# QUBO.jl

*A Julia ecosystem for*
*Quadratic Unconstrained Binary Optimization*

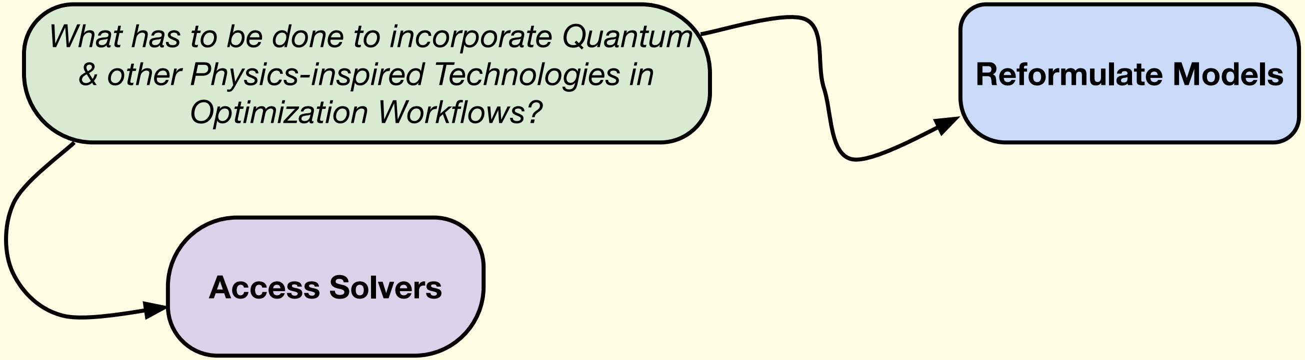
**Pedro Maciel Xavier**

Purdue University
Federal University of Rio de Janeiro

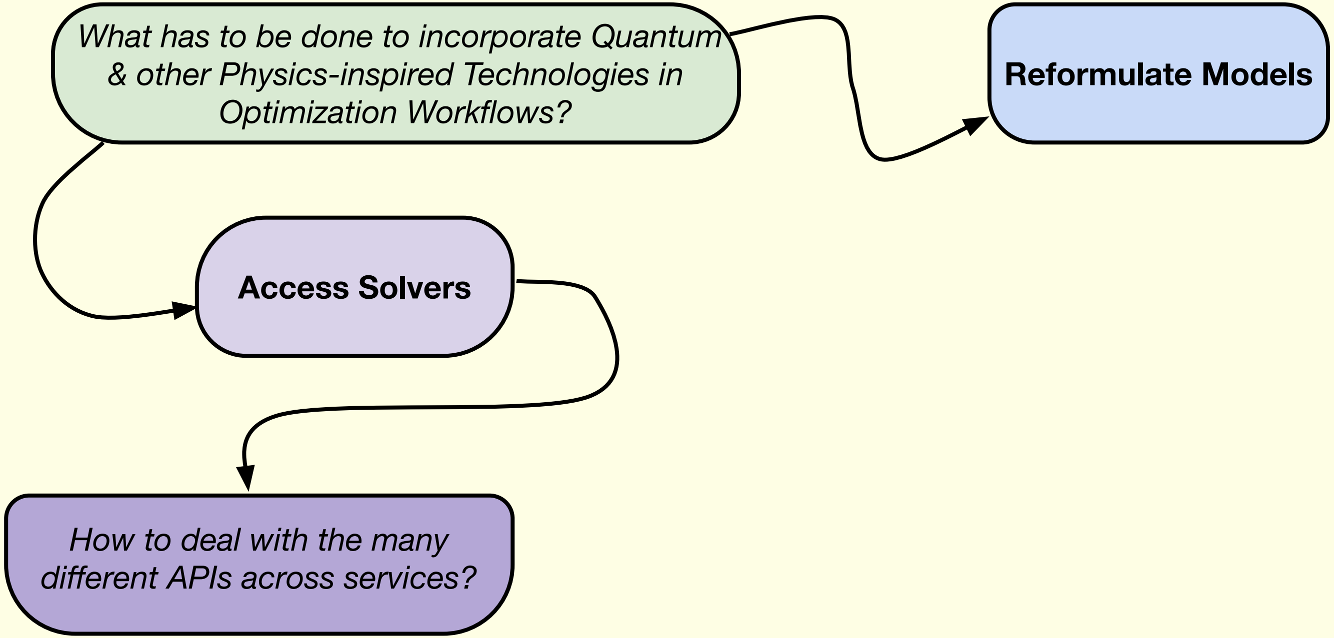
# Summary

*What has to be done to incorporate Quantum & other Physics-inspired Technologies in Optimization Workflows?*

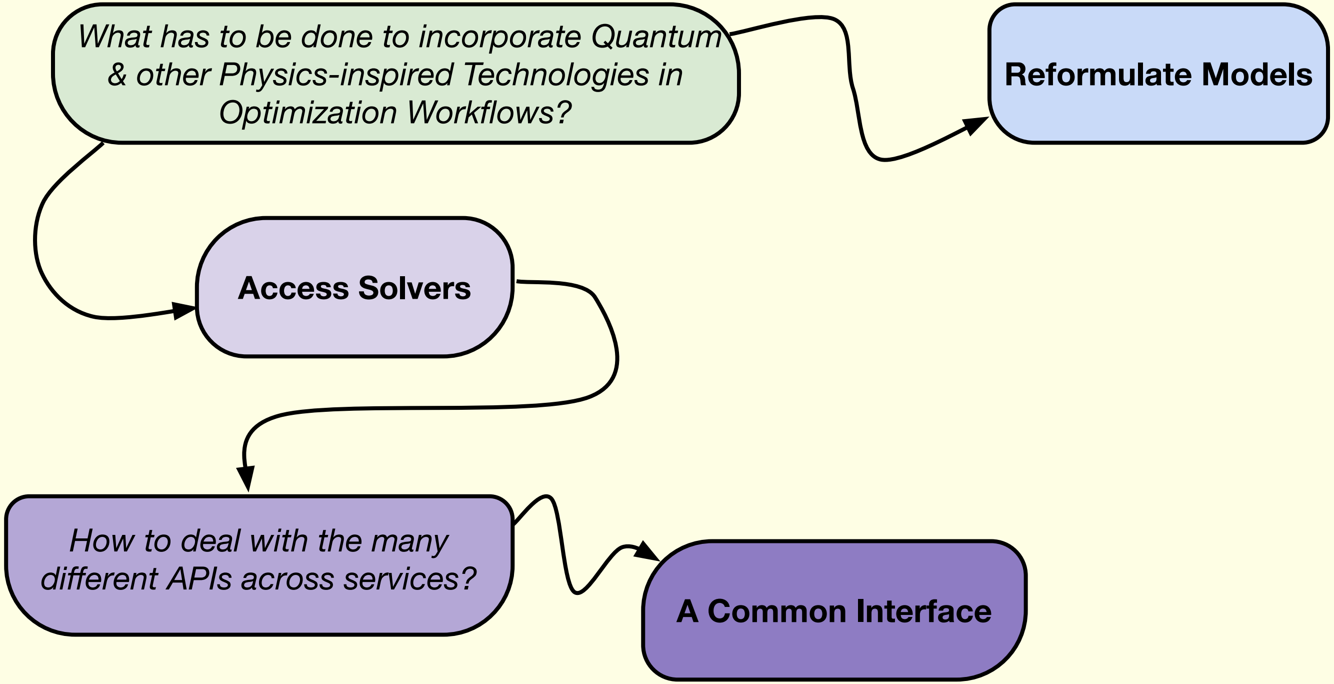
# Summary


What has to be done to incorporate Quantum & other Physics-inspired Technologies in Optimization Workflows?
Reformulate Models
Access Solvers

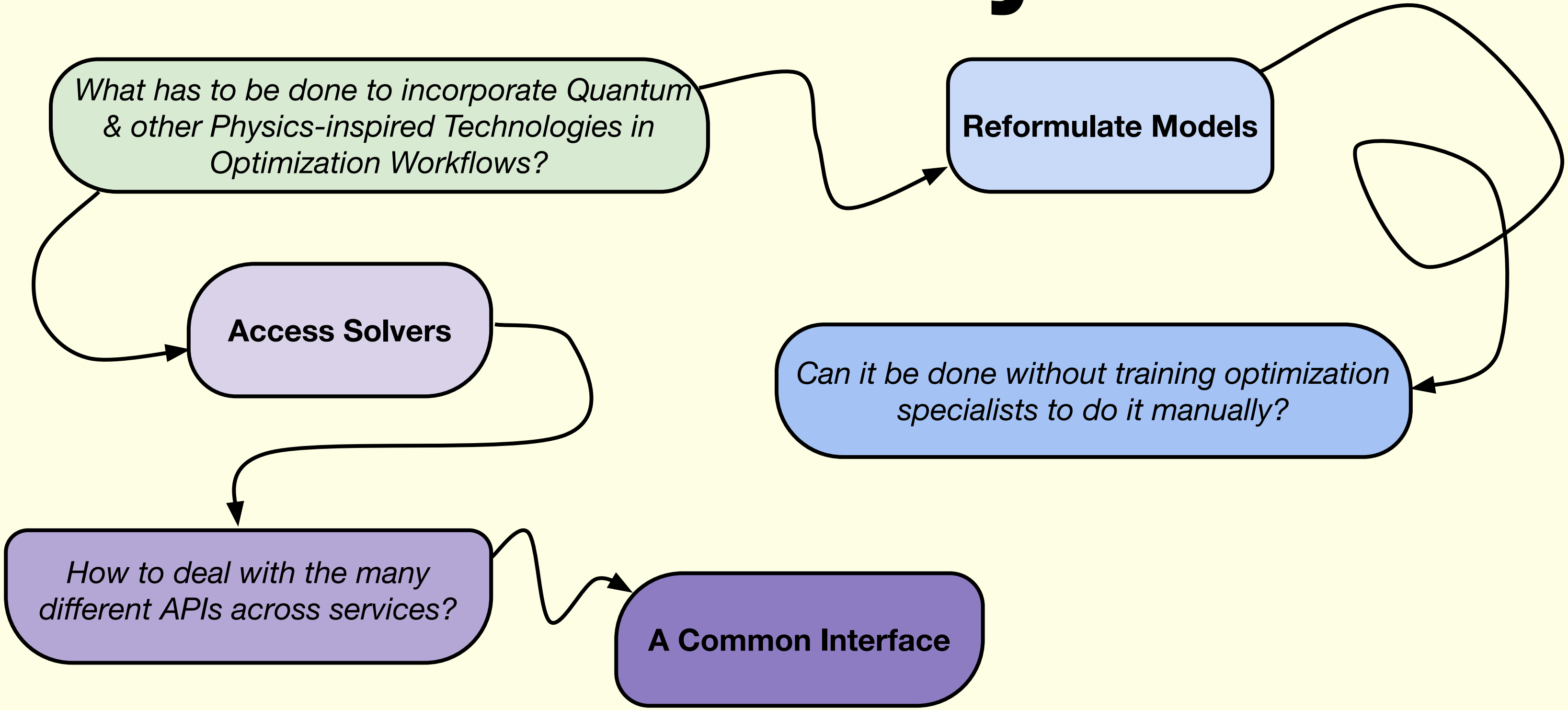
# Summary


What has to be done to incorporate Quantum & other Physics-inspired Technologies in Optimization Workflows?
Reformulate Models
Access Solvers
How to deal with the many different APIs across services?

