PURDUE UNIVERSITY

# QUBO.jl

## A tale of implementation and benchmarking of a Quantum Optimization Ecosystem in Julia

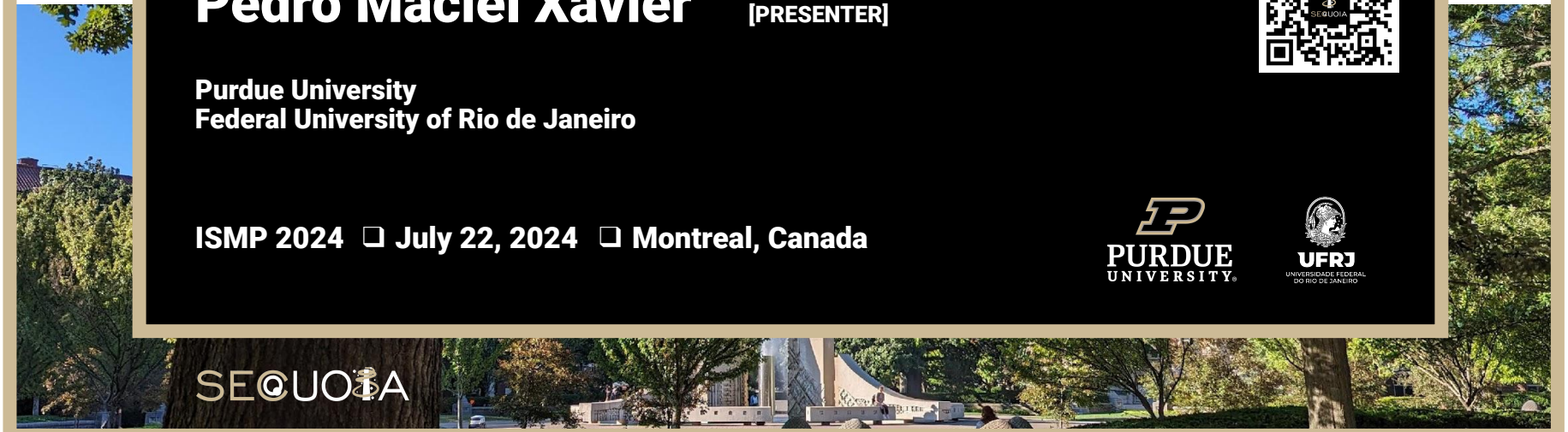

**Pedro Maciel Xavier** [PRESENTER]

Purdue University
Federal University of Rio de Janeiro


ISMP 2024 ❑ July 22, 2024 ❑ Montreal, Canada

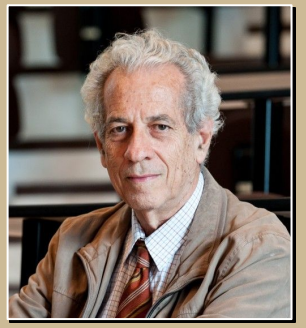
PURDUE UNIVERSITY

UFRJ
UNIVERSIDADE FEDERAL DO RIO DE JANEIRO



**Pedro Ripper**
*PUC-Rio, PSR*

**Joaquim Dias Garcia**
*PSR*



**Nelson Maculan**
*UFRJ*

**David E. Bernal Neira**
*Purdue University*
*USRA*
*NASA Quantum AI Lab*

PSR

UFRJ
UNIVERSIDADE FEDERAL
DO RIO DE JANEIRO

NASA

PURDUE
UNIVERSITY

USRA
Universities Space Research Association

# QUBO: Quadratic Unconstrained Binary Optimization

$$\min \ \mathbf{x}'\mathbf{Q}\,\mathbf{x} + \boldsymbol{\ell}'\mathbf{x} + c$$

OBJECTIVE FUNCTION

$$\text{s.t.} \ \mathbf{x} \in \{0,1\}^n$$

BINARY VARIABLES

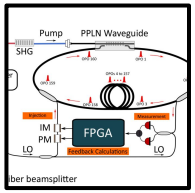
# QUBO: Quadratic Unconstrained Binary Optimization

$$\min \ \mathbf{x}'\mathbf{Q}\,\mathbf{x} + \boldsymbol{\ell}'\mathbf{x} + c$$

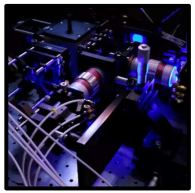
OBJECTIVE FUNCTION

$$\text{s.t.} \ \ \mathbf{x} \in \{0,1\}^n$$

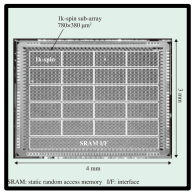
BINARY VARIABLES




P. L. McMahon et al., 2016

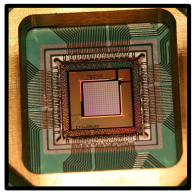


Microsoft Research

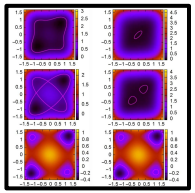



Hitachi, Yamaoka et al.

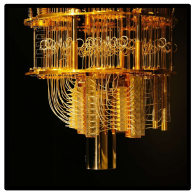
**Quantum Annealing, Digital Annealing, Variational Quantum Eigensolver, Quantum Alternating Optimization Ansatz, Coherent Ising Machine, Analog Iterative Machine, Simulated Bifurcation Machine, CMOS Annealing…**



