$, and the “energy” of S is defined as the sum of the squares of all off-peak correlations,

$$E(S) = \sum_{k=1}^{N-1} C_k^2(S). \quad (2)$$

The *low-autocorrelation binary sequence* (LABS) problem is to find a sequence S of given length N that minimizes $E(S)$ or, equivalently, maximizes the *merit factor*

$$F(S) = \frac{N^2}{2E(S)}. \quad (3)$$

Source: Tom Packebusch & Stephan Mertens
Low Autocorrelation Binary Sequences
<https://arxiv.org/abs/1512.02475>

PHYSICS

Evidence of scaling advantage for the quantum approximate optimization algorithm on a classically intractable problem

Ruslan Shaydulin^{1*}, Changhao Li¹, Shouvanik Chakrabarti¹, Matthew DeCross², Dylan Herman¹, Niraj Kumar¹, Jeffrey Larson³, Danylo Lykov^{1,4}, Pierre Minssen¹, Yue Sun¹, Yuri Alexeev⁴, Joan M. Dreiling², John P. Gaebler², Thomas M. Gatterman², Justin A. Gerber², Kevin Gilmore², Dan Gresh², Nathan Hewitt², Chandler V. Horst², Shaohan Hu¹, Jacob Johansen², Mitchell Matheny², Tanner Mengle², Michael Mills², Steven A. Moses², Brian Neyenhuis², Peter Siegfried², Romina Yalovetzky¹, Marco Pistoia¹

The quantum approximate optimization algorithm (QAOA) is a leading candidate algorithm for solving optimization problems on quantum computers. However, the potential of QAOA to tackle classically intractable problems remains unclear. Here, we perform an extensive numerical investigation of QAOA on the low autocorrelation binary sequences (LABS) problem, which is classically intractable even for moderately sized instances. We perform noiseless simulations with up to 40 qubits and observe that the runtime of QAOA with fixed parameters scales better than branch-and-bound solvers, which are the state-of-the-art exact solvers for LABS. The combination of QAOA with quantum minimum finding gives the best empirical scaling of any algorithm for the LABS problem. We demonstrate experimental progress in executing QAOA for the LABS problem using an algorithm-specific error detection scheme on Quantum trapped-ion processors. Our results provide evidence for the utility of QAOA as an algorithmic component that enables quantum speedups.

More Background – Implementing Quantum Algorithms

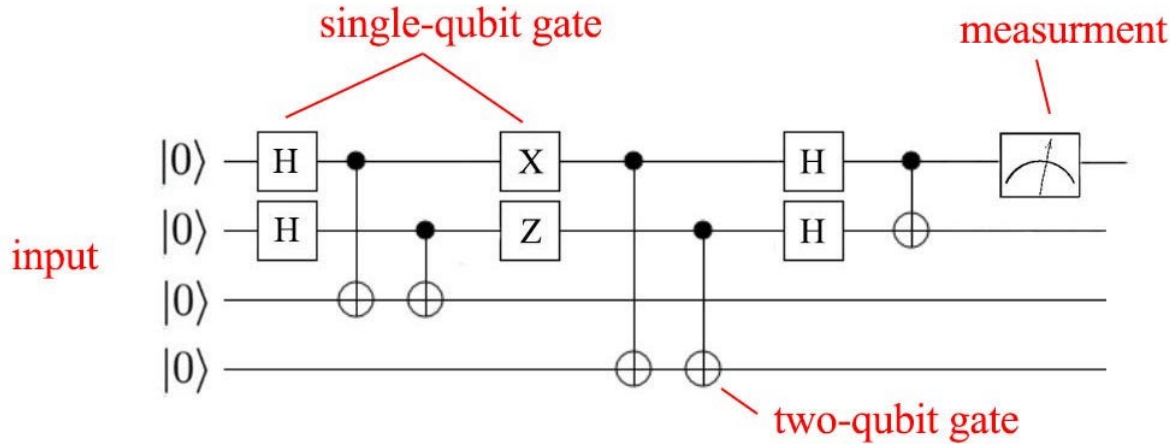


Image Src: <https://jonathan-hui.medium.com/qc-programming-with-quantum-gates-8996b667d256>