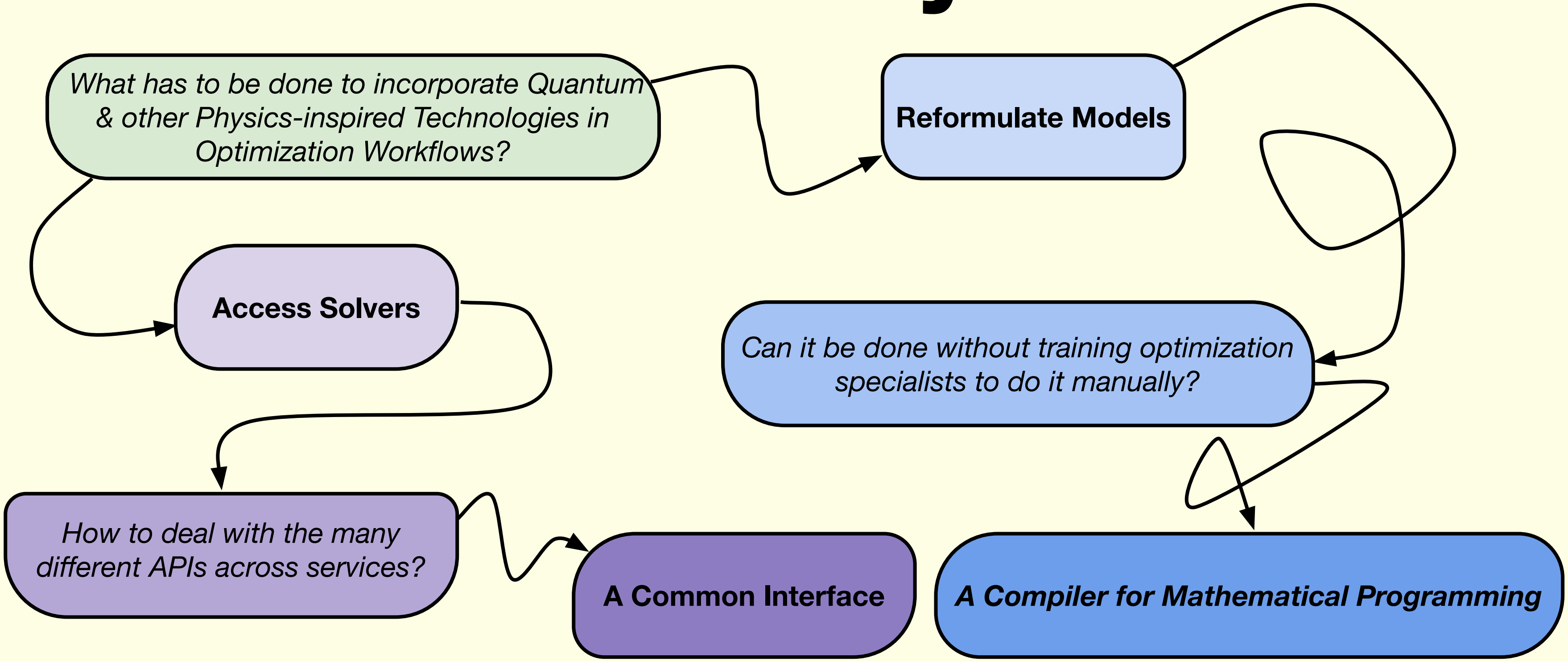
# Summary


What has to be done to incorporate Quantum & other Physics-inspired Technologies in Optimization Workflows?
Reformulate Models
Access Solvers
How to deal with the many different APIs across services?
A Common Interface

# Summary


What has to be done to incorporate Quantum & other Physics-inspired Technologies in Optimization Workflows?
Reformulate Models
Can it be done without training optimization specialists to do it manually?
Access Solvers
How to deal with the many different APIs across services?
A Common Interface

# Summary

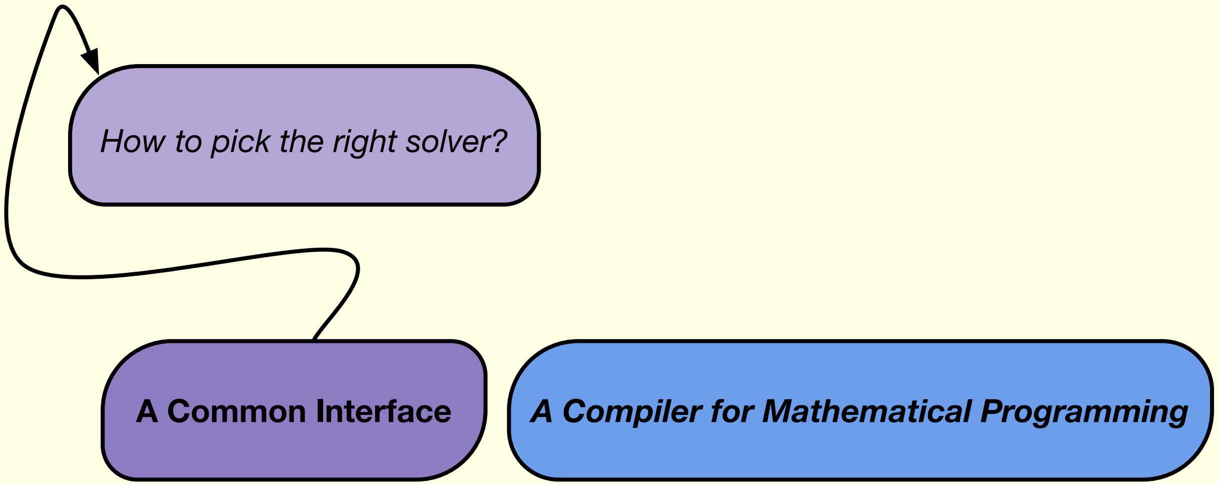

What has to be done to incorporate Quantum & other Physics-inspired Technologies in Optimization Workflows?
Reformulate Models
Access Solvers
Can it be done without training optimization specialists to do it manually?
How to deal with the many different APIs across services?
A Common Interface
A Compiler for Mathematical Programming

# Summary

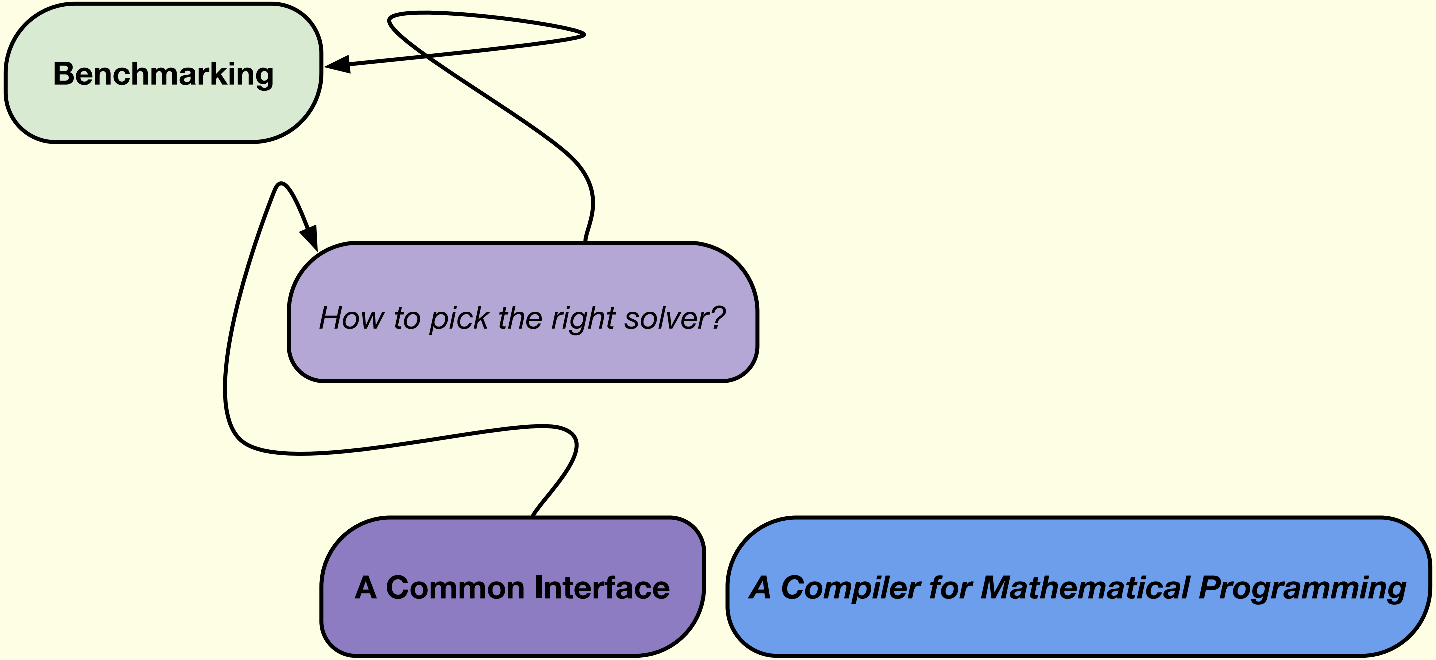
**A Common Interface**

***A Compiler for Mathematical Programming***

# Summary

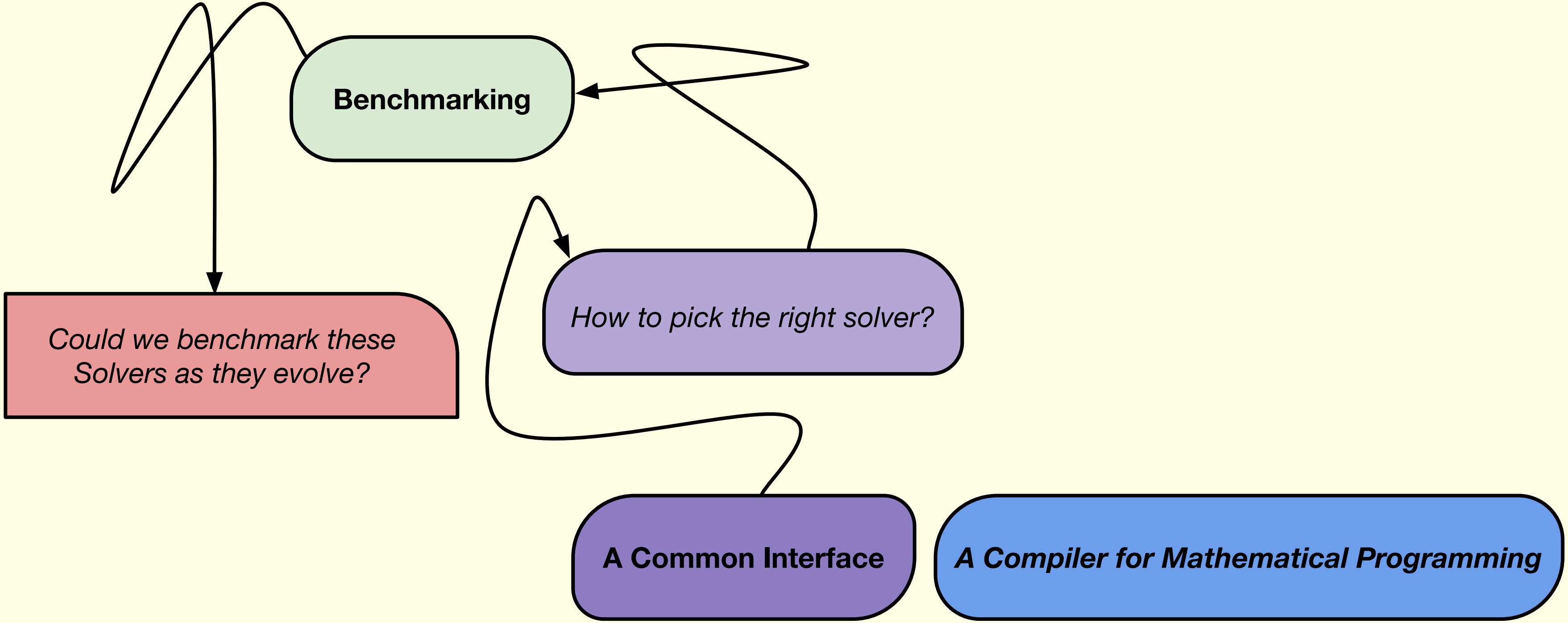

How to pick the right solver?
A Common Interface
A Compiler for Mathematical Programming

