

## ProvideQ: Short Overview of a Hybrid Optimization Toolbox

Domenik Eichhorn, Nick Poser, Maximilian Schweikart, Ina Schaefer



# 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](mailto: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

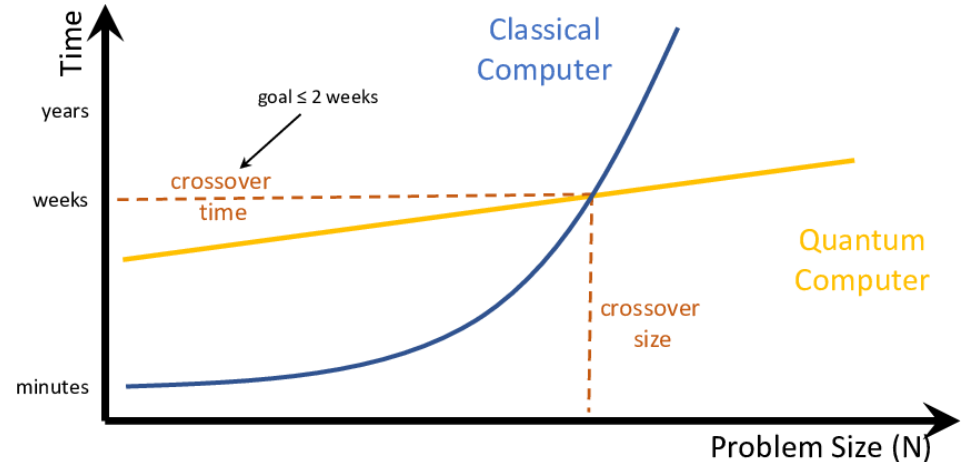


Image Src: T. Hoefler et al., *Disentangling Hype from Practicality: On Realistically Achieving Quantum Advantage*

# Motivation – Quantum Supremacy

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

## Use Cases:

- Cryptography
- Material Simulation
- **Combinatorial Optimization**

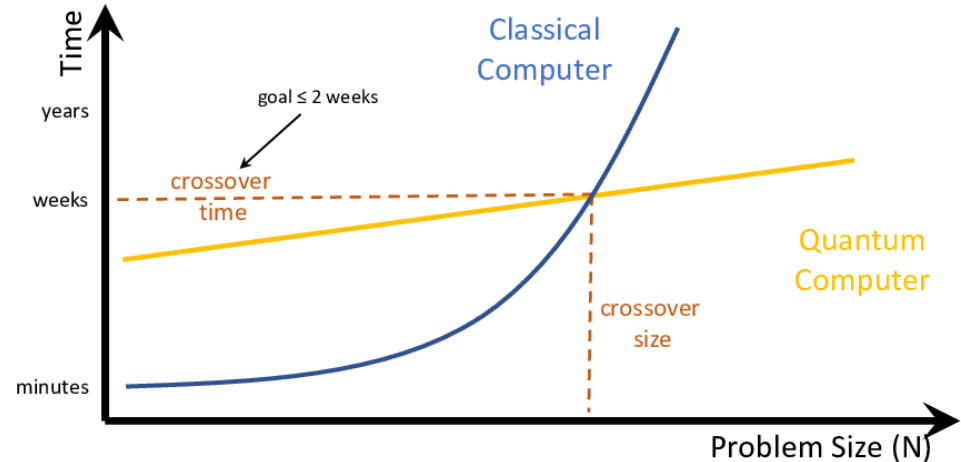
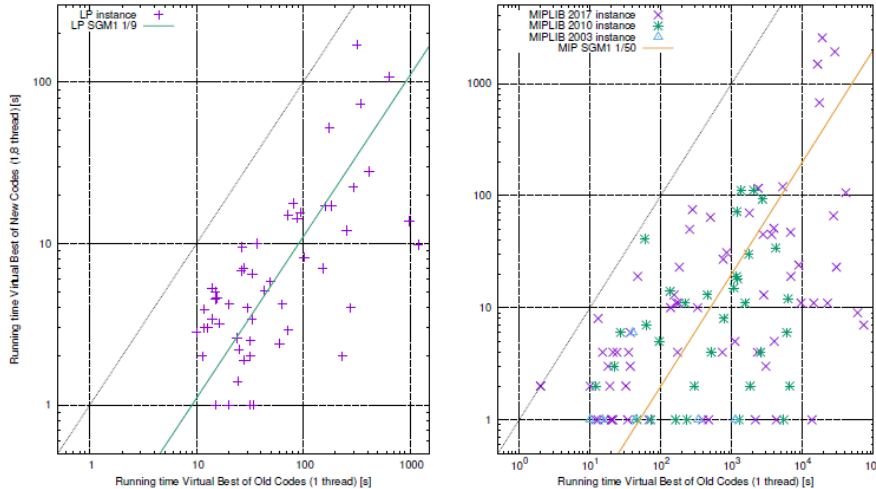


Image Src: T. Hoefler et al., *Disentangling Hype from Practicality: On Realistically Achieving Quantum Advantage*

# 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

# Classical Optimization Frameworks



# 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)
- Achieved with the „LKH-3“ Solver:  
<http://webhotel4.ruc.dk/~keld/research/LKH-3/>

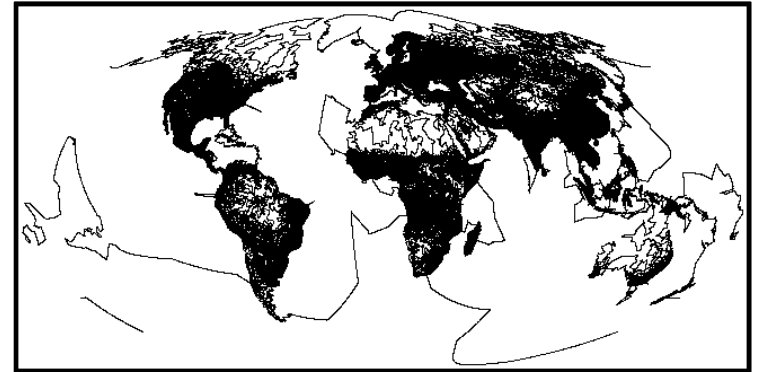
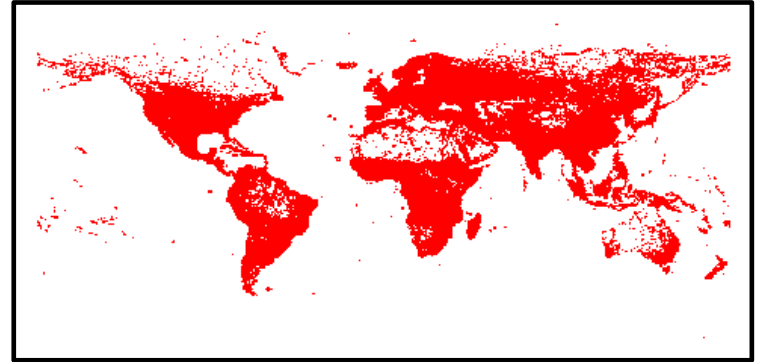