Hybrid Algorithms – A QAOA Example

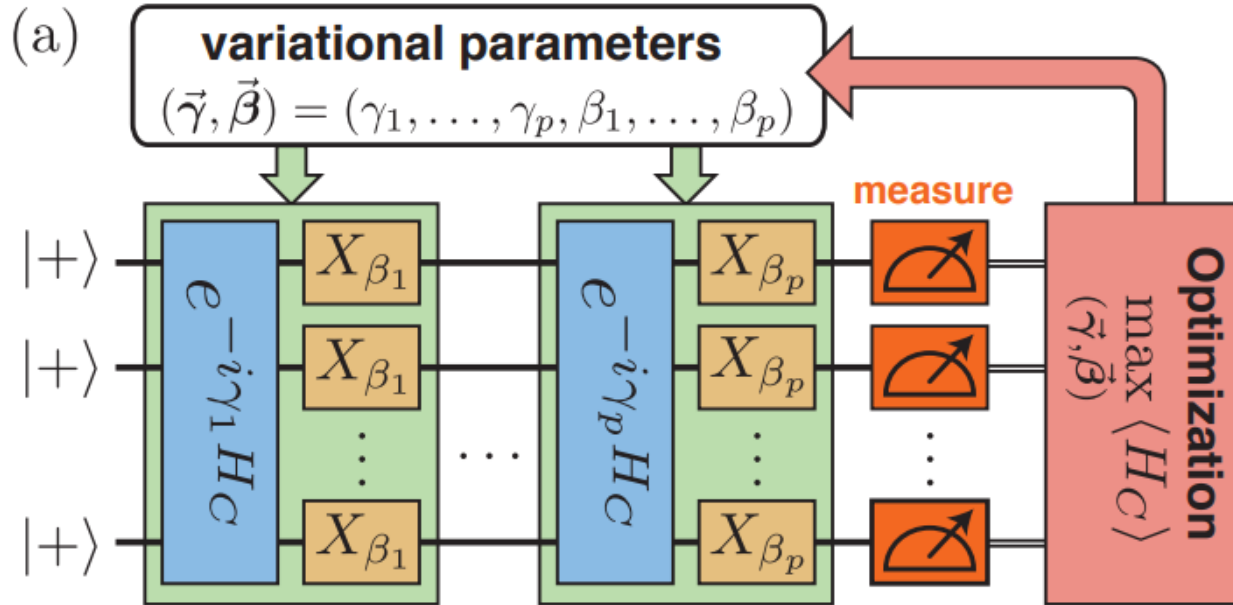
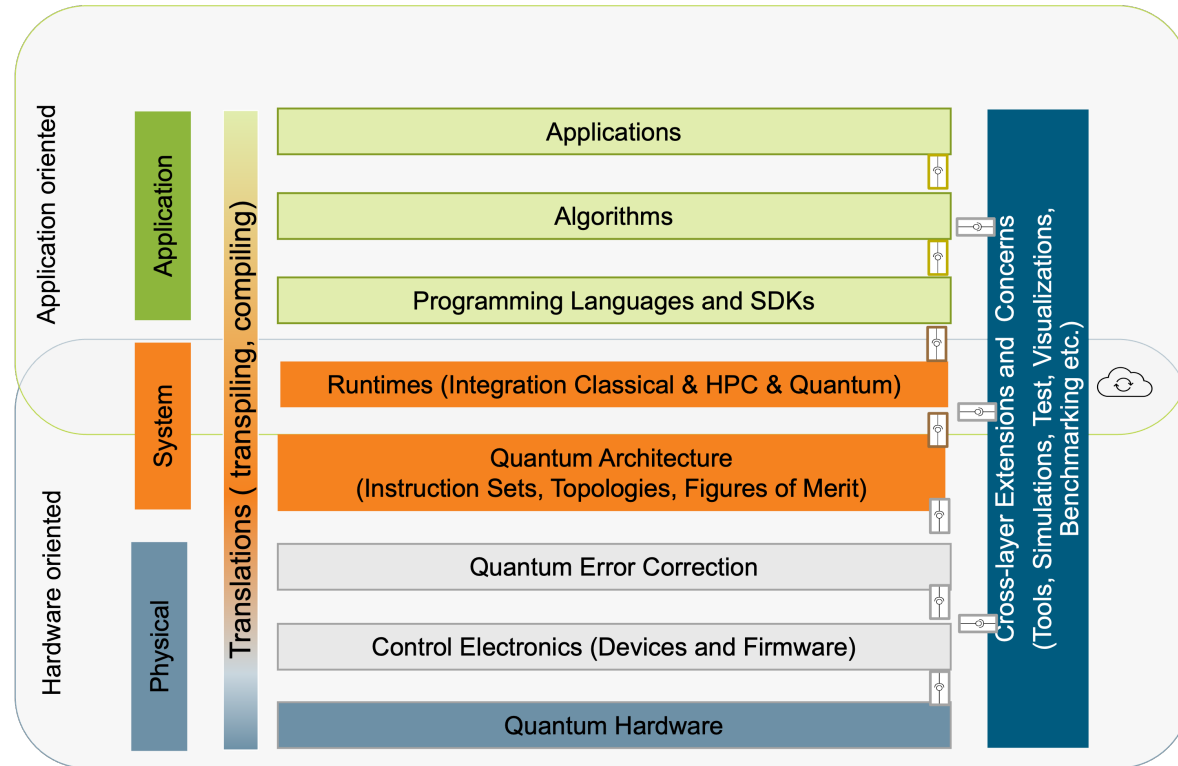