# Summary



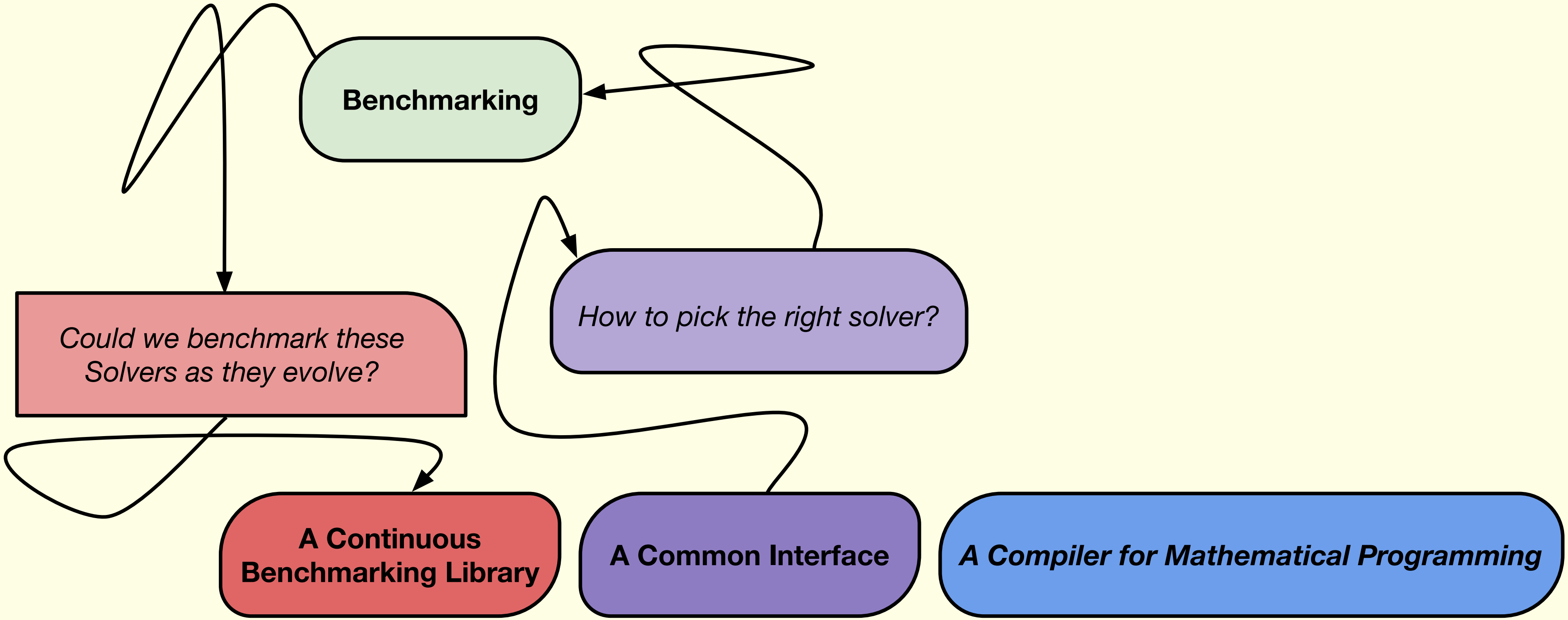
Benchmarking
How to pick the right solver?
A Common Interface
A Compiler for Mathematical Programming

# Summary



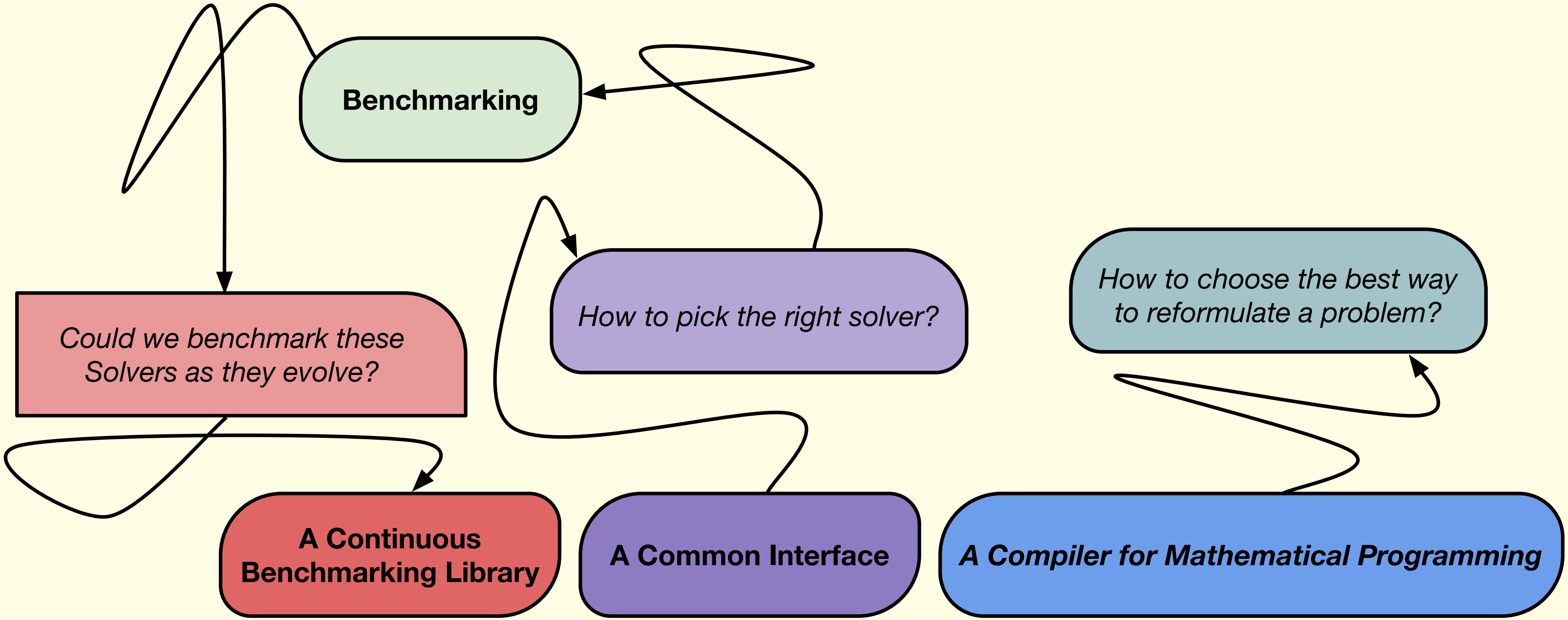
Benchmarking
Could we benchmark these Solvers as they evolve?
How to pick the right solver?
A Common Interface
A Compiler for Mathematical Programming

# Summary

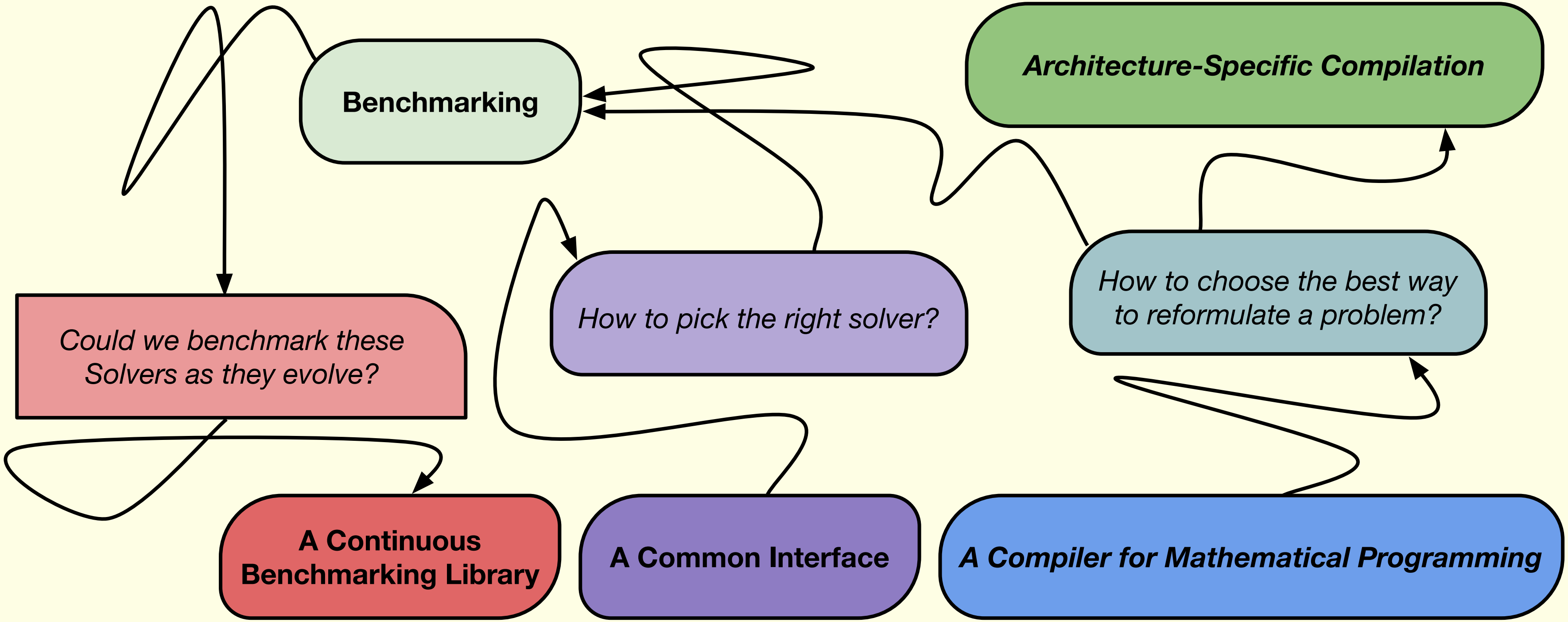


Benchmarking
Could we benchmark these Solvers as they evolve?
How to pick the right solver?
A Continuous Benchmarking Library
A Common Interface
A Compiler for Mathematical Programming