Image Source:  
<https://www.math.uwaterloo.ca/tsp/world/>

# 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

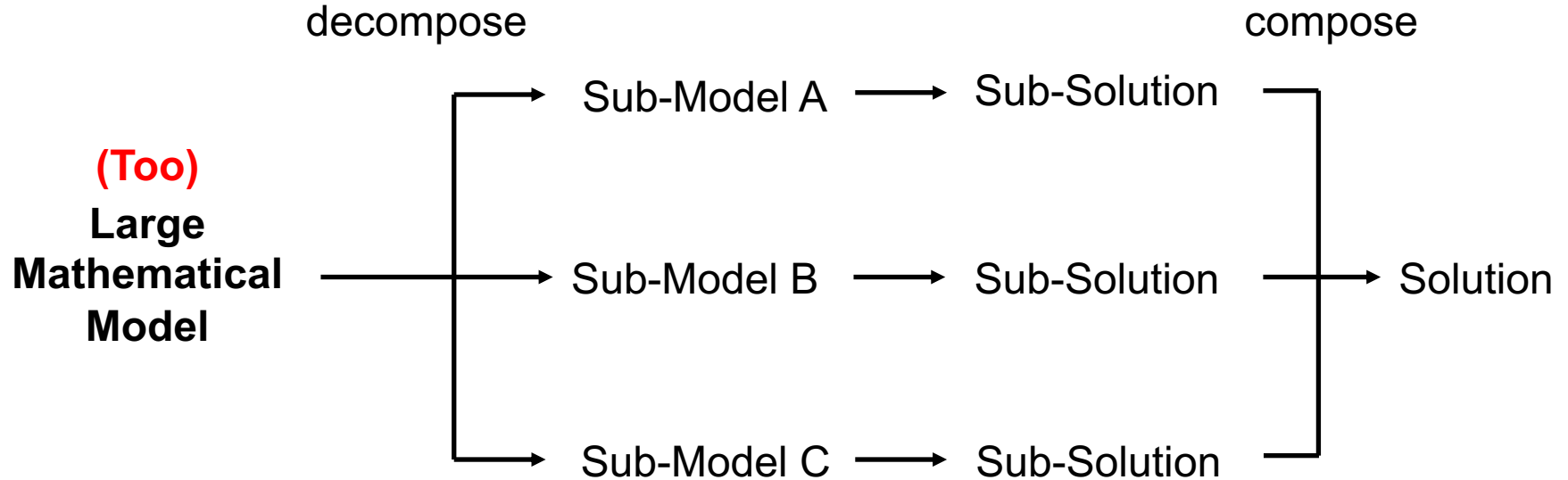
Ruslan Shaydulin<sup>1\*</sup>, Changhao Li<sup>1</sup>, Shouvanik Chakrabarti<sup>1</sup>, Matthew DeCross<sup>2</sup>, Dylan Herman<sup>1</sup>, Niraj Kumar<sup>1</sup>, Jeffrey Larson<sup>3</sup>, Danylo Lykov<sup>1,4</sup>, Pierre Minssen<sup>1</sup>, Yue Sun<sup>1</sup>, Yuri Alexeev<sup>4</sup>, Joan M. Dreiling<sup>2</sup>, John P. Gaebler<sup>2</sup>, Thomas M. Gatterman<sup>2</sup>, Justin A. Gerber<sup>2</sup>, Kevin Gilmore<sup>2</sup>, Dan Gresh<sup>2</sup>, Nathan Hewitt<sup>2</sup>, Chandler V. Horst<sup>2</sup>, Shaohan Hu<sup>1</sup>, Jacob Johansen<sup>2</sup>, Mitchell Matheny<sup>2</sup>, Tanner Mengle<sup>2</sup>, Michael Mills<sup>2</sup>, Steven A. Moses<sup>2</sup>, Brian Neyenhuis<sup>2</sup>, Peter Siegfried<sup>2</sup>, Romina Yalovetzky<sup>1</sup>, Marco Pistoia<sup>1</sup>

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.