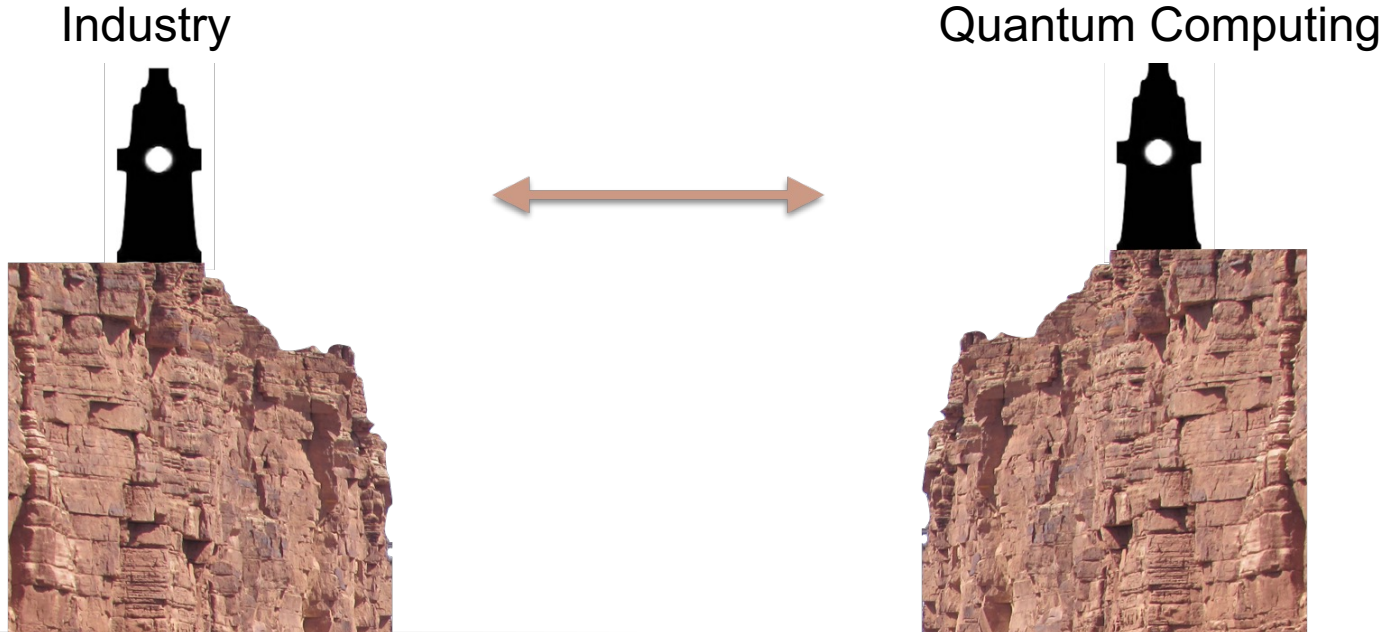
Image Src: <https://medium.com/@chs.li.work/qaoa-quantum-approximate-optimization-algorithm-1cf6dabdd581>