D-Wave



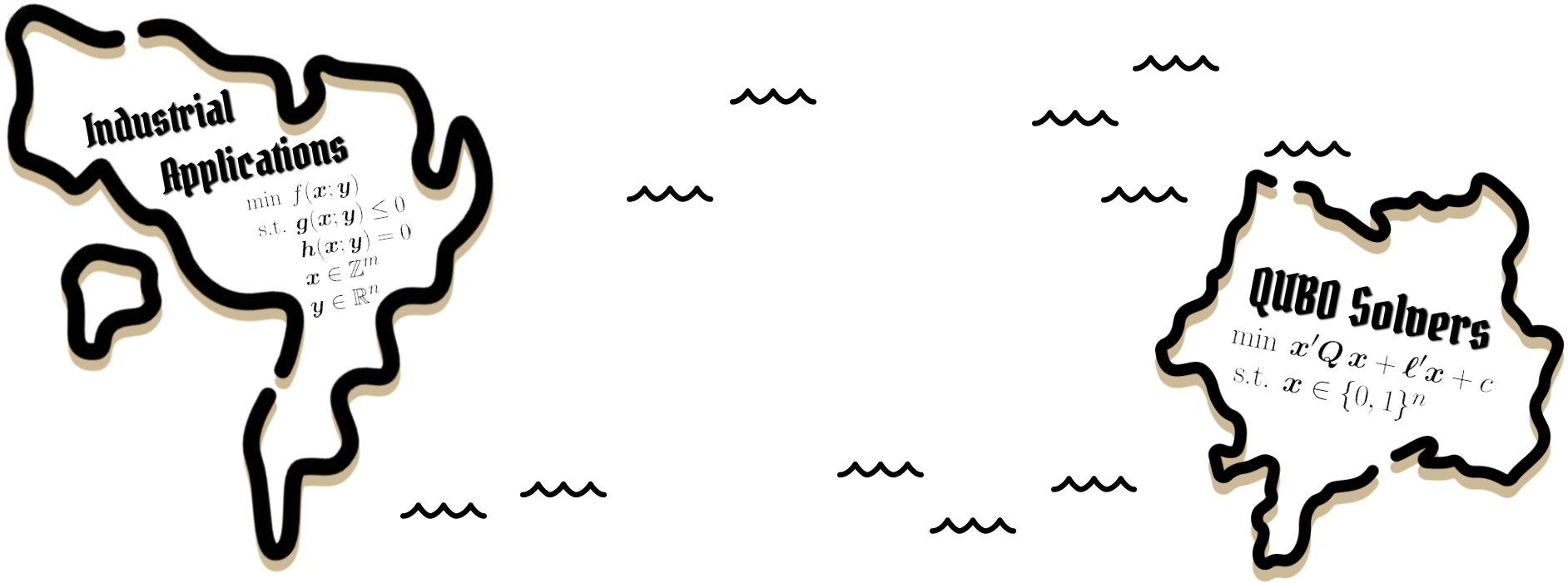
Toshiba, Goto et al., 2019



IBM


Fujitsu

# Our Goal



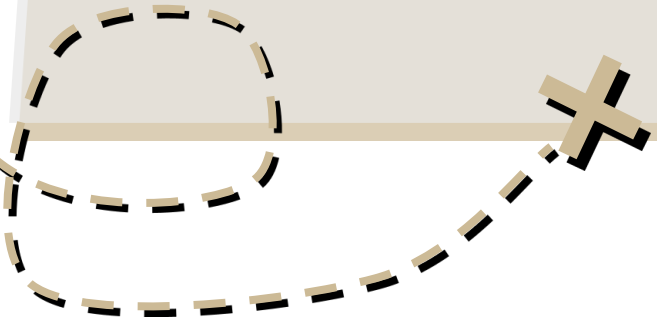
# Our Goal

# *Tasks*

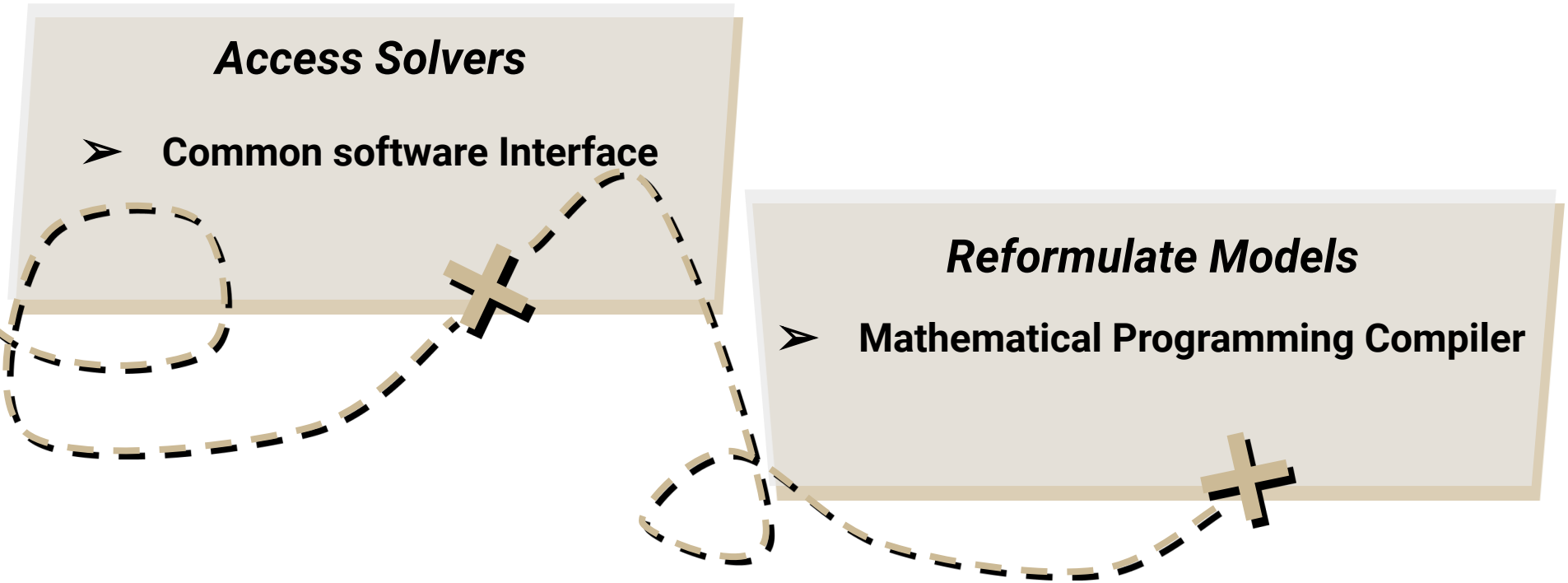
# Tasks

## Access Solvers

- Common software Interface



# Tasks



# Solution Overview

**JuMP Model (MINLP)**

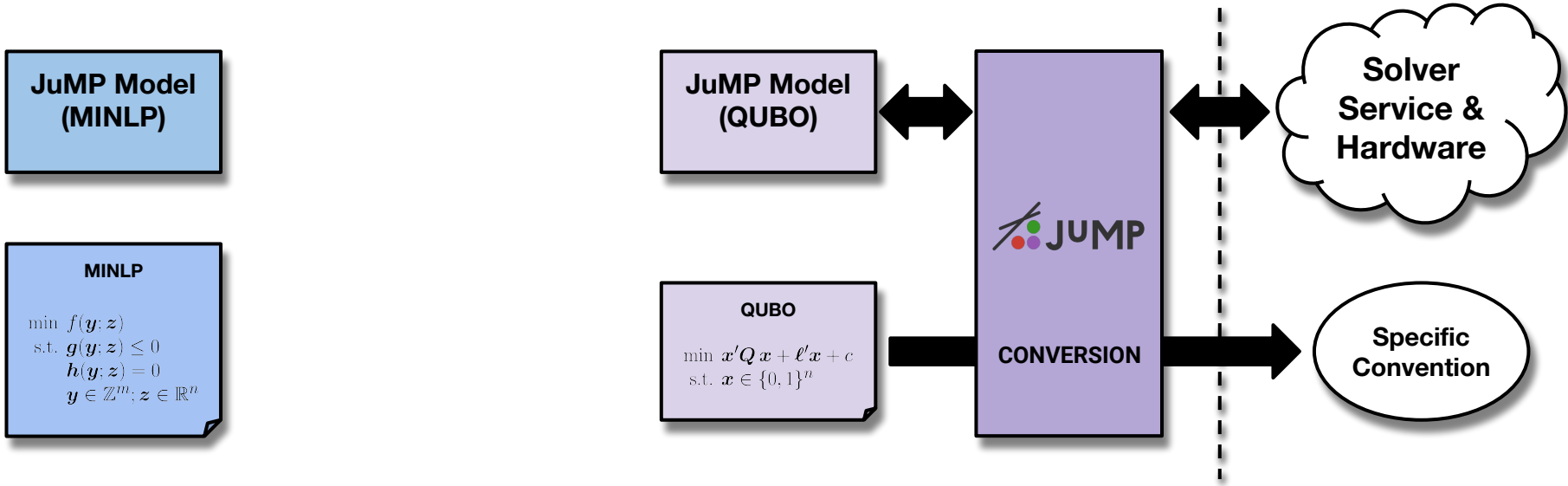
**MINLP**

$$\begin{aligned} \min\ & f(\boldsymbol{y}; \boldsymbol{z}) \\ \text{s.t. } & \boldsymbol{g}(\boldsymbol{y}; \boldsymbol{z}) \leq 0 \\ & \boldsymbol{h}(\boldsymbol{y}; \boldsymbol{z}) = 0 \\ & \boldsymbol{y} \in \mathbb{Z}^m; \boldsymbol{z} \in \mathbb{R}^n \end{aligned}$$