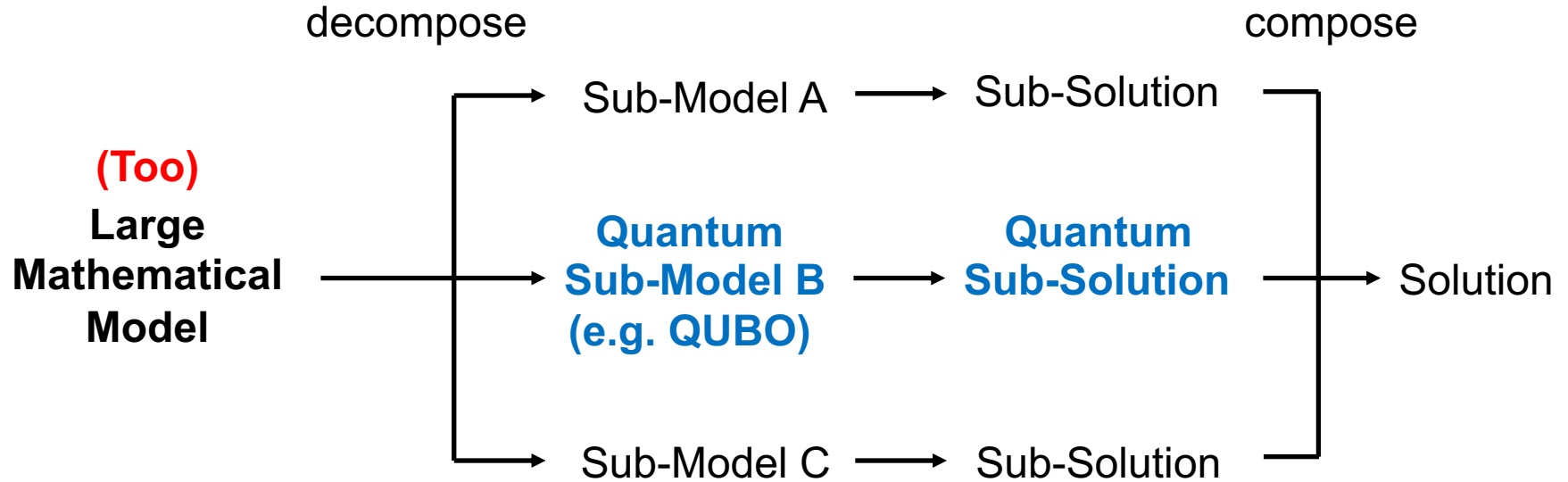
# Classical Optimization Frameworks



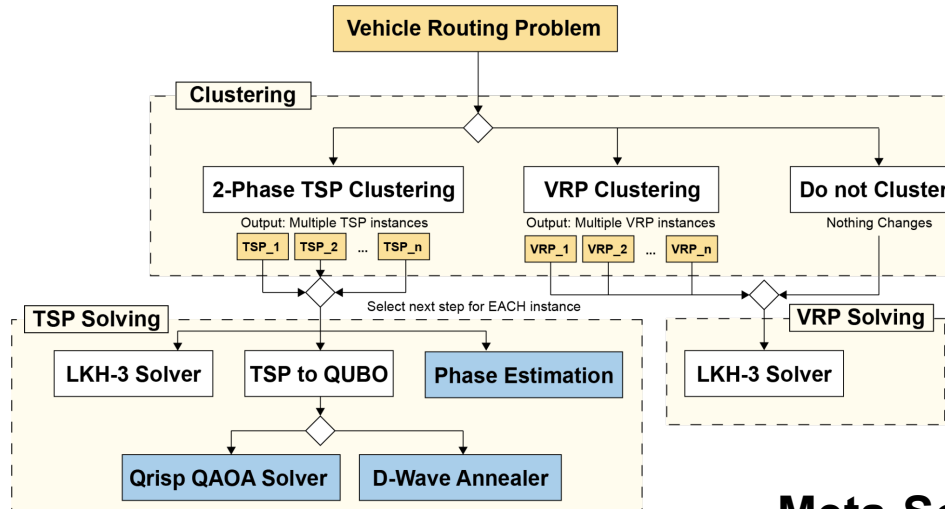
# Polyolithic Modeling



# Polyolithic Modeling

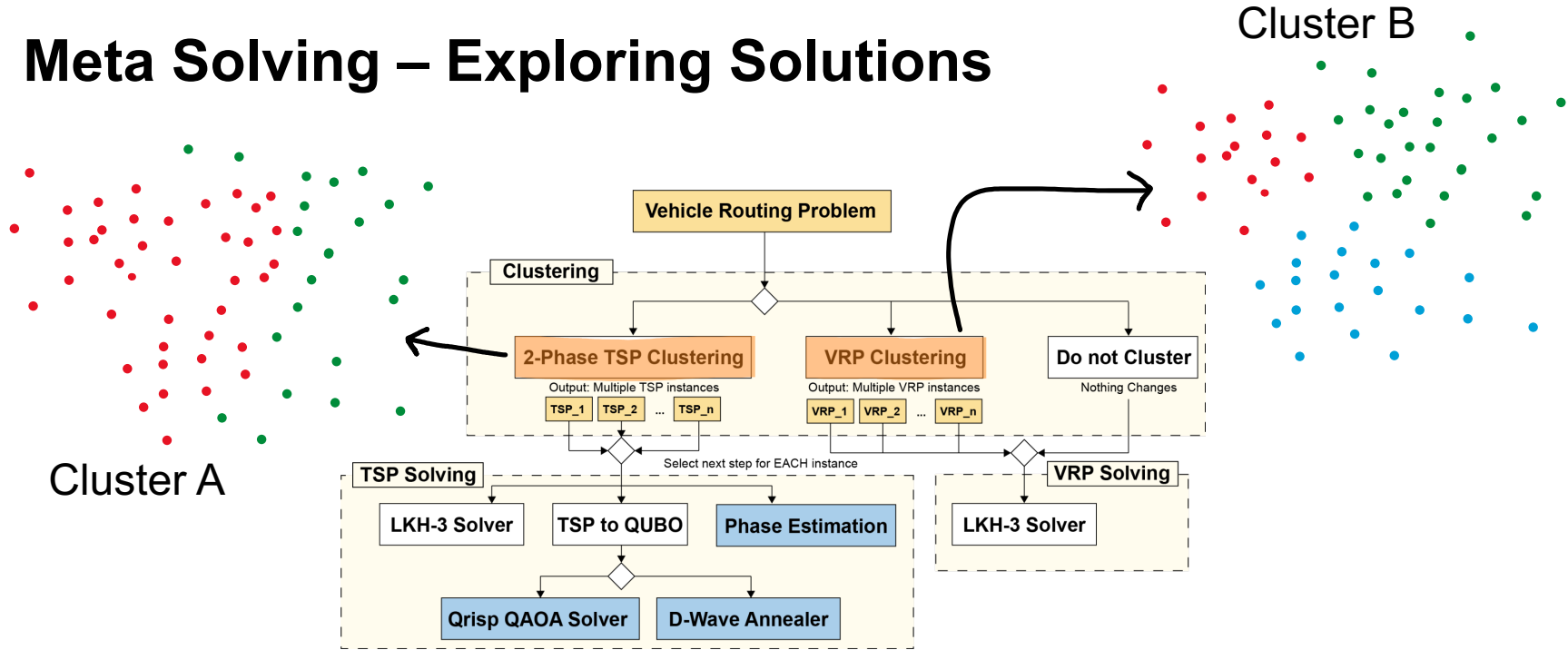


# How to add Quantum? → Decompose Problems!



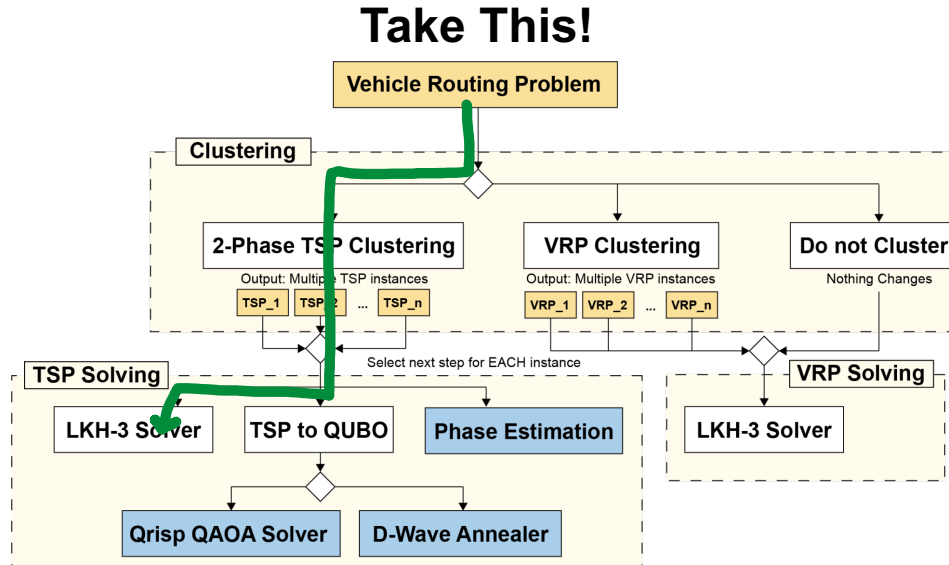
„Meta-Solving“

# Meta Solving – Exploring Solutions

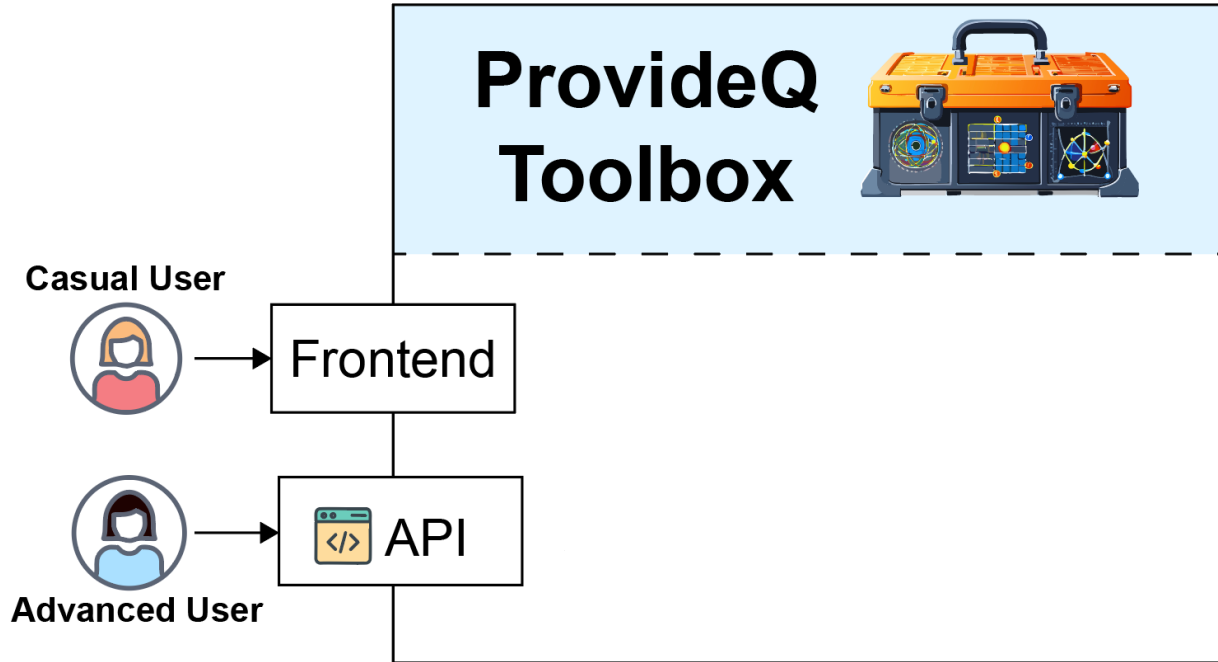


If user is unsure about the next step,  
they can try out multiple solution at once.