The „Quantum Software Stack“



Our Goal: „Enable the Enablers“

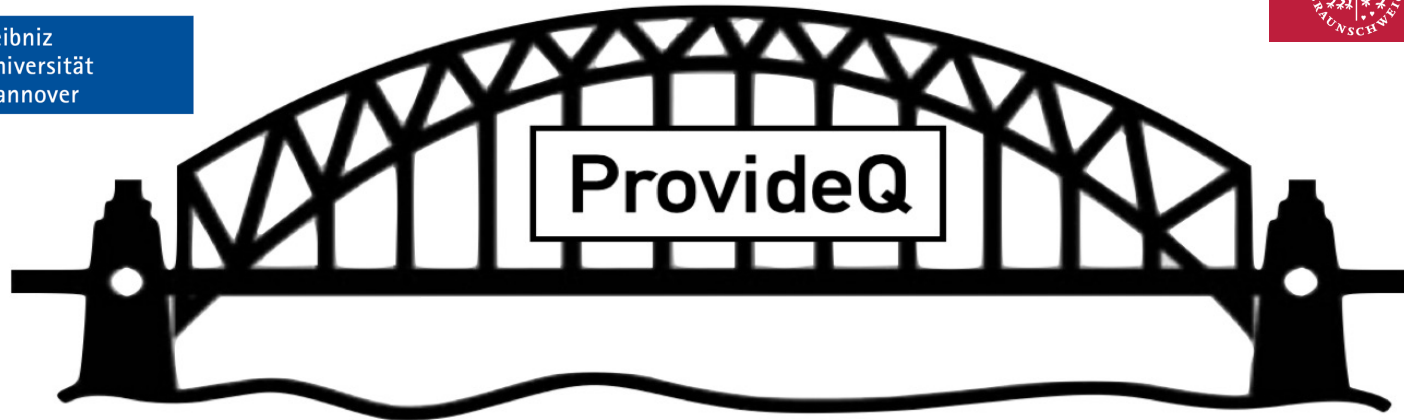
- Quantum Computing is very complicated...