# Summary


Benchmarking
How to pick the right solver?
How to choose the best way to reformulate a problem?
Could we benchmark these Solvers as they evolve?
A Continuous Benchmarking Library
A Common Interface
A Compiler for Mathematical Programming

# Summary


Benchmarking
Architecture-Specific Compilation
Could we benchmark these Solvers as they evolve?
How to pick the right solver?
How to choose the best way to reformulate a problem?
A Continuous Benchmarking Library
A Common Interface
A Compiler for Mathematical Programming

# Summary

**A Continuous Benchmarking Library**

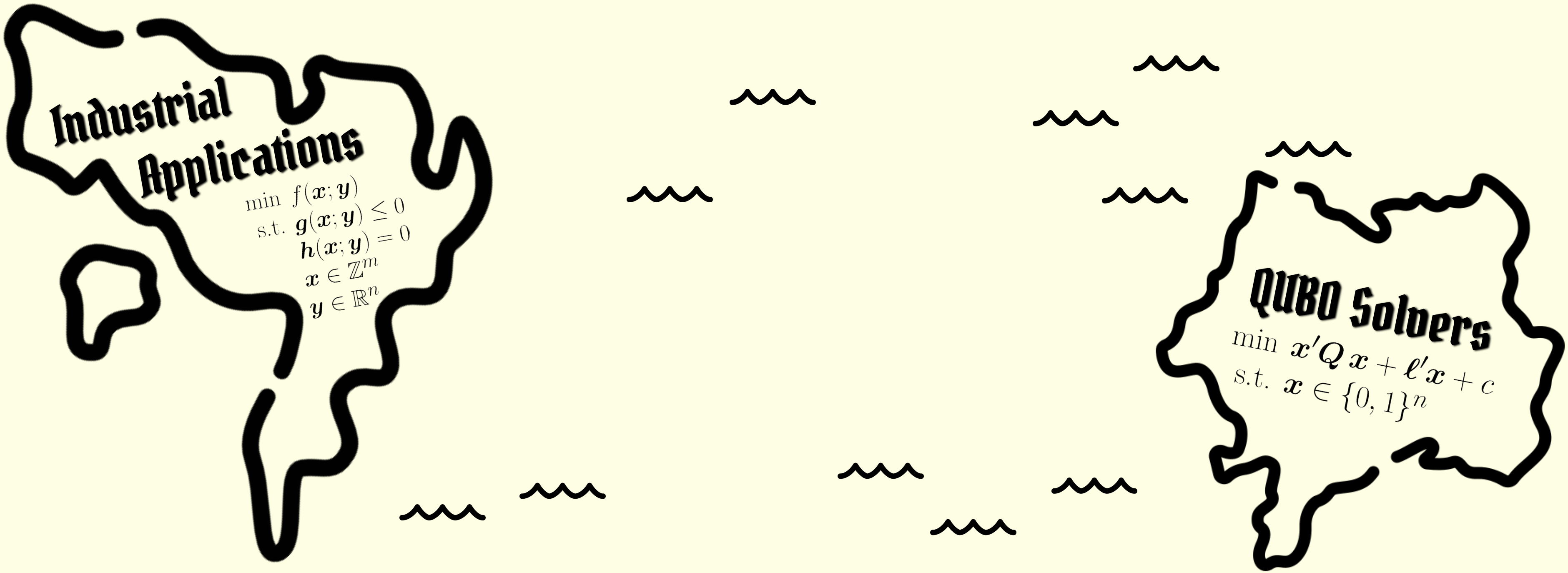
**A Common Interface**

***A Compiler for Mathematical Programming***

A Continuous Benchmarking Library

A Common Interface

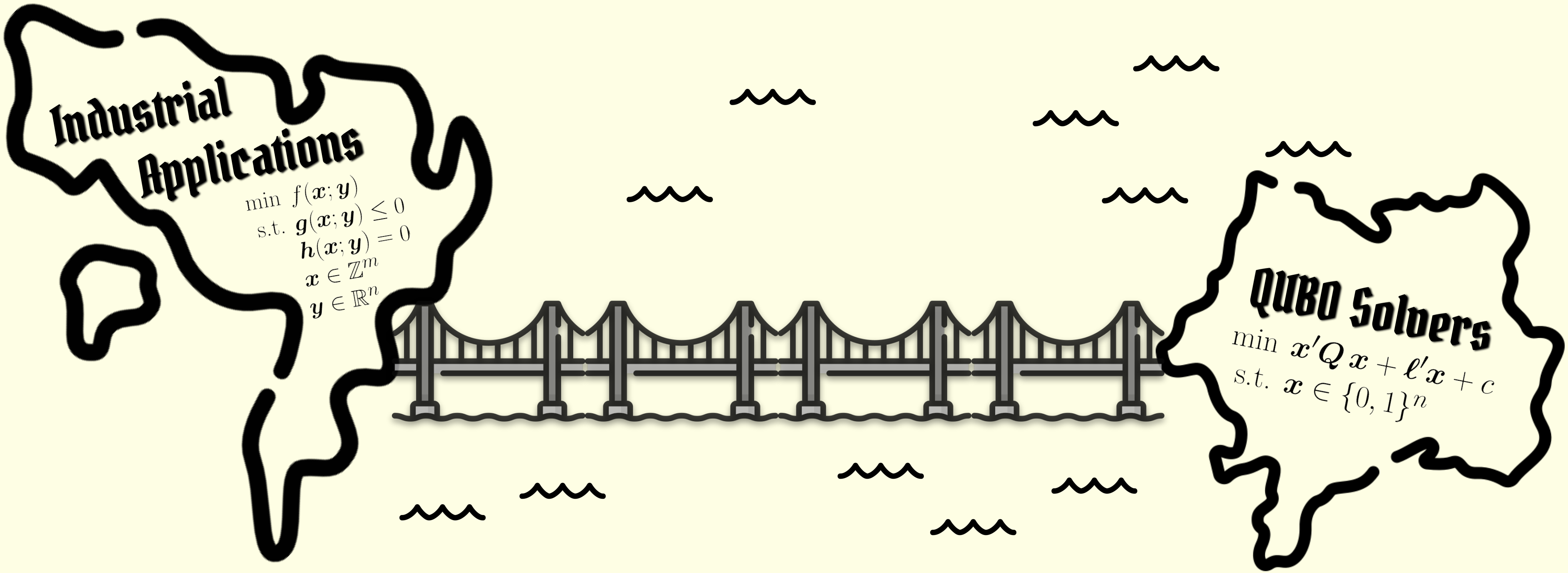
*A Compiler for Mathematical Programming*



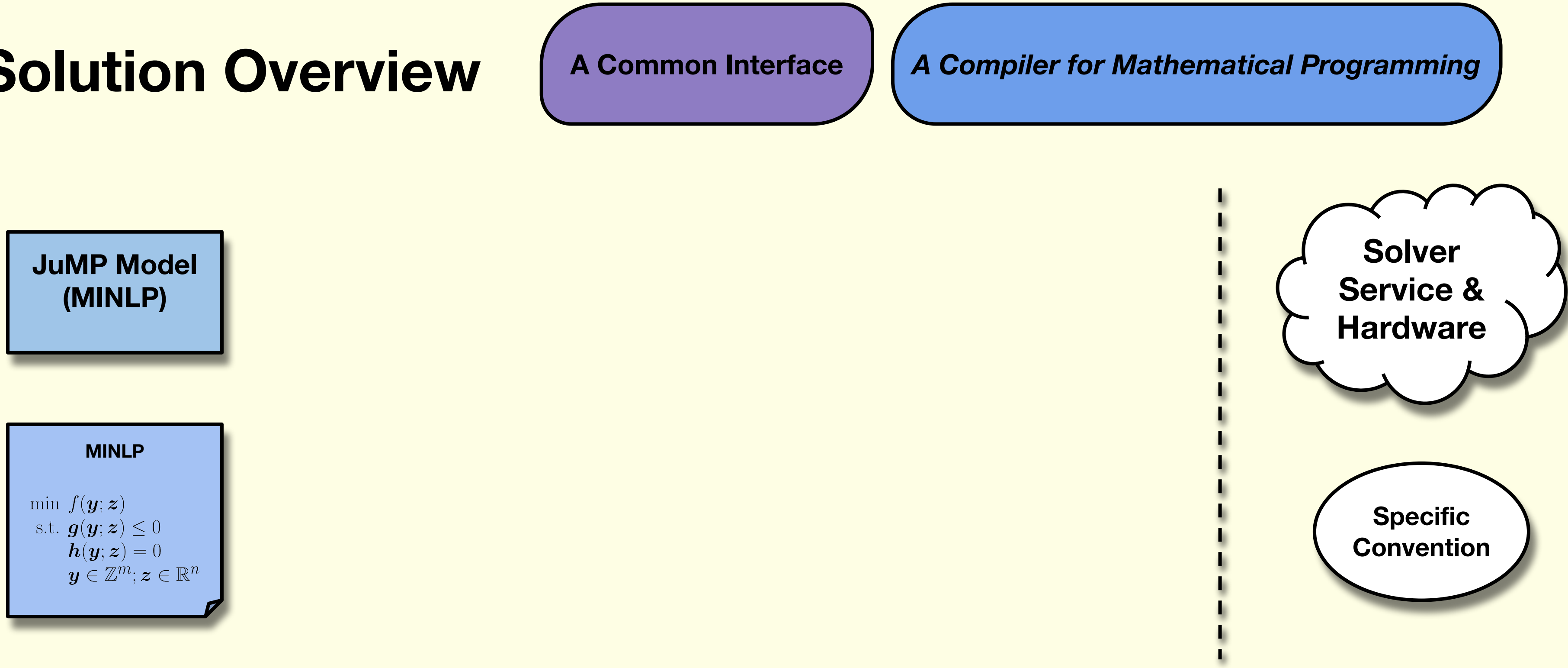
A Continuous Benchmarking Library

A Common Interface

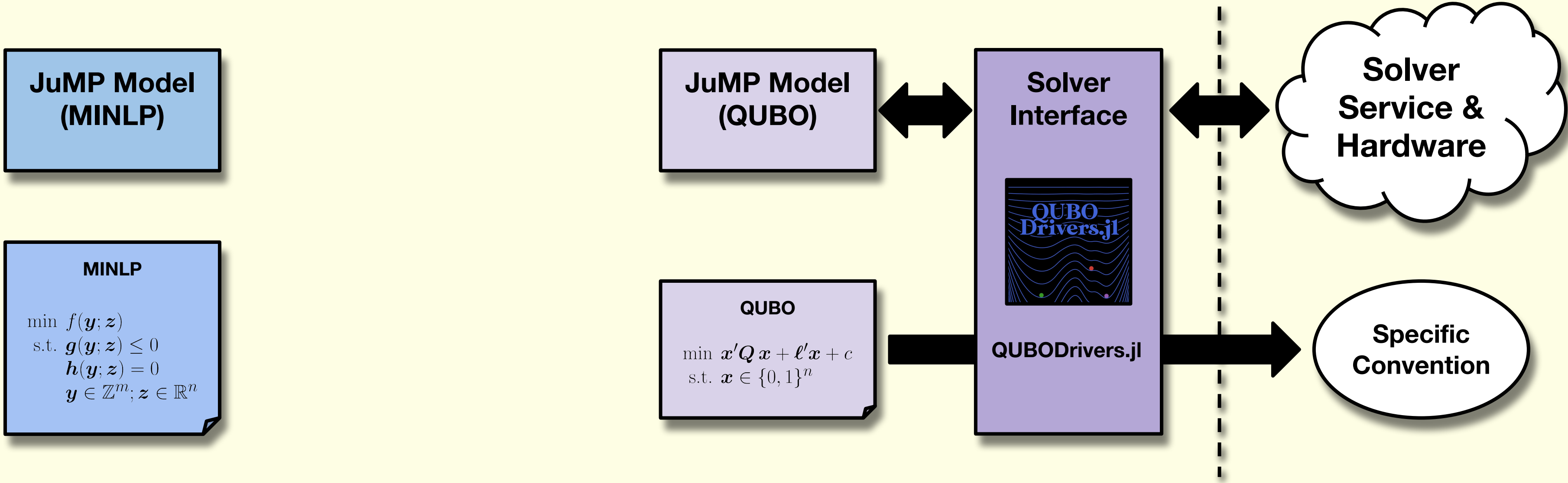
*A Compiler for Mathematical Programming*