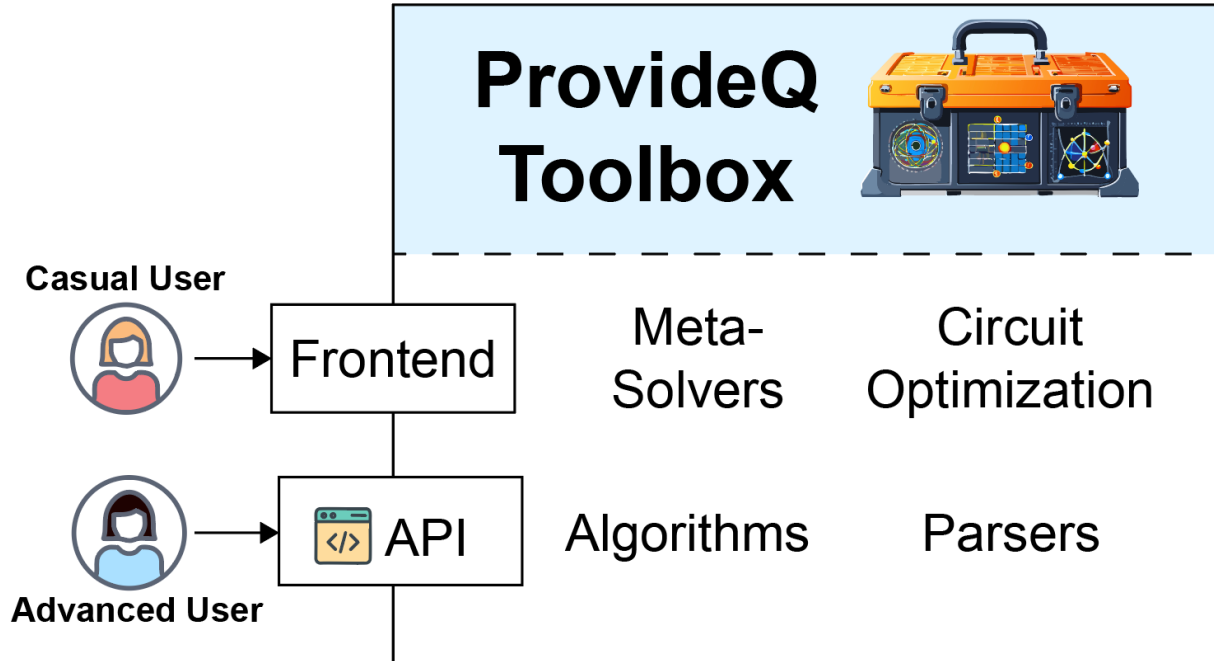
# Meta Solving – Proposing Optimal Solvers



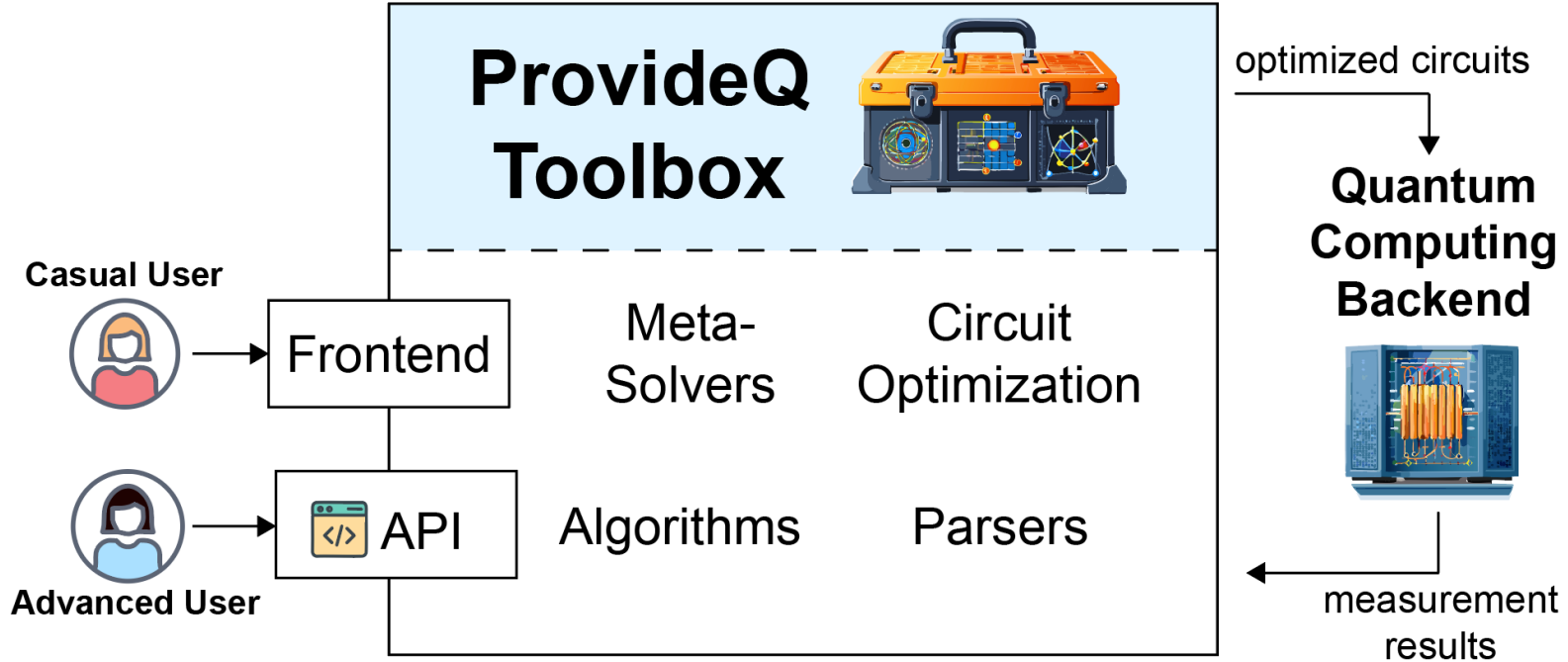
# How to implement this?