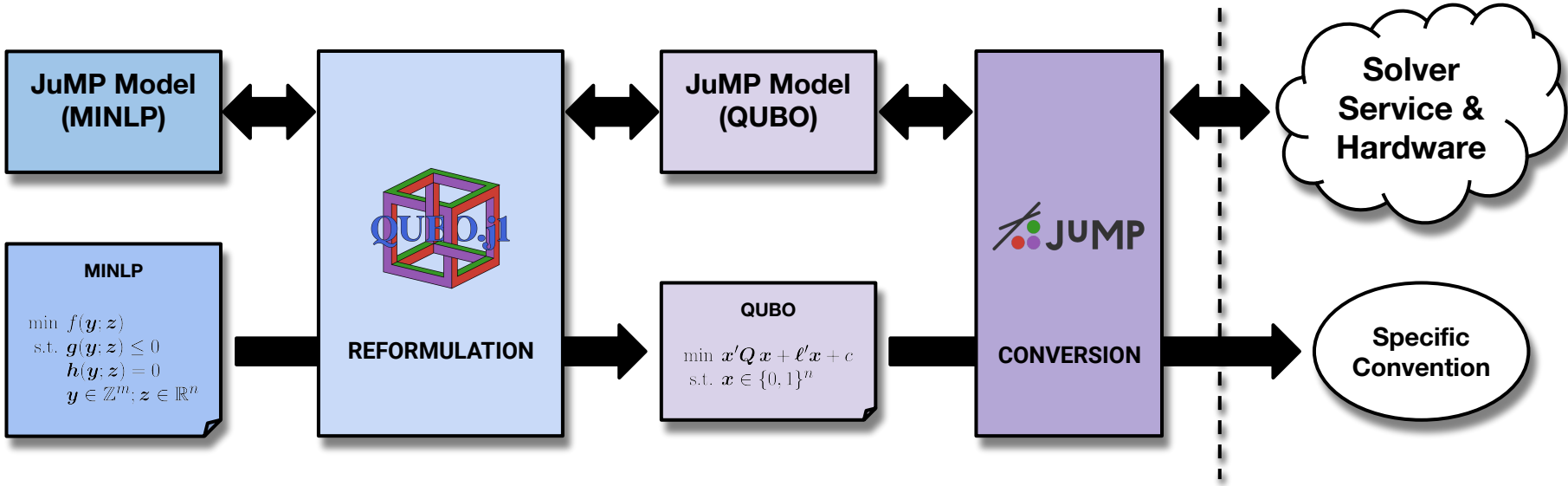
**Solver Service & Hardware**

**Specific Convention**

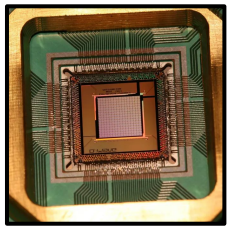
# Solution Overview



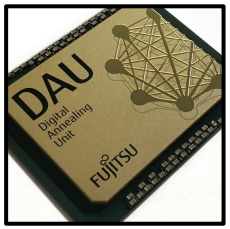
# Solution Overview



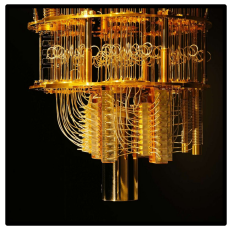
# Integrating an heterogeneous Solver Landscape



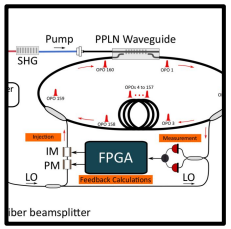
D-Wave



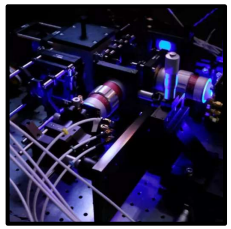

Fujitsu



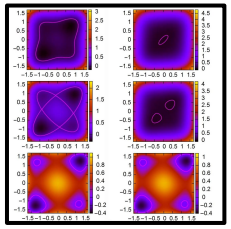
IBM




P. L. McMahon et al., 2016

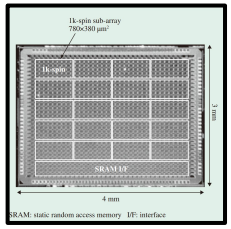


Microsoft Research



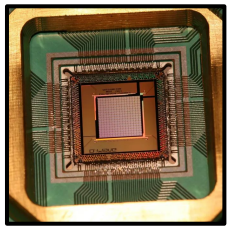
Toshiba, Goto et al., 2019

**Quantum Annealing, Digital Annealing, Variational Quantum Eigensolver, Quantum Alternating Optimization Ansatz, Coherent Ising Machine, Analog Iterative Machine, Simulated Bifurcation Machine, CMOS Annealing…**

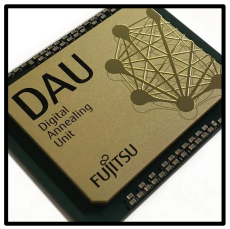



Hitachi, Yamaoka et al.

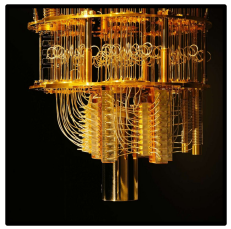
# Integrating an heterogeneous Solver Landscape