# Solution Overview



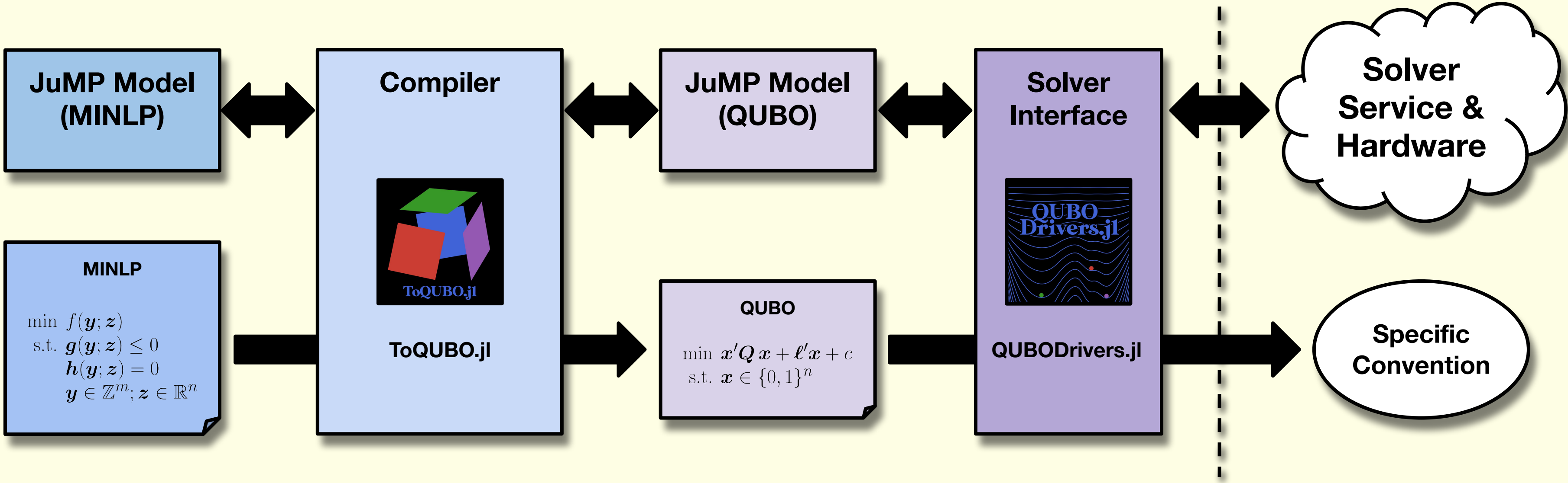
# Solution Overview



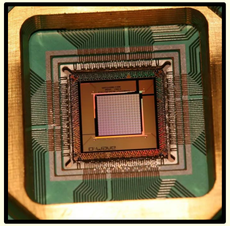
# Solution Overview

A Common Interface

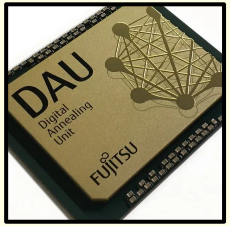
*A Compiler for Mathematical Programming*



# Integrating an heterogeneous Solver Landscape

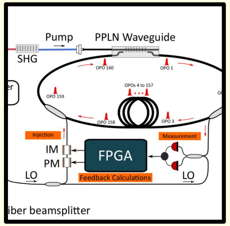


D-Wave

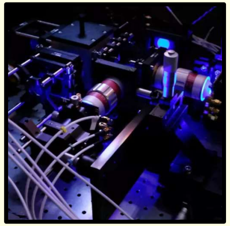



Fujitsu

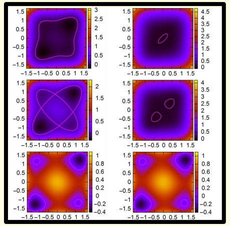
IBM




P. L. McMahon et al., 2016

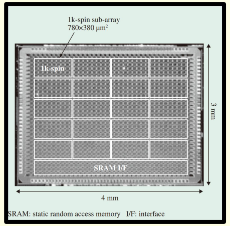


Microsoft Research



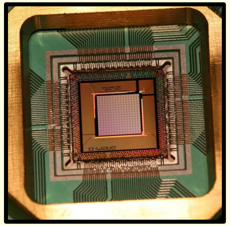
Toshiba, Goto et al., 2019

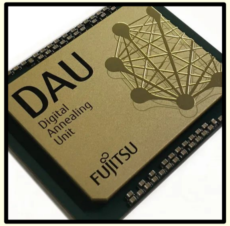
**Quantum Annealing, Digital Annealing, Variational Quantum Eigensolver, Quantum Alternating Optimization Ansatz, Coherent Ising Machine, Analog Iterative Machine, Simulated Bifurcation Machine, CMOS Annealing…**




Hitachi, Yamaoka et al.

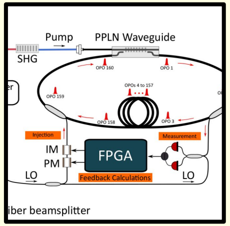
# Integrating an heterogeneous Solver Landscape


D-Wave

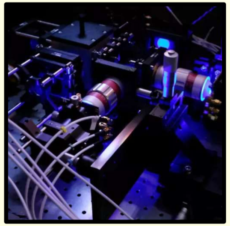


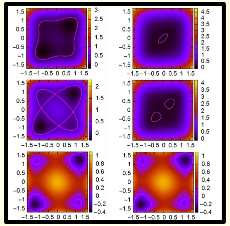
Fujitsu


IBM

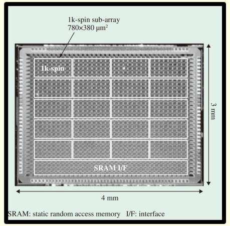


P. L. McMahon et al., 2016


Microsoft Research


Toshiba, Goto et al., 2019

Julia Mathematical Programming



Hitachi, Yamaoka et al.

# A Common Solver Interface

```julia
using JuMP
using QiskitOpt # IBM Qiskit Optimizati

model = Model(QiskitOpt.QAOA.Optimizer)

@variable(model, x[1:n], Bin)
@objective(
  model,
  Min,
  x' * Q * x + ℓ' * x + c
)

optimize!(model)

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

```julia
using JuMP
using DWave # DWave Quantum Annealing

model = Model(DWave.Optimizer)

@variable(model, x[1:n], Bin)
@objective(
  model,
  Min,
  x' * Q * x + ℓ' * x + c
)

optimize!(model)

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

```julia
ing JuMP
ing PySA # NASA Parallel Tempering

del = Model(PySA.Optimizer)

ariable(model, x[1:n], Bin)
bjective(
model,
Min,
x' * Q * x + ℓ' * x + c

timize!(model)

how objective_value(model)
how value.(x)
```