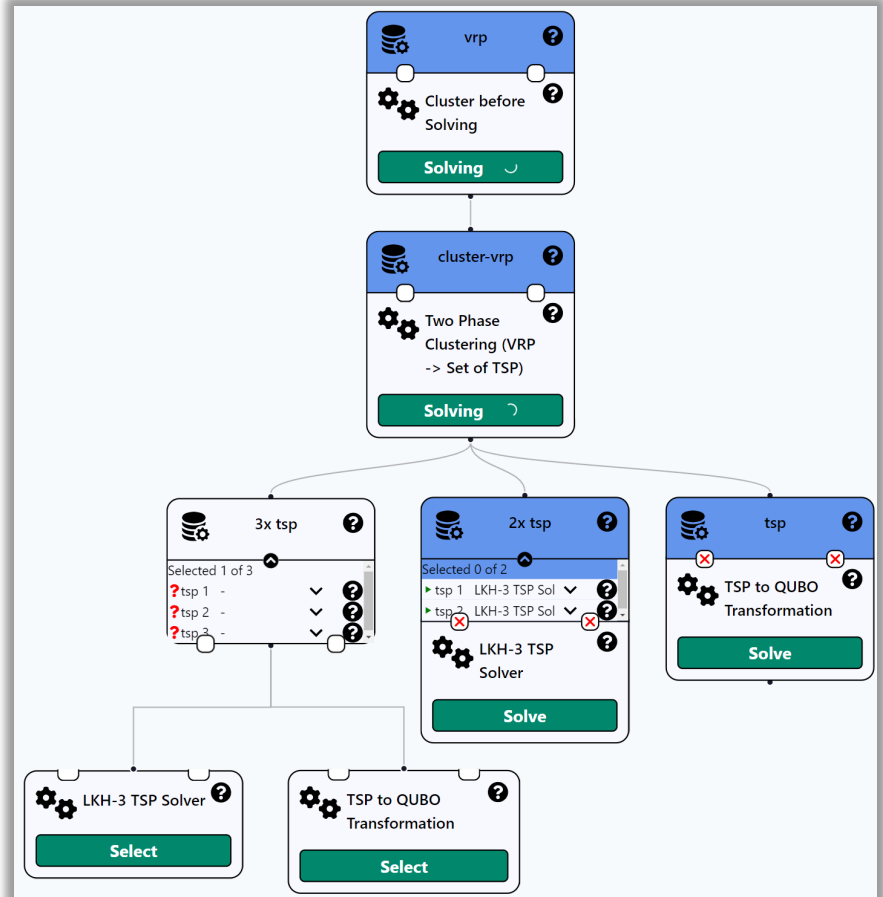
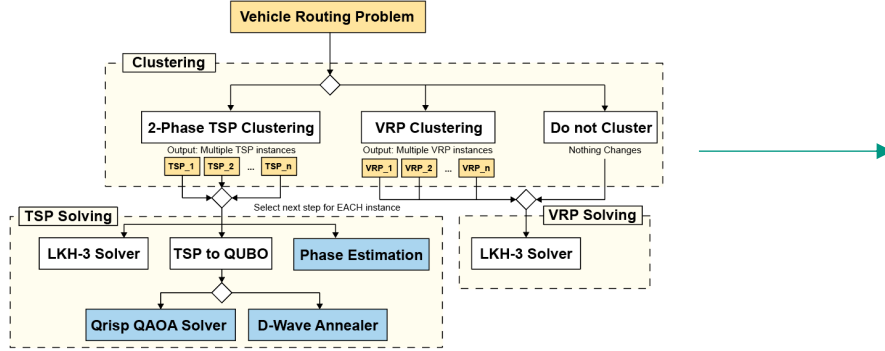
# How to implement this?

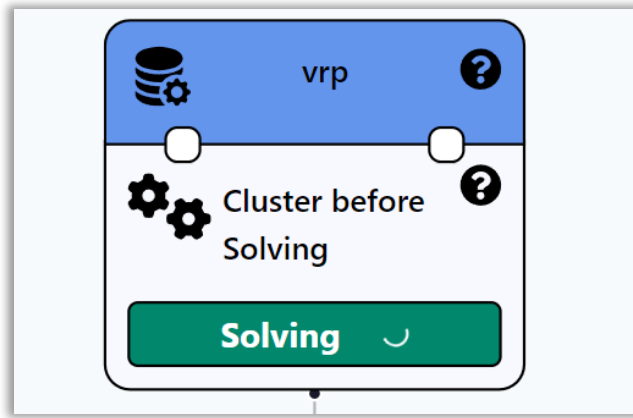


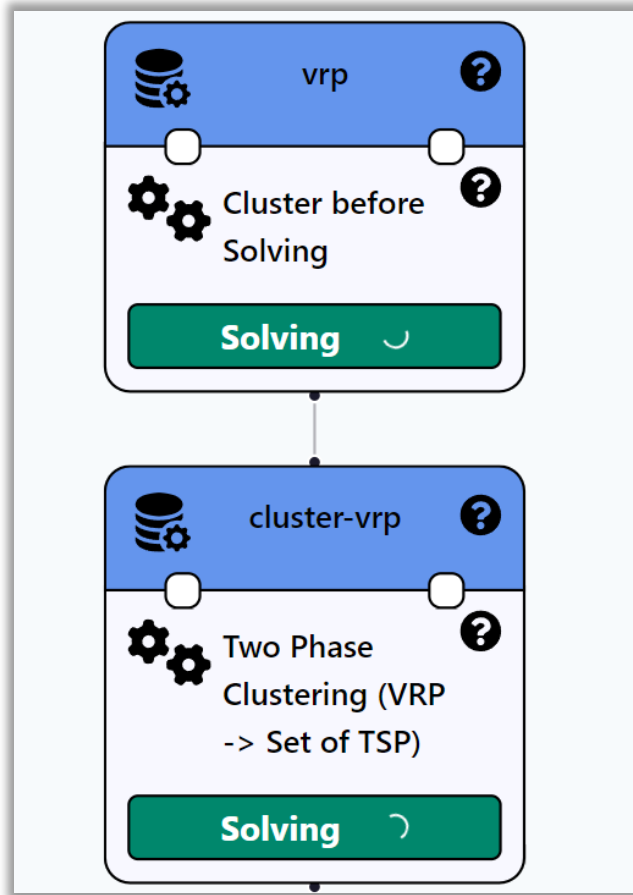
# How to implement this?