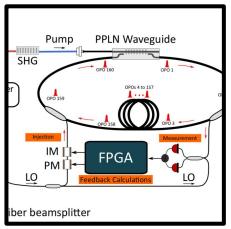
D-Wave



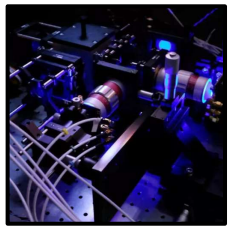

Fujitsu



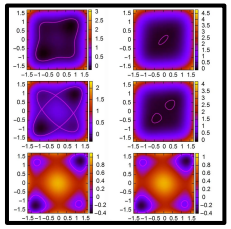
IBM



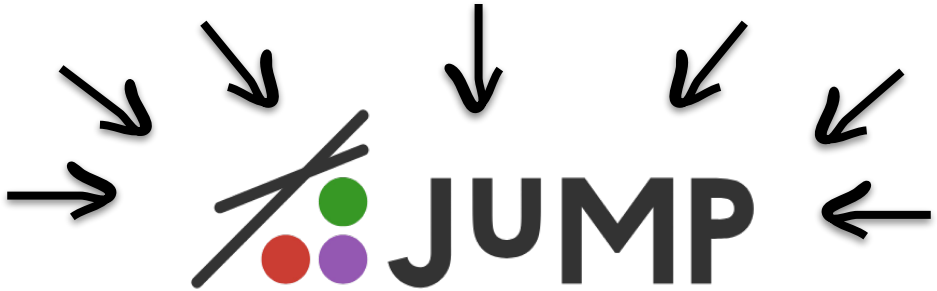

P. L. McMahon et al., 2016



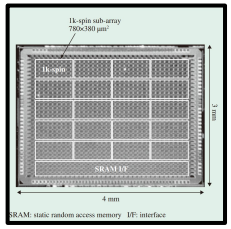
Microsoft Research



Toshiba, Goto et al., 2019



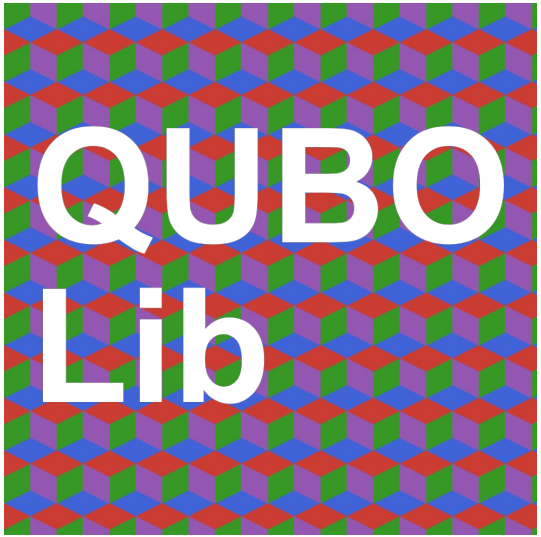

Julia Mathematical Programming



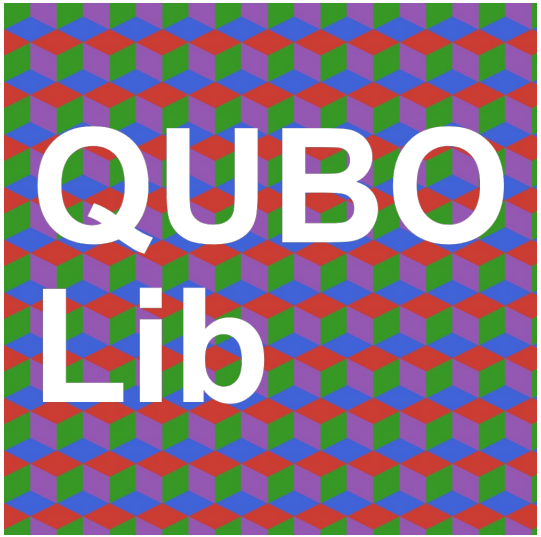

Hitachi, Yamaoka et al.

# A Common Solver Interface

# *Testing and Benchmarking Solvers*



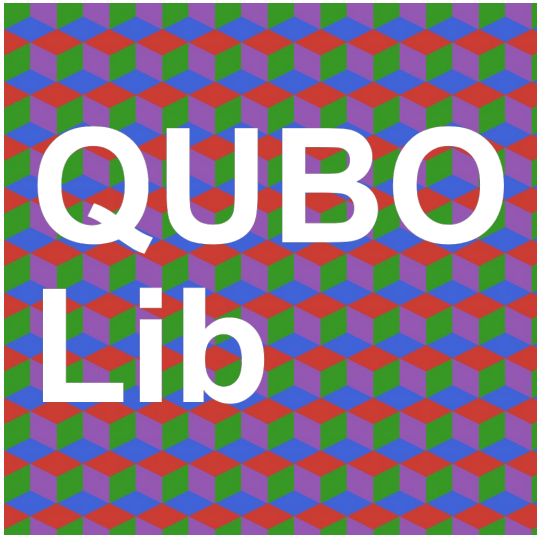
# *Testing and Benchmarking Solvers*




| Sources Summary | | |
|---|---|---|
| **Collection** | **Instances** | **Size Range** |
| **arXiv:2103.008464 (3R3X)** | **2300** | **16 - 4096** |
| **arXiv:1903.100928 (3R3X)** | **3200** | **16 - 4096** |
| **arXiv:1903.100928 (5R5X)** | **307** | **24 - 24576** |
| **qplib*** | **23** | **120 - 1225** |

*QPLIB: A Library of Quadratic Programming Instances, Mathematical Programming Computation, 2018

# *Testing and Benchmarking Solvers*




```julia
QUBOLib.load_index() do index
    db = QUBOLib.database(index)
    df = DBInterface.execute(
        db,
        """
        SELECT instance FROM Instances
        WHERE dimension < 100 AND quadratic_density < 0.5;
        """,
    ) |> DataFrame

    codes = collect(Int, df[!, :instance])

    @info "Running DWave Neal"
    QUBOLib.run!(
        index, DWave.Neal.Optimizer, codes; solver = "dwave-neal"
    )

    @info "Running DWave (Quantum)"
    QUBOLib.run!(
        index, DWave.Optimizer, codes; solver = "dwave"
    )
end
```


$$\begin{aligned} \min \ & f(\boldsymbol{y}; \boldsymbol{z}) \\ \text{s.t.} \ & g_i(\boldsymbol{y}; \boldsymbol{z}) \in S_i \\ & \boldsymbol{y} \in \mathbb{Z}^m; \boldsymbol{z} \in \mathbb{R}^n \end{aligned}$$


**VARIABLE ENCODING**

$$\begin{aligned} \min\ & f(\boldsymbol{y};\boldsymbol{z}) \\ \text{s.t.}\ & g_i(\boldsymbol{y};\boldsymbol{z}) \in S_i \\ & \boldsymbol{y} \in \mathbb{Z}^m; \boldsymbol{z} \in \mathbb{R}^n \end{aligned}$$

$$\begin{aligned} \min\ & f(\boldsymbol{x}) \\ \text{s.t.}\ & g_i(\boldsymbol{x}) \in S_i \\ & \boldsymbol{x} \in \{0,1\}^k \end{aligned}$$


**VARIABLE ENCODING**

$$\begin{aligned} \min\ & f(\boldsymbol{y};\boldsymbol{z}) \\ \text{s.t.}\ & g_i(\boldsymbol{y};\boldsymbol{z}) \in S_i \\ & \boldsymbol{y} \in \mathbb{Z}^m; \boldsymbol{z} \in \mathbb{R}^n \end{aligned}$$

$$\begin{aligned} \min\ & f(\boldsymbol{x}) \\ \text{s.t.}\ & g_i(\boldsymbol{x}) \in S_i \\ & \boldsymbol{x} \in \{0,1\}^k \end{aligned}$$

**CONSTRAINT PENALIZATION**

$$\begin{aligned} \min\ & f(\boldsymbol{x}) + \textstyle\sum_i \rho_i \left\| g_i(\boldsymbol{x}) - S_i \right\| \\ \text{s.t.}\ & \boldsymbol{x} \in \{0,1\}^k \end{aligned}$$


**VARIABLE ENCODING**

$$\begin{aligned}\min\ & f(\boldsymbol{y};\boldsymbol{z})\\ \text{s.t. }& g_i(\boldsymbol{y};\boldsymbol{z}) \in S_i\\ & \boldsymbol{y}\in\mathbb{Z}^m;\boldsymbol{z}\in\mathbb{R}^n\end{aligned}$$

$$\begin{aligned}\min\ & f(\boldsymbol{x})\\ \text{s.t. }& g_i(\boldsymbol{x}) \in S_i\\ & \boldsymbol{x}\in\{0,1\}^k\end{aligned}$$

**CONSTRAINT PENALIZATION**

**PSEUDO-BOOLEAN MAPPING**

$$\begin{aligned}\min\ & f(\boldsymbol{x}) + \textstyle\sum_i \rho_i \left\| g_i(\boldsymbol{x}) - S_i \right\|\\ \text{s.t. }& \boldsymbol{x}\in\{0,1\}^k\end{aligned}$$

$$\begin{aligned}\min\ & \varphi(\boldsymbol{x}) = \sum_{\omega} c_\omega \prod_{j\in\omega} x_j\\ \text{s.t. }& \boldsymbol{x}\in\{0,1\}^k\end{aligned}$$