# Testing and Benchmarking Solvers

A Continuous Benchmarking Library

QUBO Lib

# Testing and Benchmarking Solvers

A Continuous Benchmarking Library

| 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

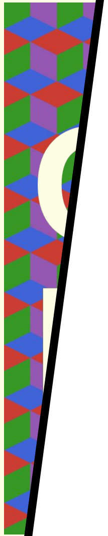
# Testing and Benchmarking Solvers

**A Continuous Benchmarking Library**

```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
```

# Testing and Benchmarking Solvers

A Continuous ...ing Library



{ ISMP 2024 }

**Extensions of Integer Programming**
**Monday, 16:20-17:50**

# A Compiler for Mathematical Programming

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

# A Compiler for Mathematical Programming

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

**VARIABLE ENCODING**

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

# A Compiler for Mathematical Programming

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

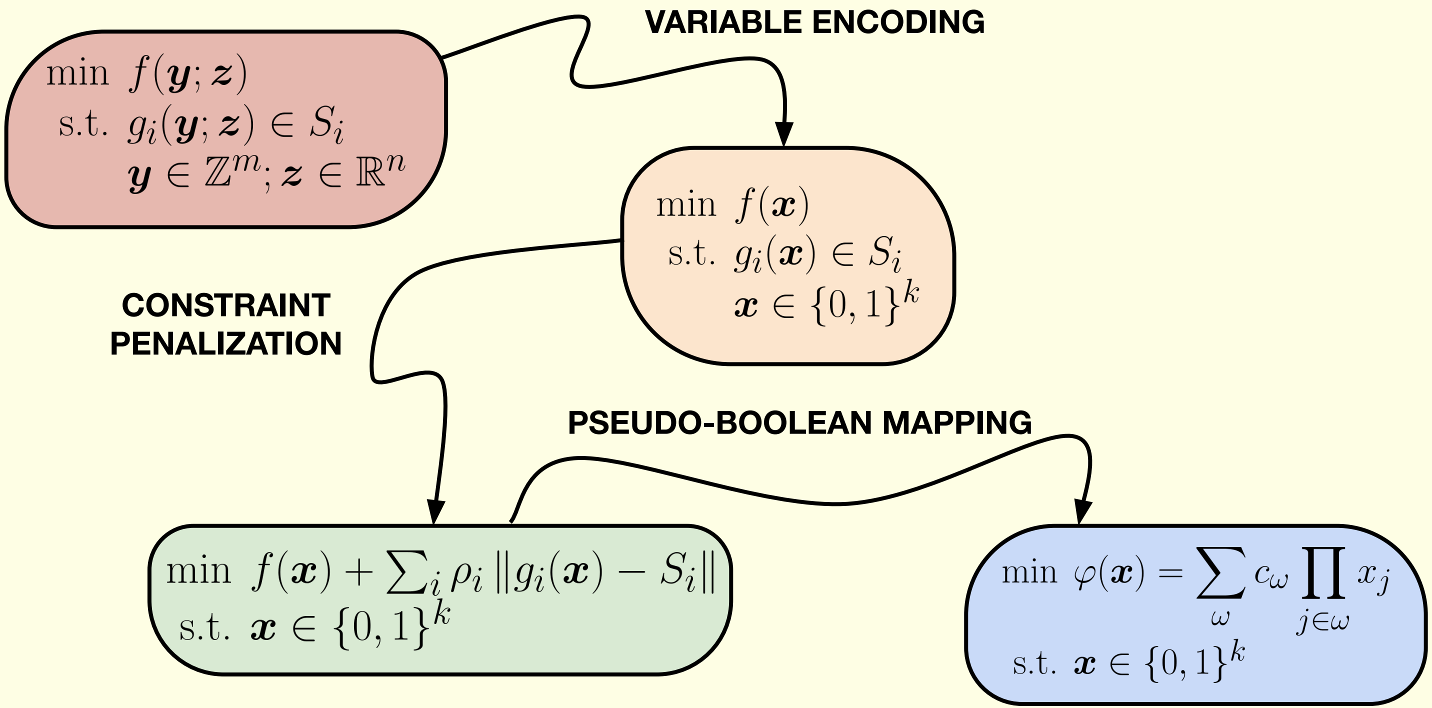
**VARIABLE ENCODING**

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

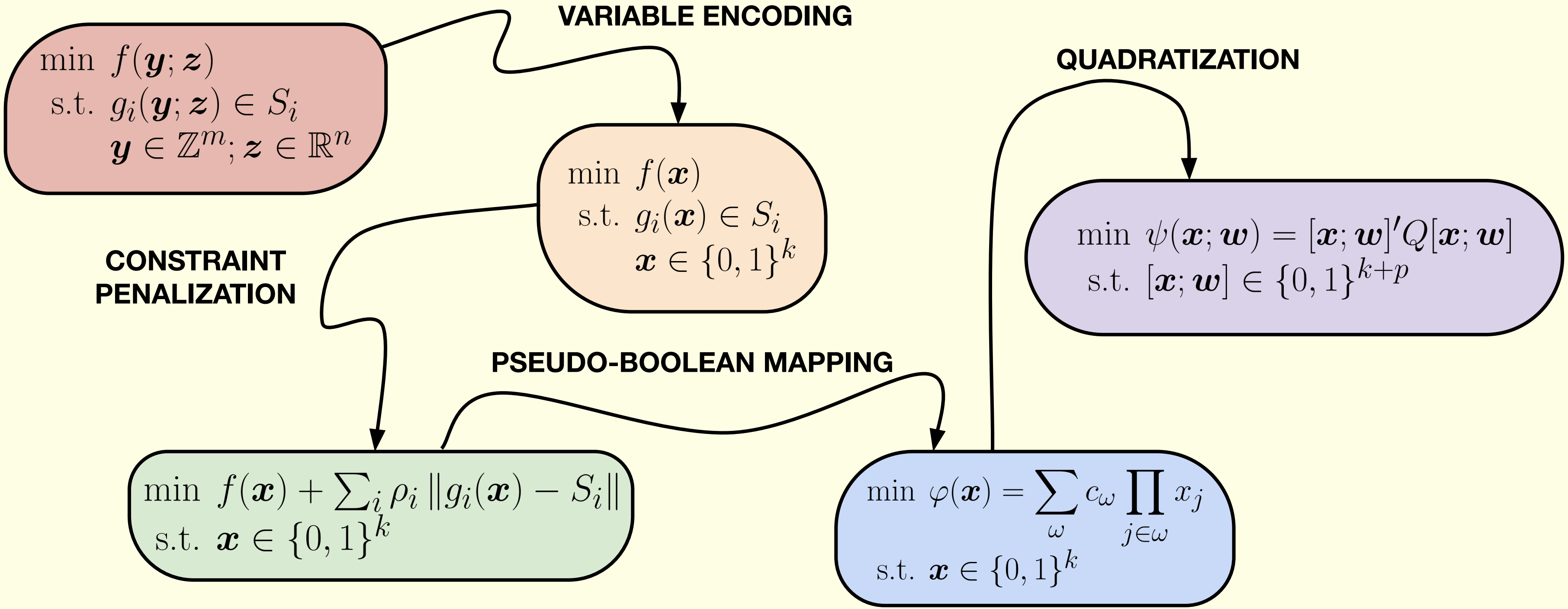
**CONSTRAINT PENALIZATION**

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

# A Compiler for Mathematical Programming



# A Compiler for Mathematical Programming



# A Compiler for Mathematical Programming

C, C++, Julia, Rust…

AMPL, JuMP, Pyomo…

# A Compiler for Mathematical Programming

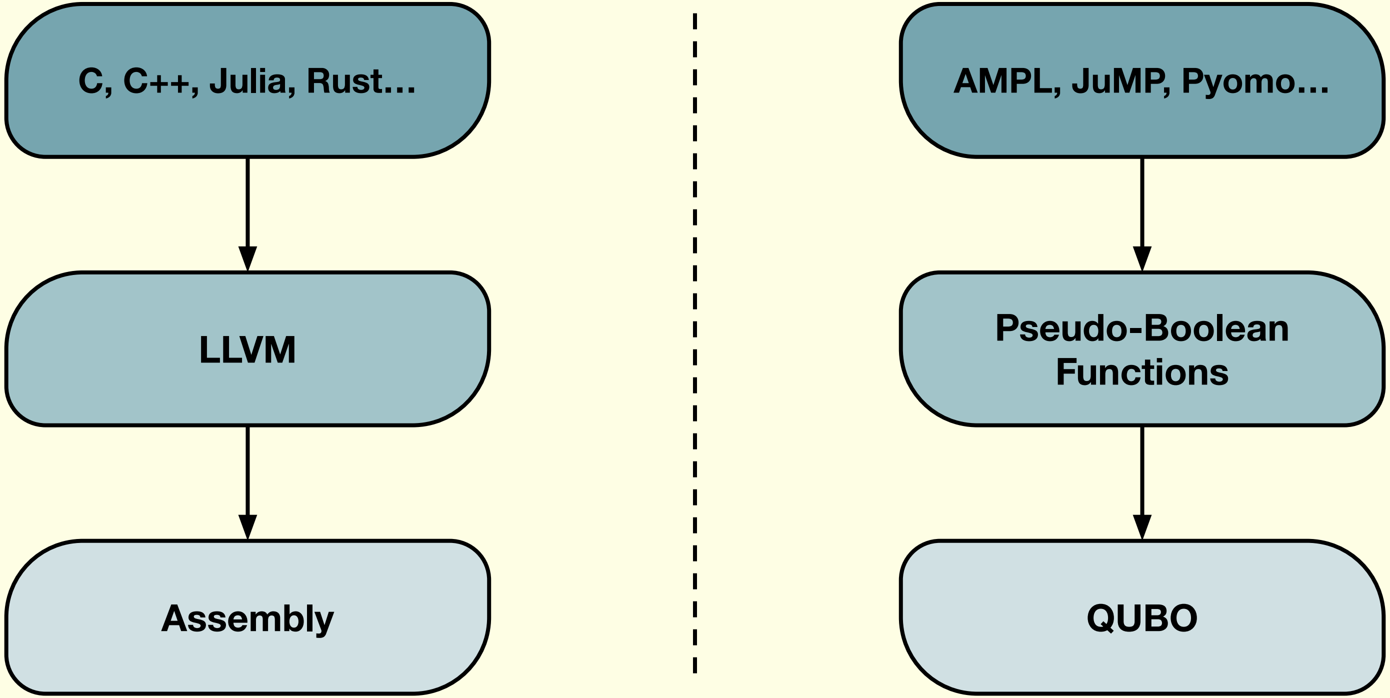
C, C++, Julia, Rust…

Assembly

AMPL, JuMP, Pyomo…

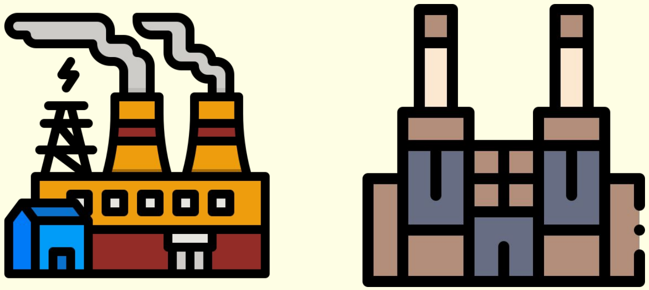
QUBO

# A Compiler for Mathematical Programming

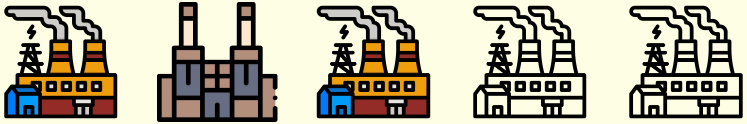


# *EXAMPLE*

# Compilation use case: Generation capacity expansion

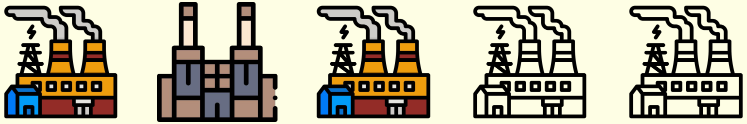


# Compilation use case: Generation capacity expansion



$$
\begin{aligned}
\min_{\mathbf{g},\mathbf{u},\mathbf{x}} \sum_t \mathbf{c}'\mathbf{g}^{(t)} + \mathbf{i}'\mathbf{x} & \\
\text{s.a.} \sum_j g_j^{(t)} = d^{(t)} & \qquad \forall t \\
g_j^{(t)} \leq u_j^{(t)} G_j^{(\max)} & \qquad \forall j, t \\
u_j^{(t)} \leq x_j & \qquad \forall j, t \\
g_j^{(t)} \in [0, G_j^{(\max)}] & \qquad \forall j, t \\
u_j^{(t)} \in \{0, 1\} & \qquad \forall j, t \\
x_j \in \{0, 1\} & \qquad \forall j
\end{aligned}
$$

# Compilation use case: 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

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

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

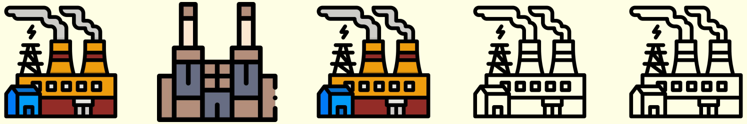
$$g_j^{(t)} \in [0, G_j^{(\max)}] \qquad \forall j, t$$

OPERATION

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

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

# Compilation use case: 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

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

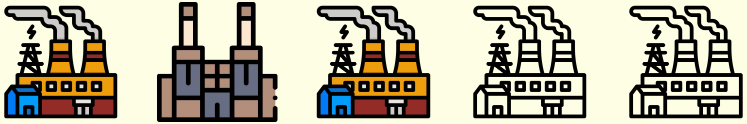
$$g_j^{(t)} \in [0, G_j^{(\max)}] \qquad \forall j, t$$

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

UNIT COMMITMENT

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

# Compilation use case: Generation capacity expansion



INVESTMENT

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

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

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

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

INVESTMENT

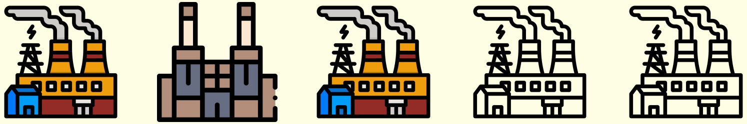
$$g_j^{(t)} \in [0, G_j^{(\max)}] \qquad \forall j, t$$

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

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

INVESTMENT

# Compilation use case: 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

# Compilation use case: 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

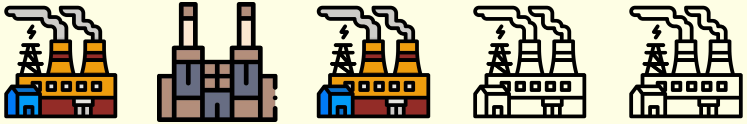
$$u_j^{(t)} \in \{0, 1\} \qquad \forall j, t$$