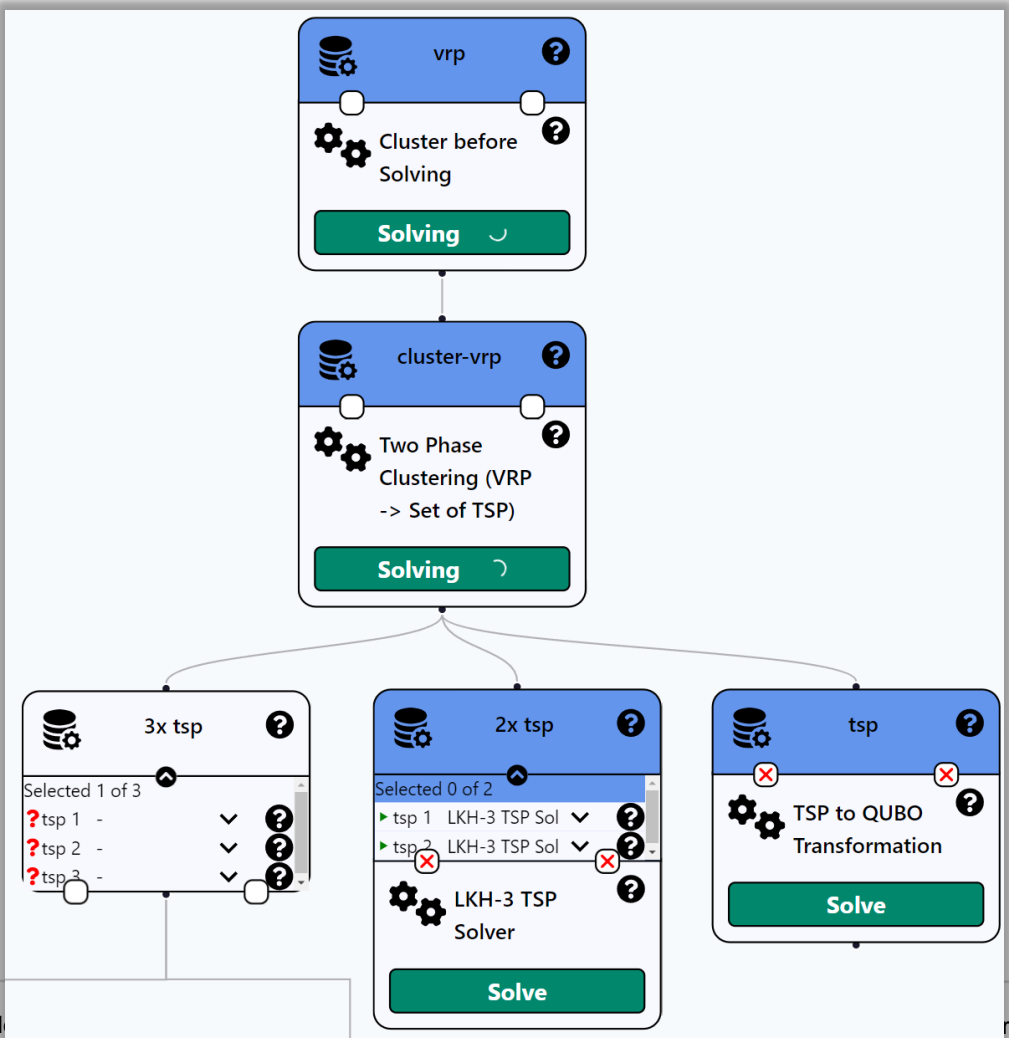


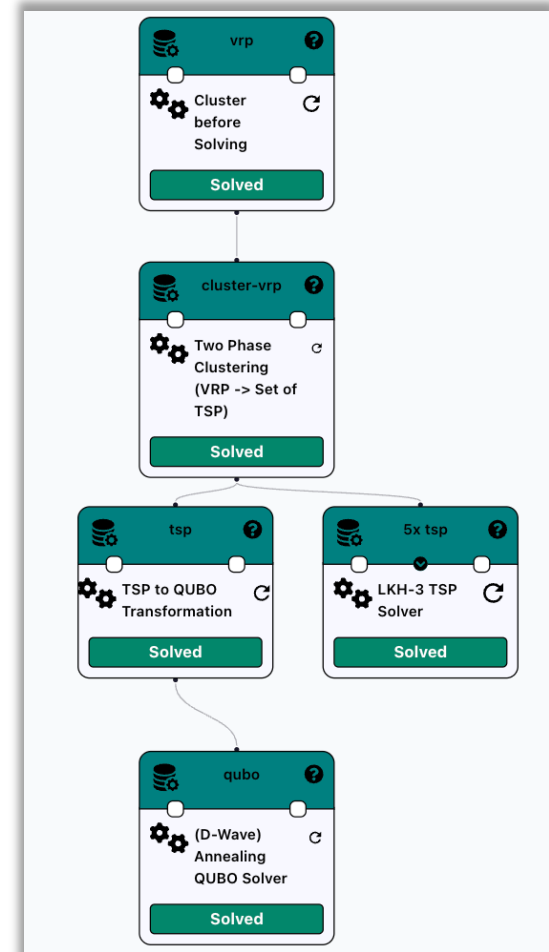
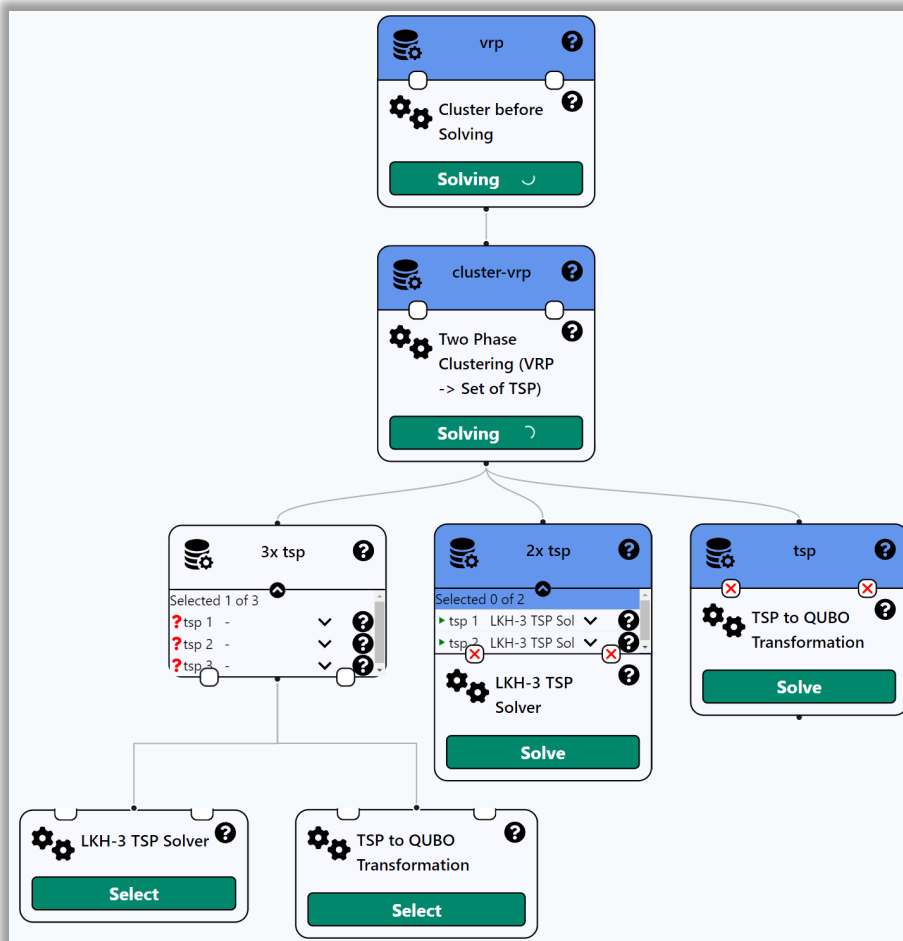
# The ProvideQ Toolbox