**VARIABLE ENCODING**

$$\begin{aligned}\min\ & f(\boldsymbol{y};\boldsymbol{z})\\ \text{s.t. }& g_i(\boldsymbol{y};\boldsymbol{z}) \in S_i\\ & \boldsymbol{y}\in\mathbb{Z}^m;\boldsymbol{z}\in\mathbb{R}^n\end{aligned}$$

$$\begin{aligned}\min\ & f(\boldsymbol{x})\\ \text{s.t. }& g_i(\boldsymbol{x}) \in S_i\\ & \boldsymbol{x}\in\{0,1\}^k\end{aligned}$$

**QUADRATIZATION**

$$\begin{aligned}\min\ & \psi(\boldsymbol{x};\boldsymbol{w}) = [\boldsymbol{x};\boldsymbol{w}]'Q[\boldsymbol{x};\boldsymbol{w}]\\ \text{s.t. }& [\boldsymbol{x};\boldsymbol{w}]\in\{0,1\}^{k+p}\end{aligned}$$

**CONSTRAINT PENALIZATION**

**PSEUDO-BOOLEAN MAPPING**

$$\begin{aligned}\min\ & f(\boldsymbol{x}) + \textstyle\sum_i \rho_i \left\| g_i(\boldsymbol{x}) - S_i \right\|\\ \text{s.t. }& \boldsymbol{x}\in\{0,1\}^k\end{aligned}$$

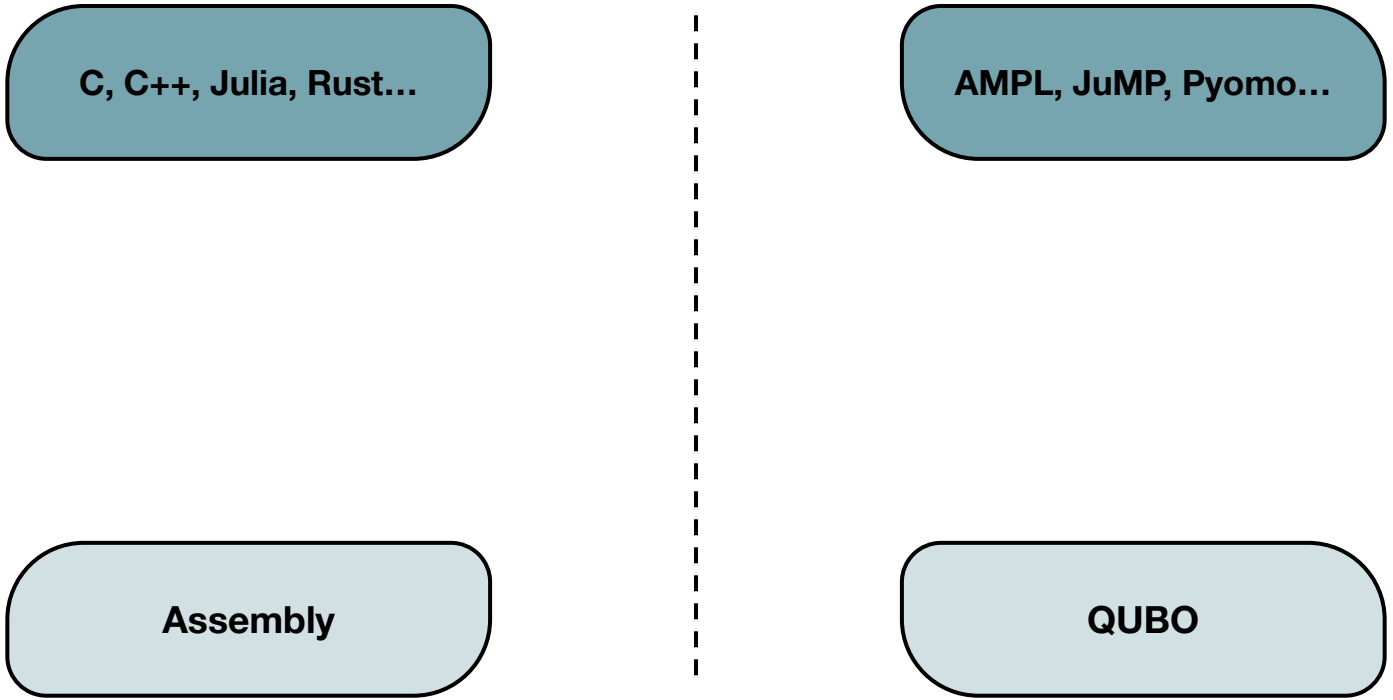
$$\begin{aligned}\min\ & \varphi(\boldsymbol{x}) = \sum_{\omega} c_\omega \prod_{j\in\omega} x_j\\ \text{s.t. }& \boldsymbol{x}\in\{0,1\}^k\end{aligned}$$