Our Goal: „Enable the Enablers“



Leibniz
Universität
Hannover



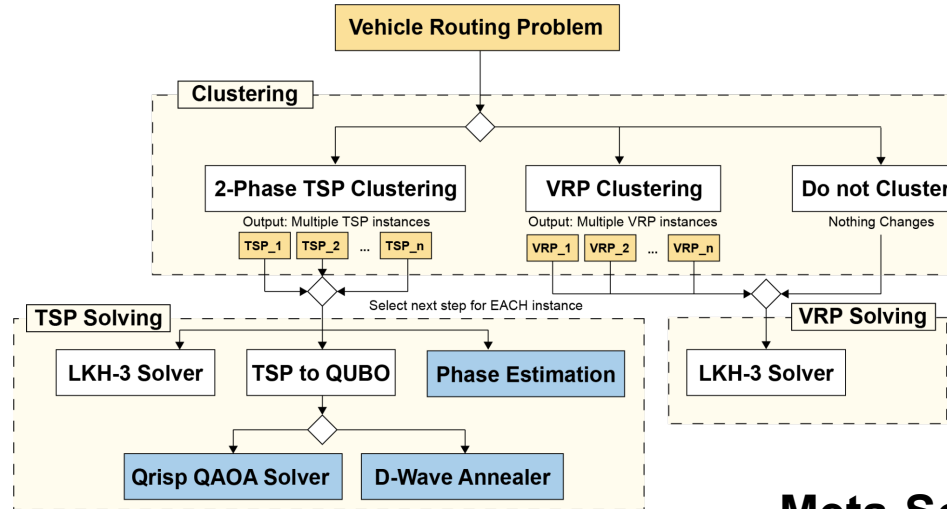
G A M S

4 F L O W

Classical Optimization Frameworks

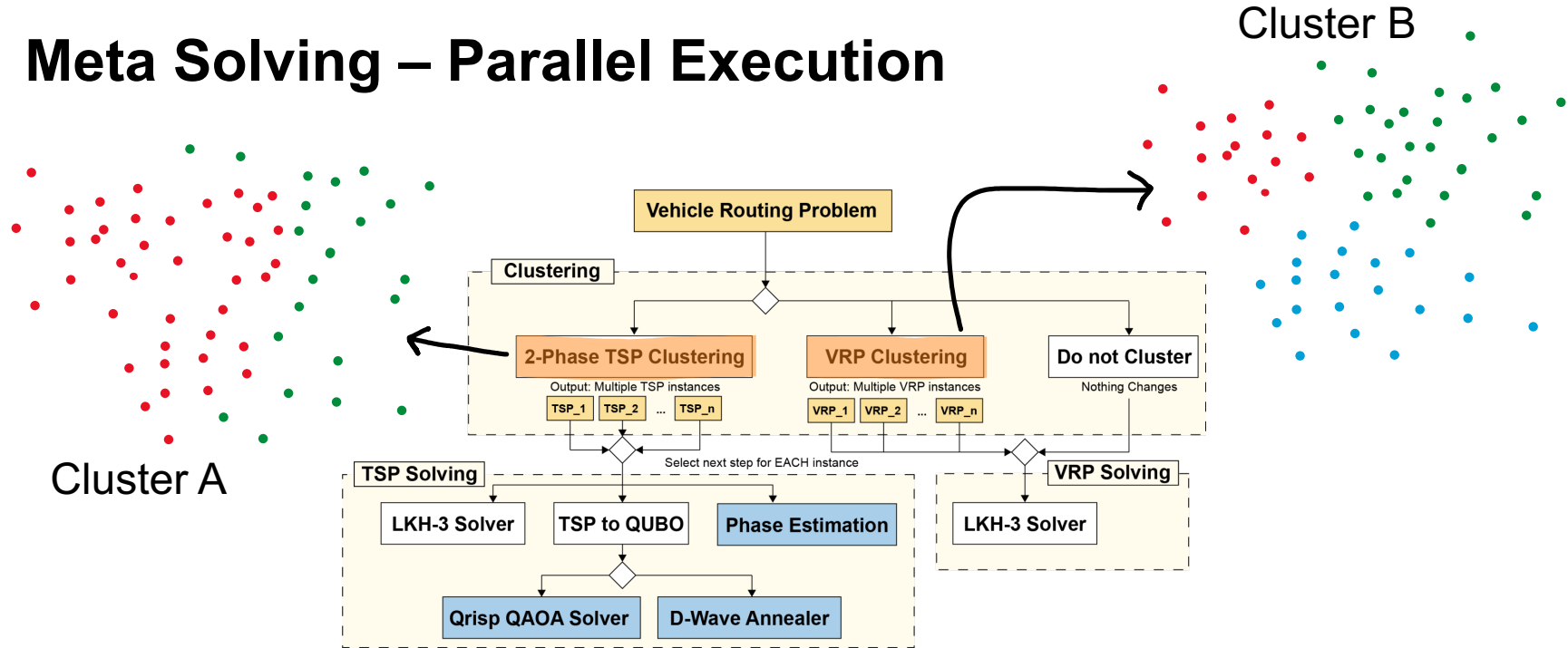


How to add Quantum? → Decompose Problems!



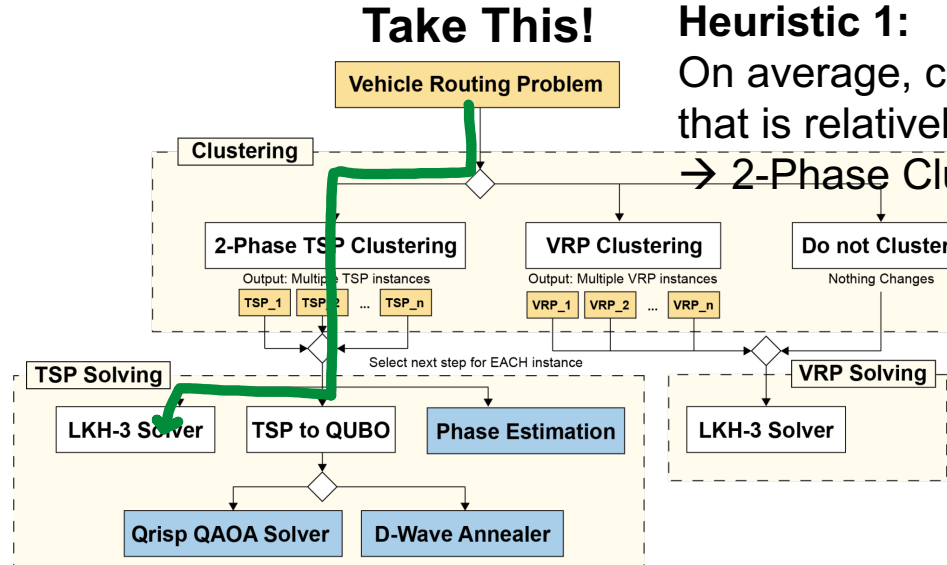
„Meta-Solving“

Meta Solving – Parallel Execution



If user is unsure about the next step,
they can try out multiple solution at once.

Meta Solving – Solution Path Suggestion



Heuristic 1:

On average, cities have low demand, that is relatively evenly split.

→ 2-Phase Clustering Approach is good.