### Solution by Two Phase Clustering (VRP -> Set of TSP) ^

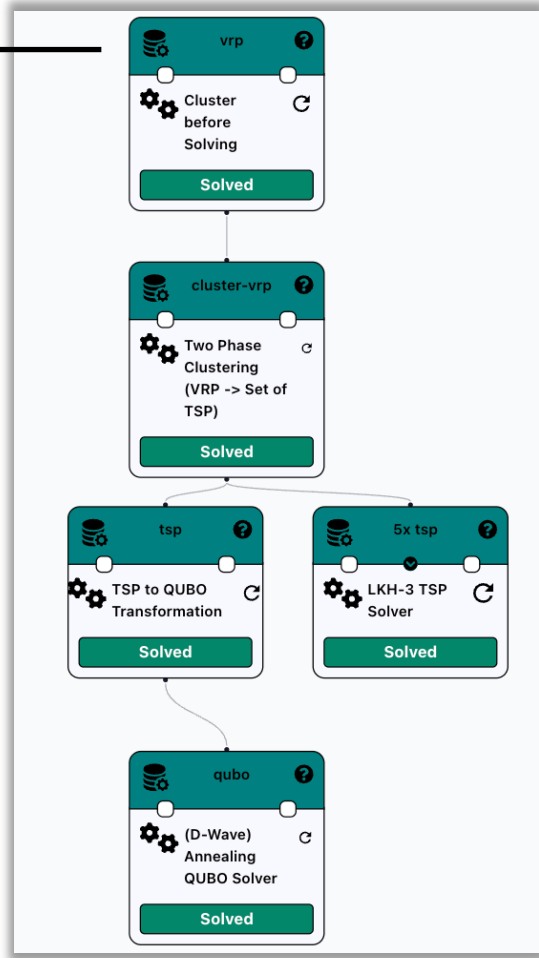
```
NAME : CMT1 solved with length 657.2501647960648
TYPE : TOUR
DIMENSION : 51
TOUR_SECTION
1 20 41 43 42 14 19 45 18 5 -1
1 7 15 26 25 44 8 24 49 28 -1
1 6 39 10 50 11 34 46 16 38 13 -1
1 33 2 23 3 30 22 51 17 12 47 48 -1
1 32 29 4 37 36 21 35 31 40 -1
1 9 27 -1
-1
EOF
```

### Meta Data ^

Problem ID: 23cbcb8d-e25f-4ca3-8282-a3d1a22097f6

Solver: Two Phase Clustering (VRP -> Set of TSP)

Execution time: 99.231s



```

tsp
NAME: CMT1_0
TYPE: CVRP
COMMENT: 524.61 - Cluster Nr.0
DIMENSION: 10
CAPACITY: 160
EDGE_WEIGHT_TYPE: EUC_2D
NODE_COORD_TYPE: TWOD_COORDS
NODE_COORD_SECTION:
9 21 10
4 27 23
8 10 17
5 17 33
2 20 26
10 30 15
7 5 6
3 5 25
1 30 40
6 13 13
DEPOT_SECTION:
1
-1
DEMAND_SECTION:
2 9
7 7
6 9
10 16

```

Solution by TSP to QUBO Transformation

```

NAME : CMT1_0 solved with length 151.491
TYPE : TOUR
DIMENSION : 10
TOUR_SECTION
1 6 7 9 8 3 5 10 4 2 -1
-1
EOF

```

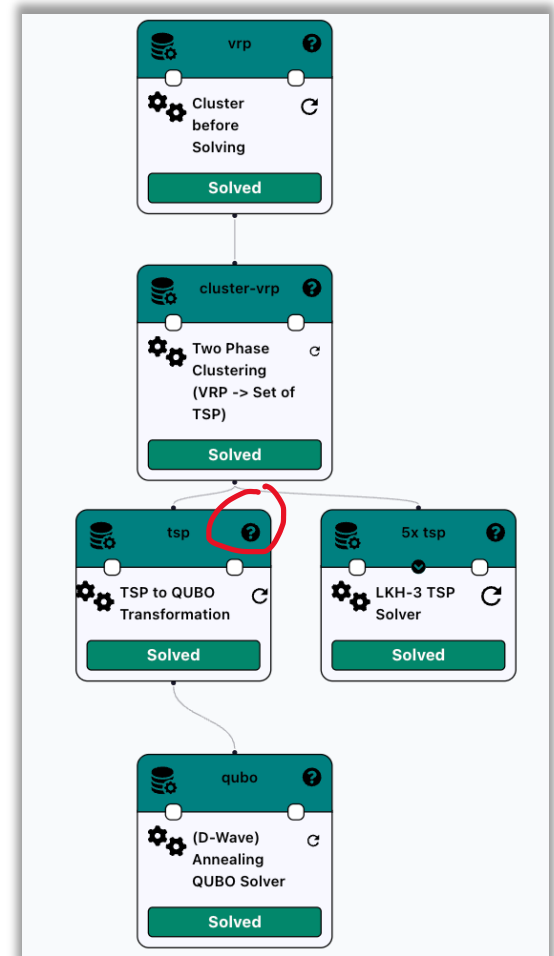
Meta Data

Problem ID: 11fbd44b-7ae2-4c32-8188-40fe

Solver: TSP to QUBO Transformation

Execution time: 73.679s

Additional meta data:



qubo

\ QUBO from TSP

Minimize

obj: 0 + [ -1688.0758276807353 x1 \* x1 +  
1688.0758276807353 x1 \* x2 +  
1688.0758276807353 x1 \* x3 +  
1688.0758276807353 x1 \* x4 +  
1688.0758276807353 x1 \* x5 +  
1688.0758276807353 x1 \* x6 +  
1688.0758276807353 x1 \* x7 +  
1688.0758276807353 x1 \* x8 +

0  
0  
0  
0  
1  
0  
0

Solver Settings:

D-Wave Token

The D-Wave token to use, needed to access the D-Wave hardware

Annealing Method

The annealing method to use, only relevant when a token is added

sim

Save

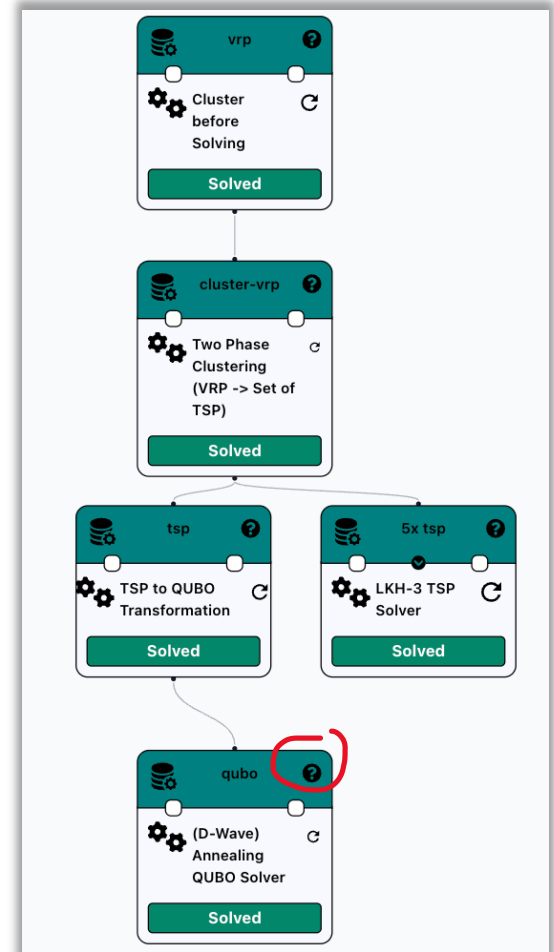
Meta Data

Problem ID: 9070c437-534e-4e25-8347-00f1

Solver: (D-Wave) Annealing QUBO Solver

Execution time: 24.239s

Additional meta data:



# Whats coming: 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];
```

React Flow

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]; cre...
```

**circuit-processing-executor**

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

**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

# Try out the API:

## 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
<b>problemId</b> * required	
string (path)	<input type="text" value="problemId"/>