# A Compiler for Mathematical Programming

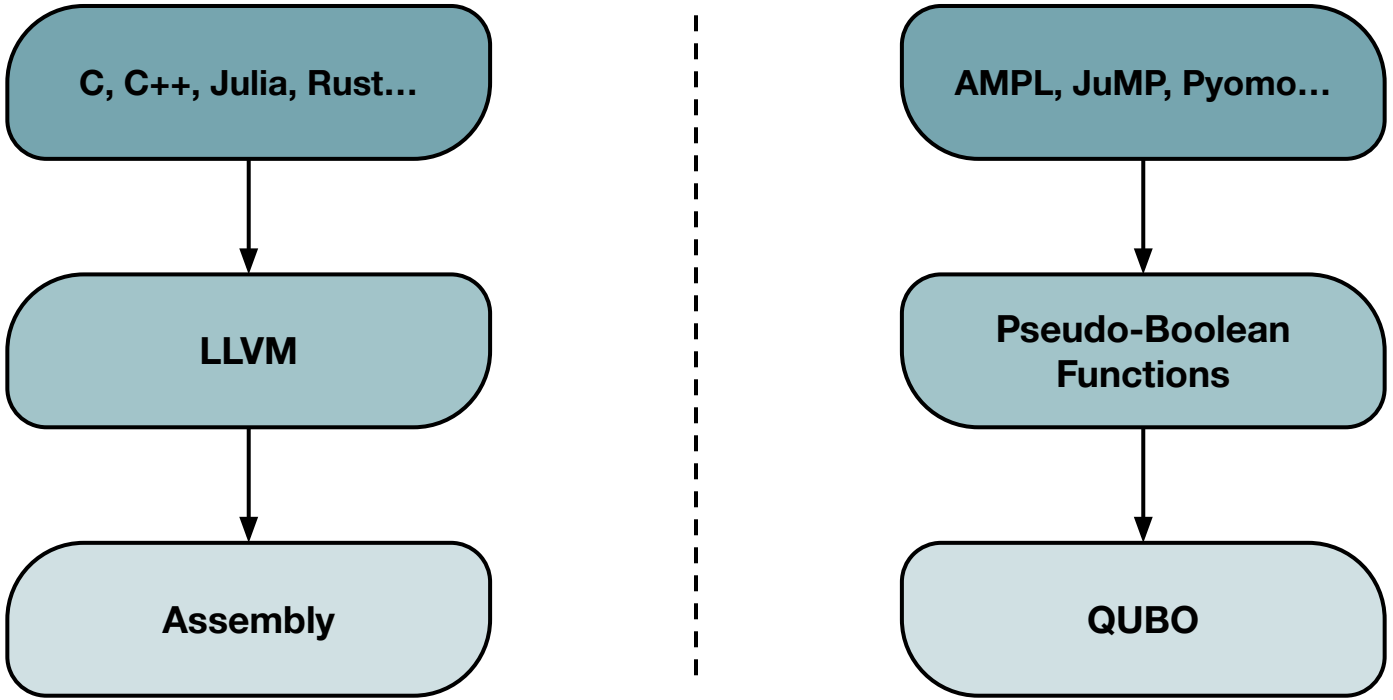
C, C++, Julia, Rust…

AMPL, JuMP, Pyomo…

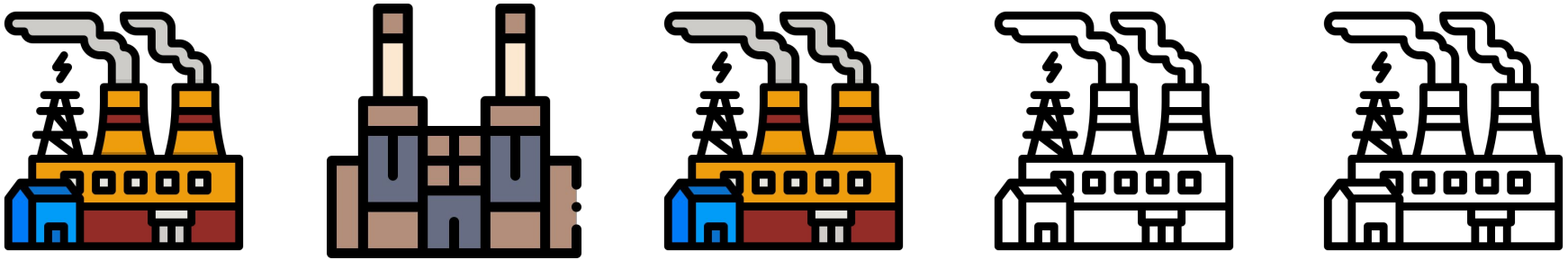
# A Compiler for Mathematical Programming



# A Compiler for Mathematical Programming



# Example: Generation capacity expansion



# Example: Generation capacity expansion

$$\min_{\mathbf{g},\mathbf{u},\mathbf{x}} \sum_t \mathbf{c}'\mathbf{g}^{(t)} + \mathbf{i}'\mathbf{x}$$

$$\text{s.a.} \sum_j g_j^{(t)} = d^{(t)} \qquad \forall t$$

$$g_j^{(t)} \leq u_j^{(t)} G_j^{(\max)} \qquad \forall j, t$$

$$u_j^{(t)} \leq x_j \qquad \forall j, t$$

$$g_j^{(t)} \in [0, G_j^{(\max)}] \qquad \forall j, t$$

$$u_j^{(t)} \in \{0, 1\} \qquad \forall j, t$$

$$x_j \in \{0, 1\} \qquad \forall j$$

# Example: Generation capacity expansion

OPERATION

$$\min_{\mathbf{g},\mathbf{u},\mathbf{x}} \sum_t \mathbf{c}'\mathbf{g}^{(t)} + \mathbf{i}'\mathbf{x}$$

$$\text{s.a.} \sum_j g_j^{(t)} = d^{(t)} \qquad \forall t$$

BALANCE

$$g_j^{(t)} \leq u_j^{(t)} G_j^{(\max)} \qquad \forall j,t$$

$$u_j^{(t)} \leq x_j \qquad \forall j,t$$

$$g_j^{(t)} \in [0, G_j^{(\max)}] \qquad \forall j,t$$

OPERATION

$$u_j^{(t)} \in \{0,1\} \qquad \forall j,t$$

$$x_j \in \{0,1\} \qquad \forall j$$

# Example: Generation capacity expansion

$$\min_{\mathbf{g},\mathbf{u},\mathbf{x}} \sum_t \mathbf{c}'\mathbf{g}^{(t)} + \mathbf{i}'\mathbf{x}$$

$$\text{s.a.} \sum_j g_j^{(t)} = d^{(t)} \qquad \forall t$$

$$g_j^{(t)} \leq u_j^{(t)} G_j^{(\max)} \qquad \forall j,t$$

UNIT COMMITMENT

$$u_j^{(t)} \leq x_j \qquad \forall j,t$$

$$g_j^{(t)} \in [0, G_j^{(\max)}] \qquad \forall j,t$$

$$u_j^{(t)} \in \{0,1\} \qquad \forall j,t$$

UNIT COMMITMENT

$$x_j \in \{0,1\} \qquad \forall j$$

# Example: Generation capacity expansion

$$\min_{\mathbf{g},\mathbf{u},\mathbf{x}} \sum_t \mathbf{c}'\mathbf{g}^{(t)} + \mathbf{i}'\mathbf{x}$$

INVESTMENT

$$\text{s.a.} \sum_j g_j^{(t)} = d^{(t)} \qquad \forall t$$

$$g_j^{(t)} \leq u_j^{(t)} G_j^{(\max)} \qquad \forall j,t$$

$$u_j^{(t)} \leq x_j \qquad \forall j,t$$

INVESTMENT

$$g_j^{(t)} \in [0, G_j^{(\max)}] \qquad \forall j,t$$

$$u_j^{(t)} \in \{0,1\} \qquad \forall j,t$$

$$x_j \in \{0,1\} \qquad \forall j$$

INVESTMENT

# Example: Generation capacity expansion

$$\min_{\mathbf{g},\mathbf{u},\mathbf{x}} \underbrace{\sum_t \mathbf{c}'\mathbf{g}^{(t)}}_{\text{OPERATION}} + \underbrace{\mathbf{i}'\mathbf{x}}_{\text{INVESTMENT}}$$

$$\text{s.a.} \quad \sum_j g_j^{(t)} = d^{(t)} \qquad \forall t \qquad \text{BALANCE}$$

$$g_j^{(t)} \leq u_j^{(t)} G_j^{(\max)} \qquad \forall j,t \qquad \text{UNIT COMMITMENT}$$

$$u_j^{(t)} \leq x_j \qquad \forall j,t \qquad \text{INVESTMENT}$$

$$g_j^{(t)} \in [0, G_j^{(\max)}] \qquad \forall j,t \qquad \text{OPERATION}$$

$$u_j^{(t)} \in \{0,1\} \qquad \forall j,t \qquad \text{UNIT COMMITMENT}$$

$$x_j \in \{0,1\} \qquad \forall j \qquad \text{INVESTMENT}$$

# Example: Generation capacity expansion

OPERATION

INVESTMENT

$$\min_{\mathbf{g},\mathbf{u},\mathbf{x}} \sum_t \mathbf{c}'\mathbf{g}^{(t)} + \mathbf{i}'\mathbf{x}$$

$$\text{s.a.} \sum_j g_j^{(t)} = d^{(t)} \quad \forall t$$

BALANCE

$$g_j^{(t)} \leq u_j^{(t)} G_j^{(\max)} \quad \forall j,t$$

UNIT COMMITMENT

$$u_j^{(t)} \leq x_j \quad \forall j,t$$

INVESTMENT

$$g_j^{(t)} \in [0, G_j^{(\max)}] \quad \forall j,t$$

OPERATION

$$u_j^{(t)} \in \{0,1\} \quad \forall j,t$$

UNIT COMMITMENT

$$x_j \in \{0,1\} \quad \forall j$$

INVESTMENT

***REFORMULATION***

$$\rho_{balance} \left( \sum_t \left( \sum_j g_j^{(t)} - d^{(t)} \right)^2 \right)$$

$$\rho_{invest} \left( \sum_{j,t} \left( g_j^{(t)} + s_{UB} - x_j G_j^{(max)} \right)^2 \right)$$

$$g_j^{(t)} = \alpha \sum_k 2^k y_{k,j}^{(t)} \text{ s.t. } y_{k,j}^{(t)} \in \{0,1\}$$