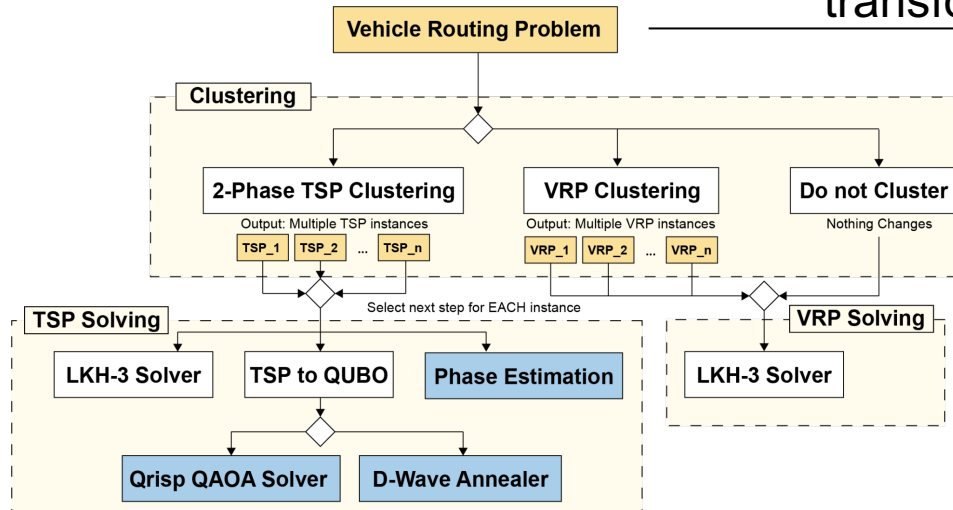
Heuristic 2:

Problem too large for a Quantum Computer

→ Use a classical solver.