**Request body** \* required application/json

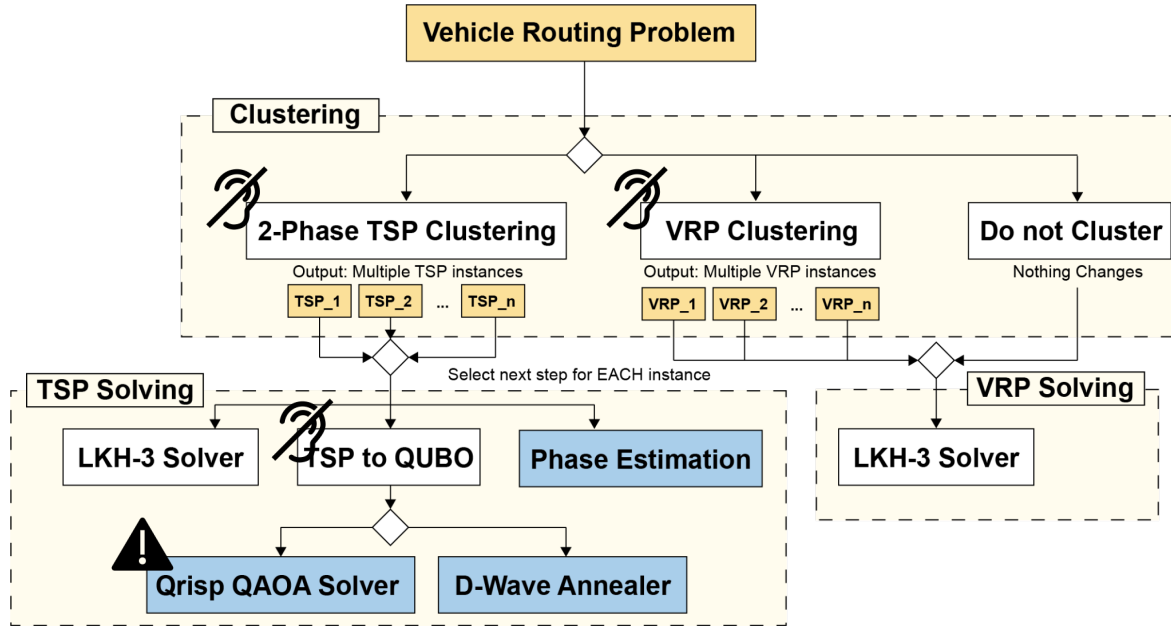
**Examples:**

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

# FW: Formalizing Meta-Solver Strategies



## Example Tags:



#invalid ==  
algorithm might return  
an invalid result



#optimal ==  
algorithms always finds  
an optimal solution



#unsound ==  
transformation could  
alter the solution space

### Rule 1 -- Invalid Paths:

A Path is invalid as soon as one subroutine is invalid.

### Rule 2 -- Optimal Paths:

A Path is optimal when all subroutines are optimal.

# FW: Meta-Solver Language

```
1 solve VRP vrp:
2   if vrp.dimension > 10:
3     vrp.ClusterAndSolveVrpSolver():
4       solve ClusterVRP clustervrp:
5         clustervrp.TwoPhaseClusterer():
6           solve TSP[] tsps:
7             foreach tsp in tsps:
8               tsp.QuboTspSolver():
9                 solve QUBO qubo:
10                  qubo.DwaveQuboSolver(
11                    "D-Wave Token" = "token",
12                    "Annealing Method" = "sim")
13   else:
14     vrp.LkhVrpSolver()
```

# FW: Meta-Solver Language

The screenshot shows the Langium Meta Solver Strategy Editor. The interface is divided into a left sidebar and a main editor area. The sidebar contains a 'Problem Type' dropdown menu with 'VRP' selected, a description of the problem, a 'Strategies' section with a 'New' button and two strategy options ('Tsp-Clusters or Lkh' and 'Lkh or Qrisp for small problems'), and a 'Note' section with instructions. The main editor area has a 'Save' button and a text input field containing 'Tsp-Clusters or Lkh'. Below this, a code editor shows two lines of code: '1 solve VRP vrp;' and '2' followed by a blank line.

**Langium** Meta Solver Strategy Editor

**Problem Type**

2 VRP

A Capacitated Vehicle Routing Problem Optimization Problem with the goal to find a minimal route for a given set of trucks and cities with demand.

**Strategies** New

Tsp-Clusters or Lkh  
VRP

Lkh or Qrisp for small problems  
VRP

**Note**  
Select a problem type, then choose an existing strategy or create a new one to edit and save.

**Hint**  
Press Ctrl+Space inside the editor to trigger code completion.

Save Tsp-Clusters or Lkh

```
1 solve VRP vrp;  
2
```

# FW: Meta-Solver Language

**Langium Meta Solver Strategy Editor**

**Problem Type**

2 VRP

A Capacitated Vehicle Routing Problem Optimization Problem with the goal to find a minimal route for a given set of trucks and cities with demand.

**Strategies** New

- Tsp-Clusters or Lkh  
VRP
- Lkh or Qrisp for small problems  
VRP

**Note**  
Select a problem type, then choose an existing strategy or create a new one to edit and save.

**Hint**  
Press Ctrl+Space inside the editor to trigger code completion.

```
1 solve VRP vrp:  
2   if vrp.dimension > 10:  
3     vrp.ClusterAndSolveVrpSol  
4     solve ClusterVRP cluste  
5     clustervrp.TwoPhaseCl  
6     solve TSP[] tsps:  
7       foreach tsp in ts  
8         tsp.QuboTspSolv  
9         solve QUBO qu  
10        qubo.DwaveQ  
11        "D-Wave T  
12        "Annealin  
13 else:  
14   vrp.LkhVrpSolver()
```

**Save**

Tsp-Clusters or Lkh

```
NAME : test  
TYPE : CVRP  
DIMENSION : 3  
EDGE_WEIGHT_TYPE : EUC_2D  
CAPACITY : 2  
NODE_COORD_SECTION  
1 0.00000 0.00000  
2 1.00000 0.50000  
3 0.50000 -1.00000  
DEMAND_SECTION  
1 0  
2 1  
3 2  
DEPOT_SECTION  
1  
-1
```

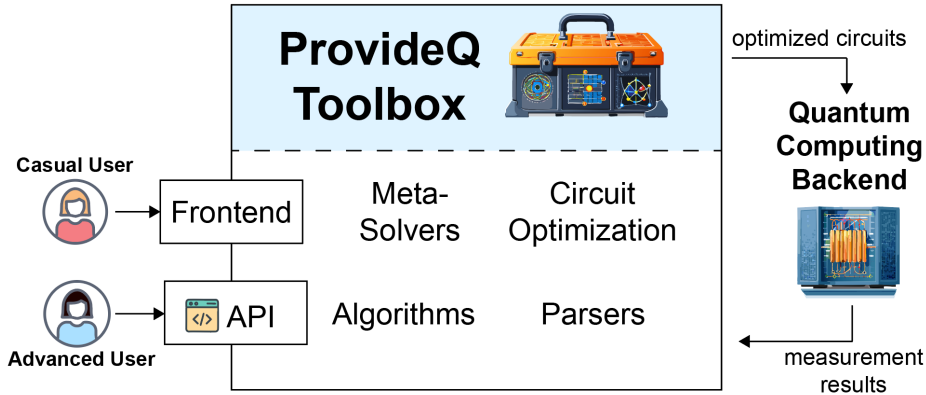
**VRP**

- Grover-based VRP Solver (Qrisp) Select Solver
- Cluster before Solving Select Solver
- LKH-3 VRP Solver Select Solver
- Tsp-Clusters or Lkh Select Strategy
- Lkh or Qrisp for small problems Select Strategy

React Flow

Karlsruher Institut für Technologie

# Conclusion



**Find us at:**  
<https://github.com/ProvideQ>  
<https://provideq.kit.edu/>