# Example: Generation capacity expansion

OPERATION

INVESTMENT

$$\min_{\mathbf{g},\mathbf{u},\mathbf{x}} \sum_t \mathbf{c}'\mathbf{g}^{(t)} + \mathbf{i}'\mathbf{x}$$

$$\text{s.a.} \sum_j g_j^{(t)} = d^{(t)} \qquad \forall t$$

BALANCE

$$g_j^{(t)} \leq u_j^{(t)} G_j^{(\max)} \qquad \forall j,t$$

UNIT COMMITMENT

$$u_j^{(t)} \leq x_j \qquad \forall j,t$$

INVESTMENT

$$g_j^{(t)} \in [0, G_j^{(\max)}] \qquad \forall j,t$$

OPERATION

$$u_j^{(t)} \in \{0,1\} \qquad \forall j,t$$

UNIT COMMITMENT

$$x_j \in \{0,1\} \qquad \forall j$$

INVESTMENT

```julia
using JuMP
using PySA

model = Model(PySA.Optimizer)

@variable(model, 0 ⩽ g[1:T,j=1:n] ⩽ Gmax[j])
@variable(model, u[1:T,1:n], Bin)
@variable(model, x[1:n], Bin)

@objective(model, Min, sum(c'g[t,:] for t=1:T) + i'x)

@constraint(model, [t=1:T], sum(g[t,:]) == d[t])
@constraint(model, [t=1:T,j=1:n], g[t,j] ⩽ u[t,j] * Gmax[j])
@constraint(model, [t=1:T,j=1:n], u[t,j] ⩽ x[j])

optimize!(model)

@show objective_value(model)
@show value.(x)
```

# Example: Generation capacity expansion

OPERATION INVESTMENT

$$\min_{\mathbf{g},\mathbf{u},\mathbf{x}} \sum_t \mathbf{c}'\mathbf{g}^{(t)} + \mathbf{i}'\mathbf{x}$$

$$\text{s.a.} \sum_j g_j^{(t)} = d^{(t)} \qquad \forall t \qquad \text{BALANCE}$$

$$g_j^{(t)} \leq u_j^{(t)} G_j^{(\max)} \qquad \forall j,t \qquad \text{UNIT COMMITMENT}$$

$$u_j^{(t)} \leq x_j \qquad \forall j,t \qquad \text{INVESTMENT}$$

$$g_j^{(t)} \in [0, G_j^{(\max)}] \qquad \forall j,t \qquad \text{OPERATION}$$

$$u_j^{(t)} \in \{0,1\} \qquad \forall j,t \qquad \text{UNIT COMMITMENT}$$

$$x_j \in \{0,1\} \qquad \forall j \qquad \text{INVESTMENT}$$

```julia
using JuMP
using PySA

model = Model(PySA.Optimizer)

@variable(model, 0 ≤ g[1:T,j=1:n] ≤ Gmax[j])
@variable(model, u[1:T,1:n], Bin)
@variable(model, x[1:n], Bin)

@objective(model, Min, sum(c'g[t,:] for t=1:T) + i'x)

@constraint(model, [t=1:T], sum(g[t,:]) == d[t])
@constraint(model, [t=1:T,j=1:n], g[t,j] ≤ u[t,j] * Gmax[j])
@constraint(model, [t=1:T,j=1:n], u[t,j] ≤ x[j])

optimize!(model)

@show objective_value(model)
@show value.(x)
```

# Example: Generation capacity expansion

OPERATION

INVESTMENT

$$\min_{\mathbf{g},\mathbf{u},\mathbf{x}} \sum_t \mathbf{c}'\mathbf{g}^{(t)} + \mathbf{i}'\mathbf{x}$$

$$\text{s.a.} \sum_j g_j^{(t)} = d^{(t)} \qquad \forall t$$

BALANCE

$$g_j^{(t)} \leq u_j^{(t)} G_j^{(\max)} \qquad \forall j,t$$

UNIT COMMITMENT

$$u_j^{(t)} \leq x_j \qquad \forall j,t$$

INVESTMENT

$$g_j^{(t)} \in [0, G_j^{(\max)}] \qquad \forall j,t$$

OPERATION

$$u_j^{(t)} \in \{0,1\} \qquad \forall j,t$$

UNIT COMMITMENT

$$x_j \in \{0,1\} \qquad \forall j$$

INVESTMENT

```julia
using JuMP
using PySA

model = Model(PySA.Optimizer)

@variable(model, 0 ≤ g[1:T,j=1:n] ≤ Gmax[j])
@variable(model, u[1:T,1:n], Bin)
@variable(model, x[1:n], Bin)

@objective(model, Min, sum(c'g[t,:] for t=1:T) + i'x)

@constraint(model, [t=1:T], sum(g[t,:]) == d[t])
@constraint(model, [t=1:T,j=1:n], g[t,j] ≤ u[t,j] * Gmax[j])
@constraint(model, [t=1:T,j=1:n], u[t,j] ≤ x[j])

optimize!(model)

@show objective_value(model)
@show value.(x)
```

# Example: Generation capacity expansion

OPERATION INVESTMENT

$$\min_{\mathbf{g},\mathbf{u},\mathbf{x}} \sum_t \mathbf{c}'\mathbf{g}^{(t)} + \mathbf{i}'\mathbf{x}$$

$$\text{s.a.} \sum_j g_j^{(t)} = d^{(t)} \qquad \forall t \quad \text{BALANCE}$$

$$g_j^{(t)} \le u_j^{(t)} G_j^{(\max)} \qquad \forall j,t \quad \text{UNIT COMMITMENT}$$