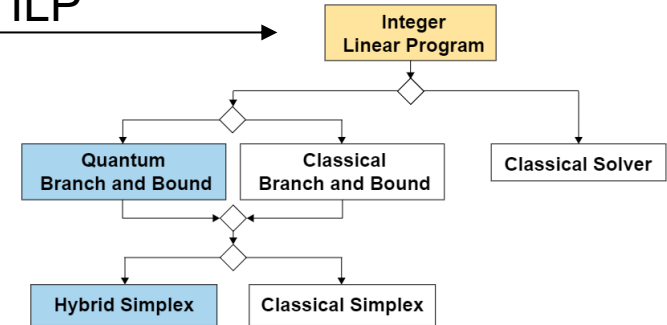
Meta Solving – Capsuling Strategies

Specialized Strategy

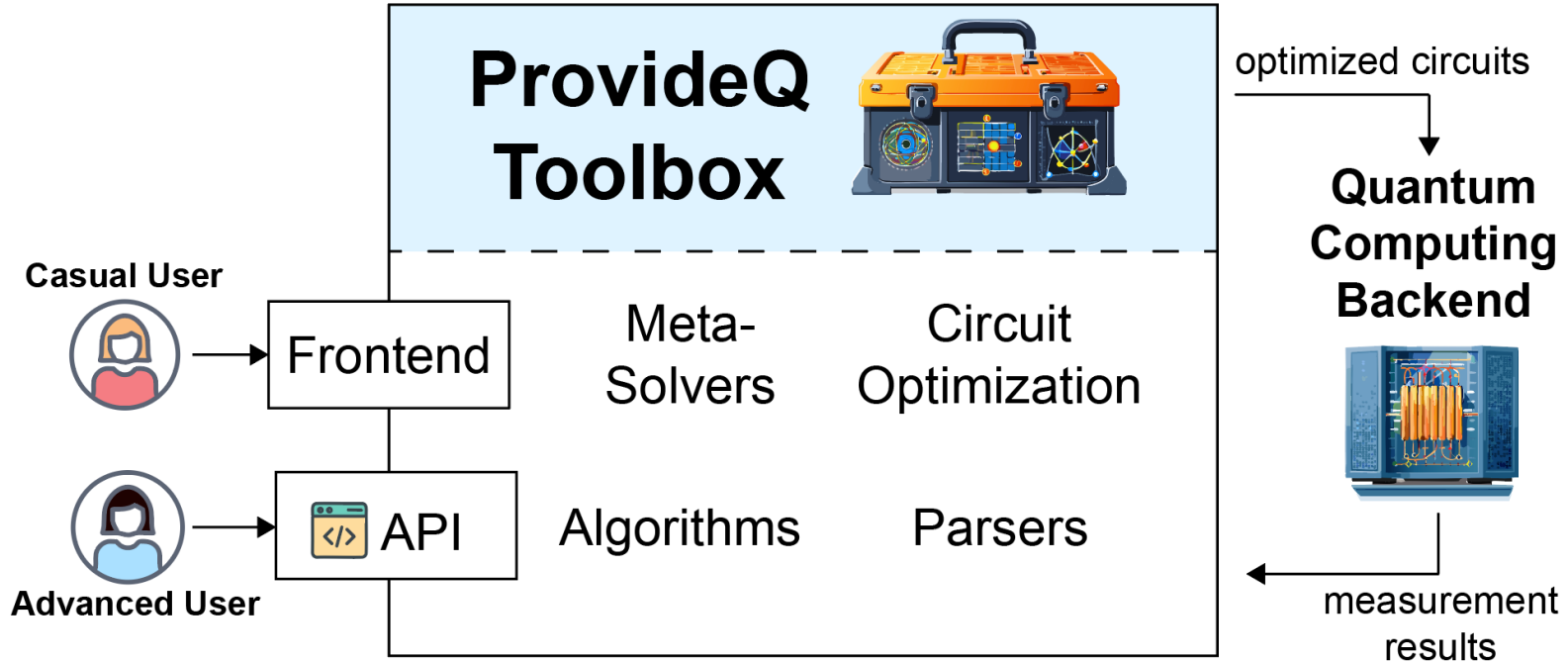


transform to ILP

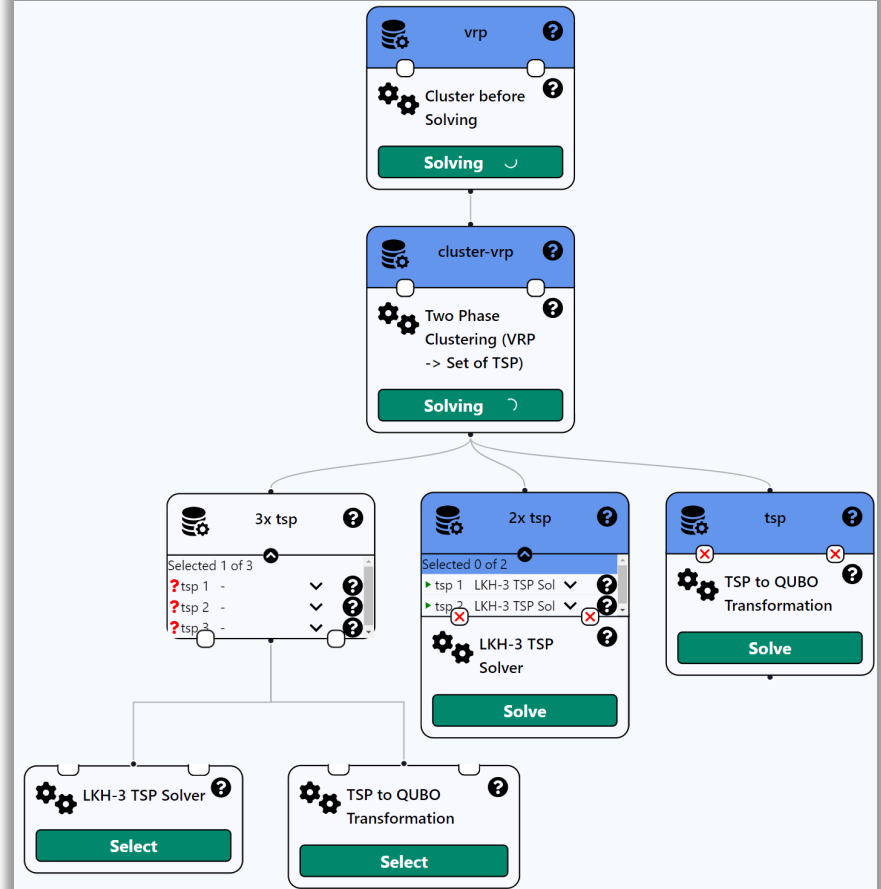
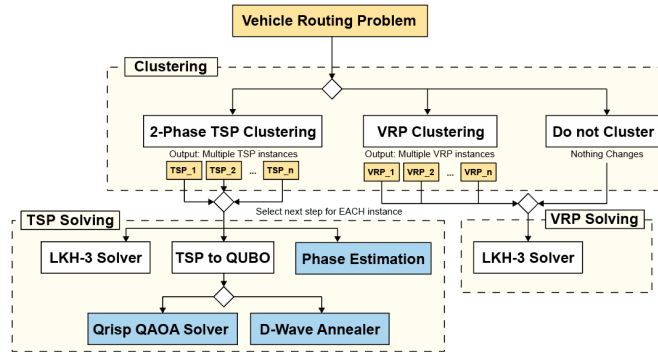
General Strategy



How to implement this?



The ProvideQ Toolbox



WIP: Quantum Circuit Execution & Optimization

Give a circuit in [OpenQASM](#) to process it. You can choose to execute the circuit, optimize it or apply error mitigation strategies.

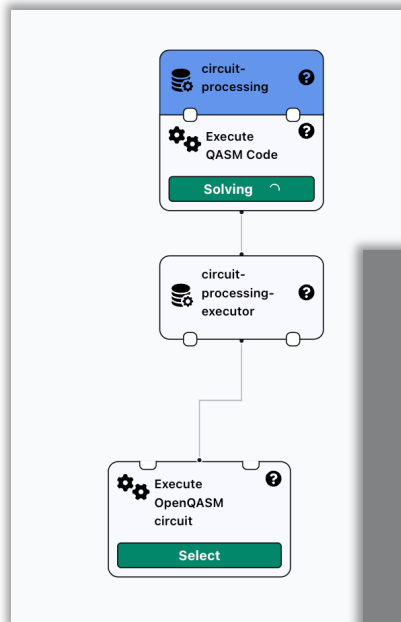
qreg q[1]; cre... Enter your OpenQASM code

```
qreg q[1]; creg c[1]; h q[0]; measure q[0] -> c[0];
```

circuit-processing

- Execute QASM Code** (Select)
- Optimize QASM Code** (Select)
- Mitigate QASM Code Errors** (Select)

React Flow



Give a circuit in [OpenQASM](#) to process it. You can choose to execute the circuit, optimize it or apply error mitigation strategies.

```
qreg q[1]; cre...  
qreg q[1]; creg c[1]; h q[0]; measure q[0] -> c[0];
```

circuit-processing-executor

Status: Ready to Solve

Solver: Execute OpenQASM circuit

Solver Settings:

- ☒ Number of shots
The number of shots to run
1024
- ☒ Selected Simulator
The simulator to run the code with
 - ✓ AerBackend
 - ForestStateBackend
 - ProjectQBackend
 - QulacsBackend

React Flow

ProvideQ API

knapsack

GET

/problems/knapsack

POST

/problems/knapsack

GET

/problems/knapsack/{problemId}

PATCH

/problems/knapsack/{problemId}

GET

/problems/knapsack/{problemId}/bound

GET

/problems/knapsack/{problemId}/bound/comp

GET

/problems/knapsack/examples

GET

/solvers/knapsack

GET

/solvers/knapsack/{solverId}/settings

GET

/solvers/knapsack/{solverId}/sub-routines

PATCH /problems/knapsack/{problemId}

Updates the problem of type 'knapsack' with the given problem ID. Only the 'input', 'solverId', 'solverSettings', and 'state' fields can be updated, all other fields will be ignored. Changes to the input or solver will reset the problem state to 'READY_TO_SOLVE'. If the problem is fully configured, changing the state to 'SOLVING' will start the solution process. Setting the state to another value is not allowed. The endpoint will respond with the updated problem.

Parameters

Name	Description
problemId * required string (path)	problemId

Request body * required

application/json

Examples:

knapsack

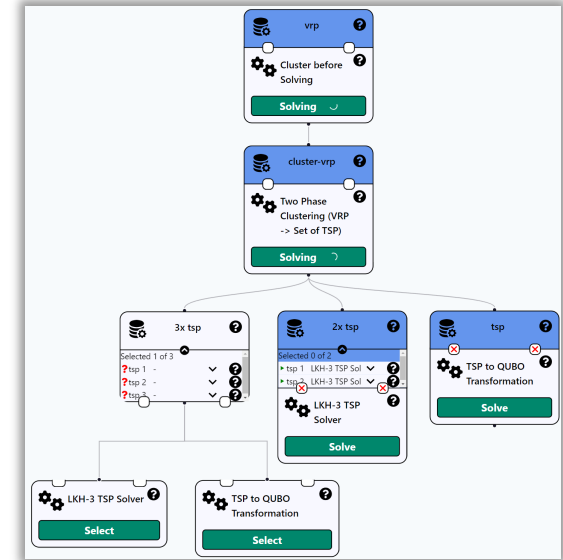
Example Value | Schema

```
{
  "id": "f4077a46-1a42-419e-80d3-1a058c3dc3c4",
  "typeId": "knapsack",
  "input": "4\n1 5 5\n2 3 4\n3 4 3\n4 3 2\n9",
  "solution": null,
  "bound": null,
  "state": "NEEDS_CONFIGURATION",
  "solverId": null,
  "solverSettings": [],
  "subProblems": []
}
```

Example Description

Conclusion

- Quantum Computing is not the answer to everything, but it has potential to provide supremacy for solving combinatorial optimization problems.
- Meta-Solving and the ProvideQ toolbox provide prototypical concepts and implementations that allow an easy adaption for hybrid optimization.
- We have a hardware bottleneck.



Find us at:

<https://github.com/ProvideQ>

<https://provideq.kit.edu/>