$$u_j^{(t)} \le x_j \qquad \forall j,t \quad \text{INVESTMENT}$$

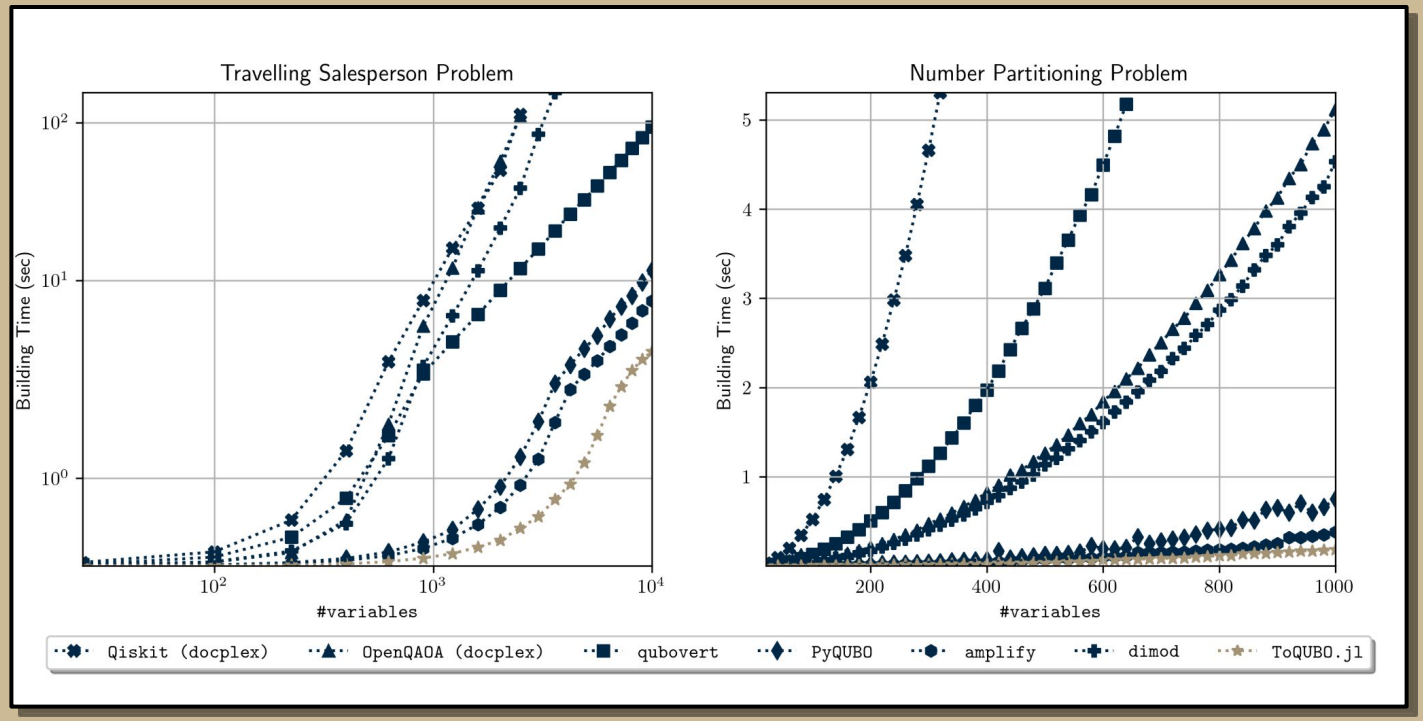
$$g_j^{(t)} \in [0, G_j^{(\max)}] \qquad \forall j,t \quad \text{OPERATION}$$

$$u_j^{(t)} \in \{0,1\} \qquad \forall j,t \quad \text{UNIT COMMITMENT}$$

$$x_j \in \{0,1\} \qquad \forall j \quad \text{INVESTMENT}$$

```julia
using JuMP, QUBO
using PySA

model = Model(() -> ToQUBO.Optimizer(PySA.Optimizer))

@variable(model, 0 ≤ g[1:T,j=1:n] ≤ Gmax[j])
@variable(model, u[1:T,1:n], Bin)
@variable(model, x[1:n], Bin)

@objective(model, Min, sum(c'g[t,:] for t=1:T) + i'x)

@constraint(model, [t=1:T], sum(g[t,:]) == d[t])
@constraint(model, [t=1:T,j=1:n], g[t,j] ≤ u[t,j] * Gmax[j])
@constraint(model, [t=1:T,j=1:n], u[t,j] ≤ x[j])

optimize!(model)

@show objective_value(model)
@show value.(x)
```

# Reformulation Performance



# Qualitative Analysis


TABLE I: Comparing the support provided by of proposed framework and existing libraries and framework in each step of quantum optimization.

✓ indicates that the corresponding action is performed automatically.
✓ signifies that a proper function is available for implementing the step.
✗ indicates that the method is not fully supported.
$^{+}$ denotes that logarithmic encoding is also compatible with bases different from two.
$^{*}$ signifies that the encoding techniques can be exploited only for constraints translation.
$^{\dagger}$ indicates that the polynomial reduction is implemented by exploiting the corresponding qubovert function.

| Supports for each step | | Existing Libraries | | | | | | Existing Frameworks | | Proposed |
|---|---|---|---|---|---|---|---|---|---|---|
| | | pyqubo [31] | qubovert [32] | dimod [33] | Qiskit [34] | fixstars [35] | openQAOA [36] | AutoQUBO [37] | QUBO.jl [38] | Framework |
| Floating Encoding | | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✓ |
| Integer Encoding | Logarithmic [39] | ✓ | ✓ | ✓ | ✓ | ✓* | ✗ | ✗ | ✓ | ✓+ |
| | Unitary [39] | ✓ | ✓ | ✗ | ✗ | ✓* | ✗ | ✗ | ✓ | ✓ |
| | Dictionary [39] | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✓ |
| | Domain-Wall [40] | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✓ |
| | Bounded-Coeff [41] | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✓ |
| | Arithmetic [42] | ✗ | ✗ | ✗ | ✗ | ✓* | ✗ | ✗ | ✓ | ✓ |
| Penalty Functions | Equality [22] [21] | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ |
| | Inequality [22] | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ |
| | Boolean [22] | ✓ | ✓ | ✓ | ✗ | ✓ | ✗ | ✗ | ✗ | ✓ |
| Penalty Weight | UB positive [43] | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ |
| | MQC [43] | ✗ | ✗ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ |
| | VLM [44] | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✗ | ✓ |
| | MOMC [43] | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ |
| | MOC [43] | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ |
| | UB Naive [45], [46] | ✗ | ✓ | ✗ | ✓ | ✗ | ✗ | ✓ | ✗ | ✓ |
| | UB posiform [45], [46] | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✗ | ✓ |
| Polynomial Reduction | | ✓ | ✓ | ✓ | ✗ | ✓ | ✗ | ✓ | ✓ | ✓† |
| Solvers | Dwave QA | ✓ | ✗ | ✓ | ✗ | ✓ | ✗ | ✓ | ✓ | ✓ |
| | QAOA | ✗ | ✗ | ✗ | ✓ | ✗ | ✓ | ✗ | ✓ | ✓ |
| | VQE | ✗ | ✗ | ✗ | ✓ | ✗ | ✓ | ✗ | ✓ | ✓ |
| | GAS | ✗ | ✗ | ✗ | ✓ | ✗ | ✗ | ✗ | ✗ | ✓ |
| | SA | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ | ✓ |
| Solution Decoding | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Check Constraints | | ✓ | ✓ | ✓ | ✗ | ✓ | ✗ | ✓ | ✓ | ✓ |
| Penalty Update | Sequential [47] | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ |
| | Scaled [47] | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ |
| | Binary search [47] | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ |

***Towards an Automatic Framework for Solving Optimization Problems with Quantum Computers***, arXiv:2406.12840, 2024

# *Conclusions & Future Work*

**MAIN TAKEAWAYS**

- ❏ ***QUBO.jl gives OR practitioners a smooth experience for accessing advanced hardware***
- ❏ ***Provides tools & infrastructure for benchmarking novel optimization technologies***

# *Conclusions & Future Work*

**MAIN TAKEAWAYS**

- ❑ ***QUBO.jl gives OR practitioners a smooth experience for accessing advanced hardware***
- ❑ ***Provides tools & infrastructure for benchmarking novel optimization technologies***

**NEXT STEPS**

- ❑ ***Set up benchmarking service and use results to guide reformulation***
- ❑ ***Expand Reformulation Library and Solver support***
- ❑ ***Investigate architecture-oriented compilation***

# Thanks!

pmacielx@purdue.edu
pedrox@cos.ufrj.br