UNIT COMMITMENT

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

INVESTMENT

**REFORMULATION** →



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

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

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

# Compilation use case: 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

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

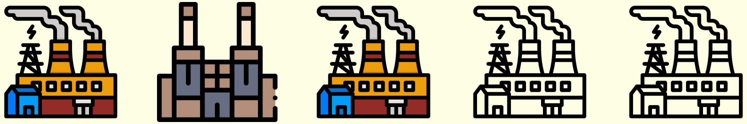
OPERATION

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

UNIT COMMITMENT

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

INVESTMENT



```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)
```

# Compilation use case: 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

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

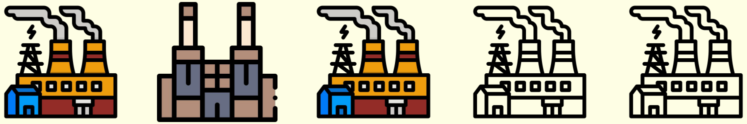
OPERATION

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

UNIT COMMITMENT

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

INVESTMENT



```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)
```

# Compilation use case: Generation capacity expansion

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

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

**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**

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

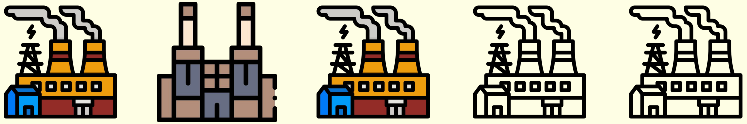
**OPERATION**

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

**UNIT COMMITMENT**

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

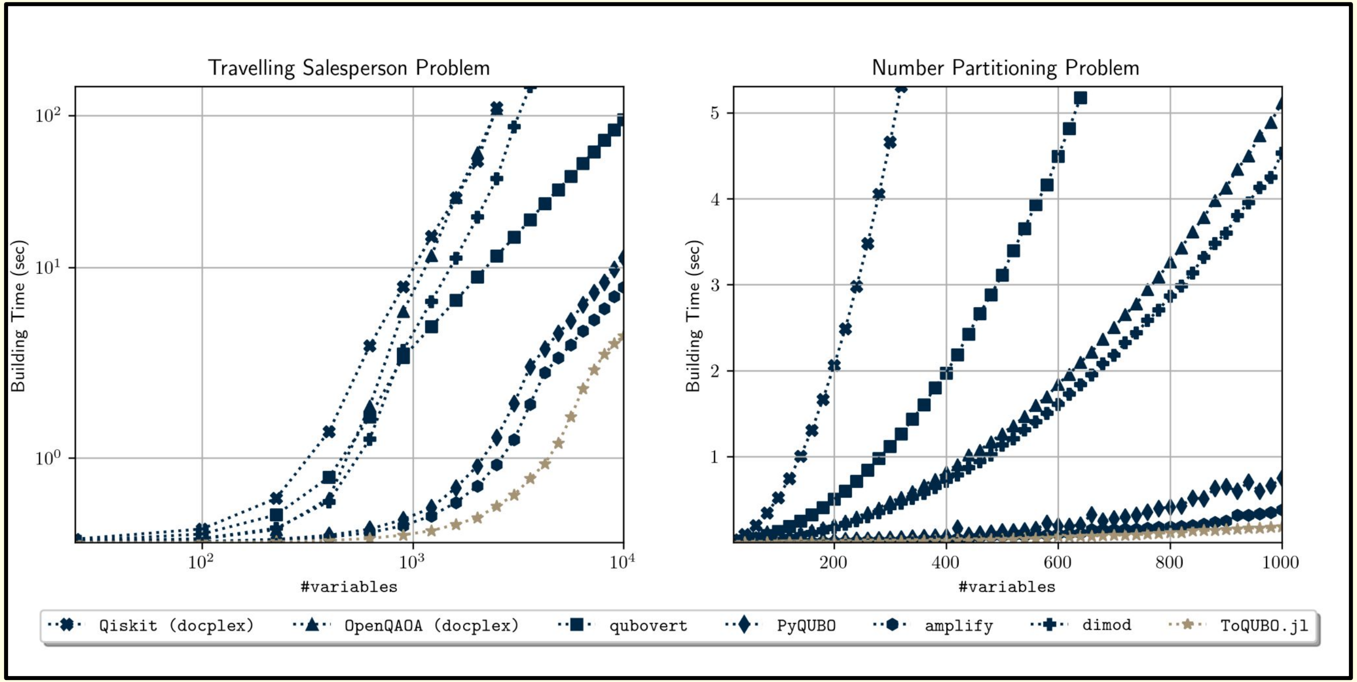
**INVESTMENT**



```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)
```

# RESULTS

# Reformulation Performance



# Qualitative Analysis

**CITATION ALERT**

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.
† 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] | ✗ | ✗ | ✗ | ✗ | ✓* | ✗ | ✗ | ✓ | ✓ |
| 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

# Qualitative Analysis

**CITATION ALERT**

TABLE I: Comparing the support provided by of proposed framework and existing libra… of quantum optimization.

✓ (green) indicates that the corresponding action is performed automatically.

✓ (orange) 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.

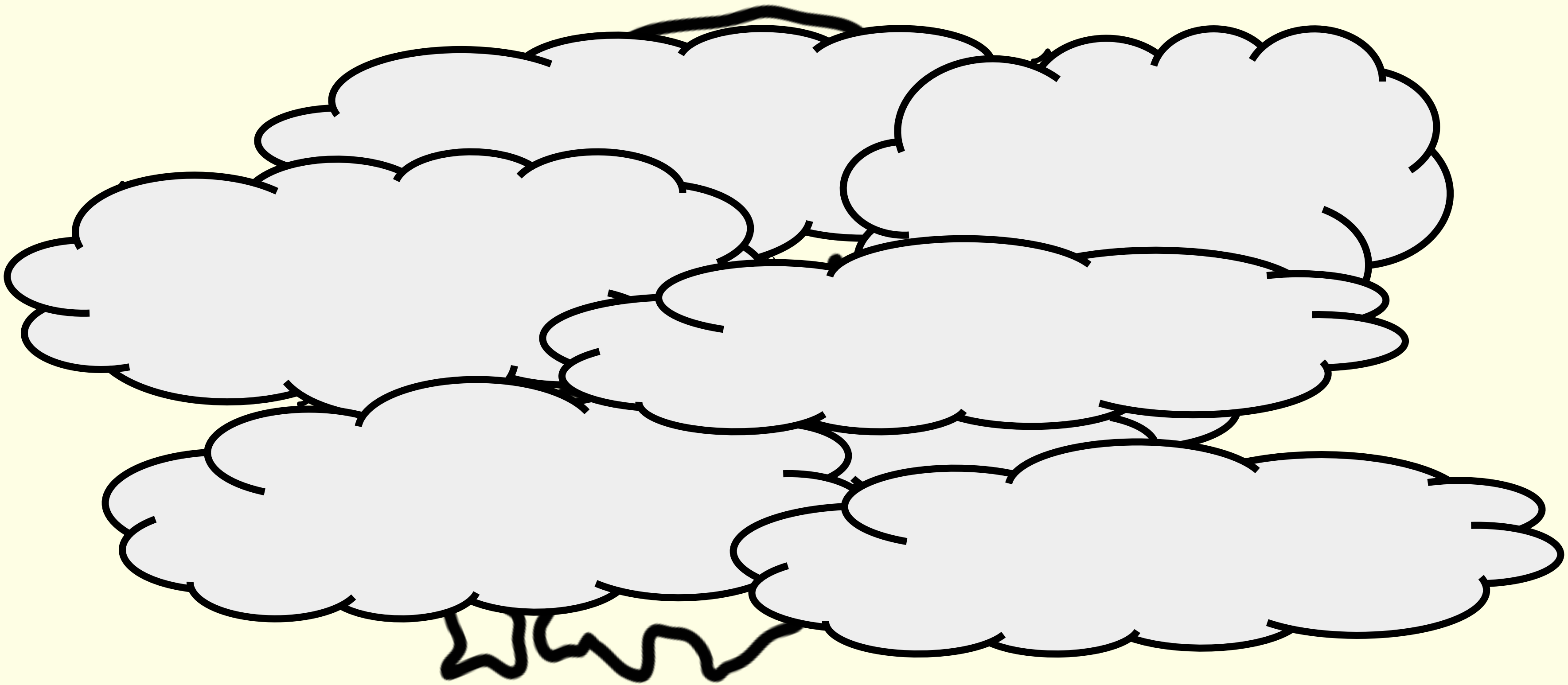
† indicates that the polynomial reduction is implemented by exploiting the corresponding qubovert function.

| 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] | ✓ (orange) | ✓ (orange) | ✓ (orange) | ✓ (orange) | ✓* (orange) | ✗ | ✗ | ✓ | ✓+ |
| | Unitary [39] | ✓ (orange) | ✓ (orange) | ✗ | ✗ | ✓* (orange) | ✗ | ✗ | ✓ | ✓ |
| | Dictionary [39] | ✓ (orange) | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✓ |
| | Domain-Wall [40] | ✓ (orange) | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✓ |
| | Bounded-Coeff [41] | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✓ |
| | Arithmetic [42] | ✗ | ✗ | ✗ | ✗ | ✓* (orange) | ✗ | ✗ | ✓ | ✓ |
| … Functions | Equality [22] [21] | ✗ | ✓ (orange) | ✓ (orange) | ✓ (orange) | ✓ (orange) | ✓ (orange) | ✗ | ✓ | ✓ |
| | Inequality [22] | ✗ | ✓ (orange) | ✓ (orange) | ✓ (orange) | ✓ (orange) | ✓ (orange) | ✗ | ✓ | ✓ |
| | Boolean [22] | ✓ (orange) | ✓ (orange) | ✓ (orange) | ✗ | ✓ (orange) | ✗ | ✗ | ✗ | ✓ |
| Penalty Weight | UB positive [43] | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ |
| | MQC [43] | ✗ | ✗ | ✓ (orange) | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ |
| | VLM [44] | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✗ | ✓ |
| | MOMC [43] | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ |
| | MOC [43] | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ |
| | UB Naive [45], [46] | ✗ | ✓ (orange) | ✗ | ✓ | ✗ | ✗ | ✓ | ✗ | ✓ |
| | UB posiform [45], [46] | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✗ | ✓ |
| Polynomial Reduction | | ✓ (orange) | ✓ (orange) | ✓ (orange) | ✗ | ✓ (orange) | ✗ | ✓ | ✓ | ✓† |
| Solvers | Dwave QA | ✓ | ✗ | ✓ | ✗ | ✓ | ✗ | ✓ | ✓ | ✓ |
| | QAOA | ✗ | ✗ | ✗ | ✓ | ✗ | ✓ | ✗ | ✓ | ✓ |
| | VQE | ✗ | ✗ | ✗ | ✓ | ✗ | ✓ | ✗ | ✓ | ✓ |
| | GAS | ✗ | ✗ | ✗ | ✓ | ✗ | ✗ | ✗ | ✗ | ✓ |
| | SA | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ | ✓ |
| Solution Decoding | | ✓ (orange) | ✓ (orange) | ✓ (orange) | ✓ (orange) | ✓ (orange) | ✓ (orange) | ✓ | ✓ | ✓ |
| Check Constraints | | ✓ (orange) | ✓ (orange) | ✓ (orange) | ✗ | ✓ (orange) | ✗ | ✓ | ✓ | ✓ |
| Penalty Update | Sequential [47] | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ |
| | Scaled [47] | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ |
| | Binary search [47] | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ |

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

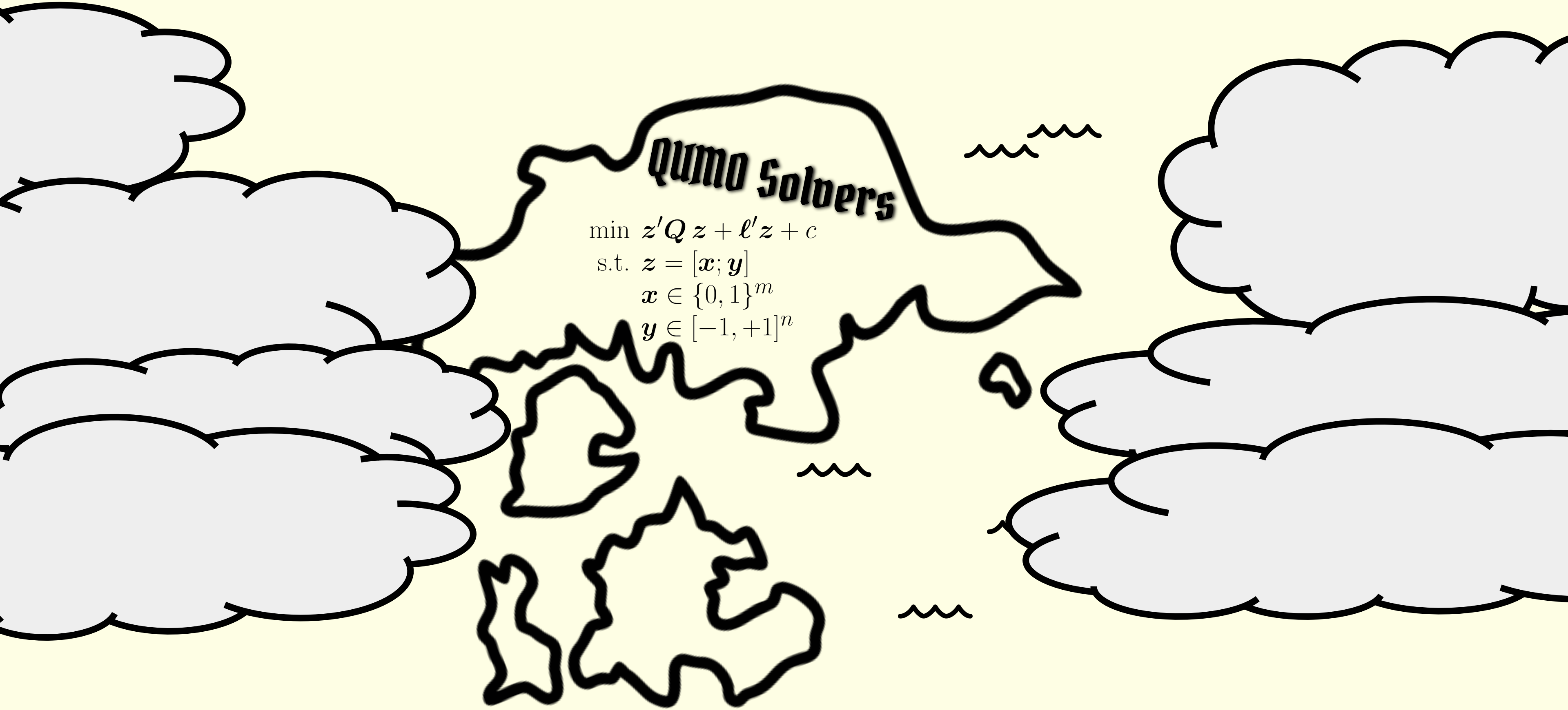
# What's new?

- Additional attributes to control reformulation
  - e.g. `ConstraintPenaltyHint`, `VariableEncodingMethod`
- Reformulation Callbacks
- Architecture-based dispatch
- Disjuctive Programming (`DisjunctiveToQUBO.jl`)



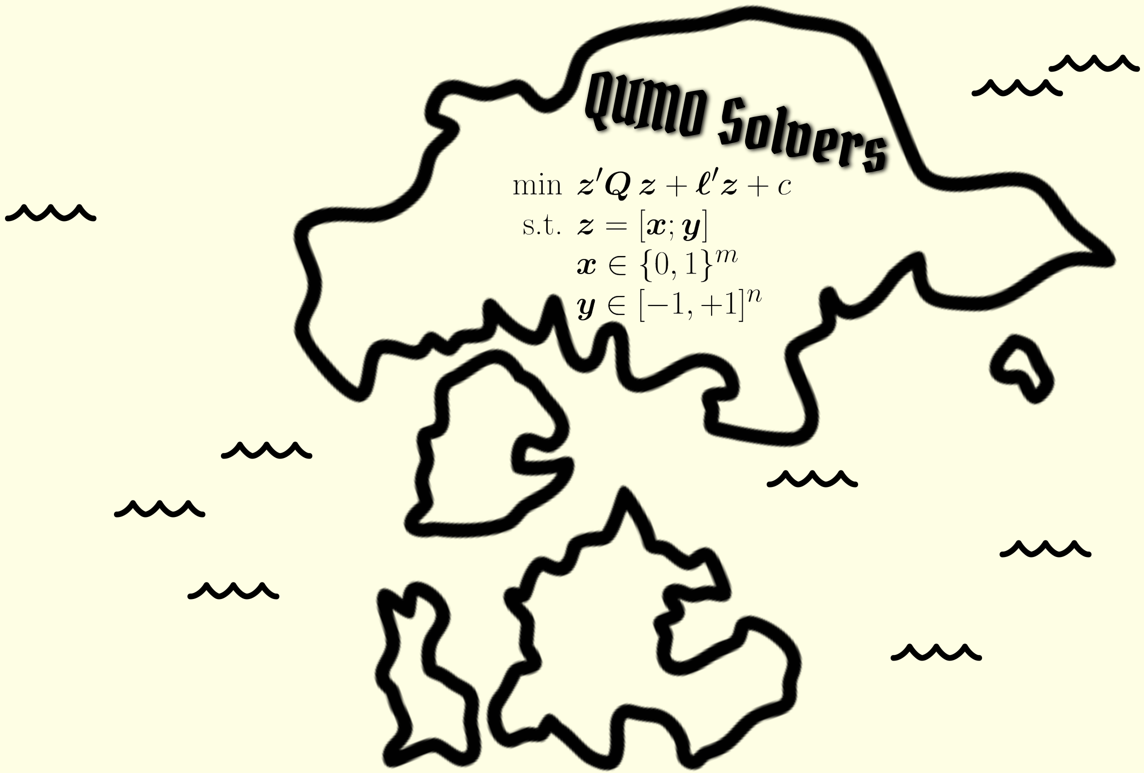
# QUMO Solvers

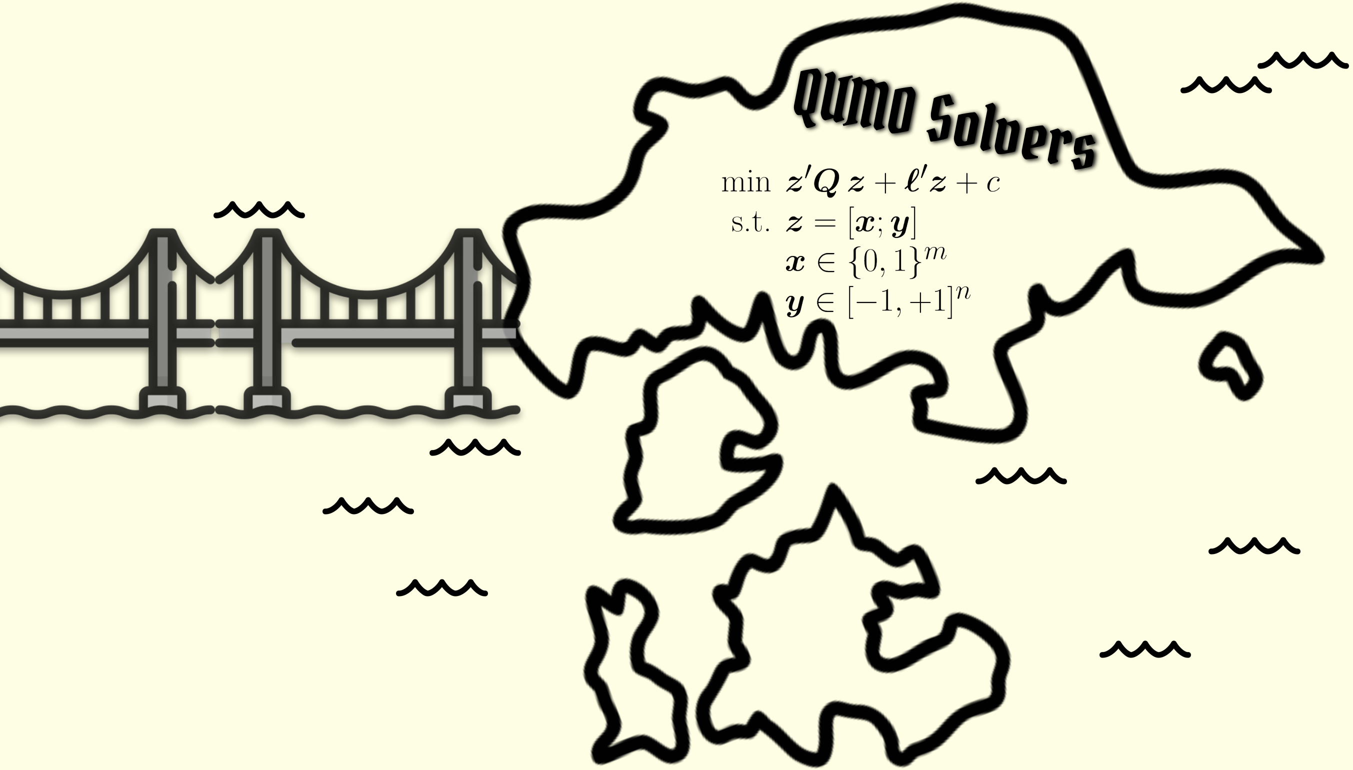
$$
\begin{aligned}
\min \quad & \boldsymbol{z}'\boldsymbol{Q}\,\boldsymbol{z} + \boldsymbol{\ell}'\boldsymbol{z} + c \\
\text{s.t.} \quad & \boldsymbol{z} = [\boldsymbol{x}; \boldsymbol{y}] \\
& \boldsymbol{x} \in \{0, 1\}^m \\
& \boldsymbol{y} \in [-1, +1]^n
\end{aligned}
$$



# QUMO Solvers

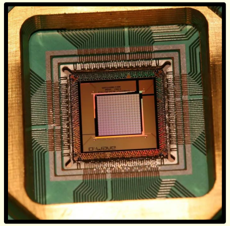
$$
\begin{aligned}
\min \quad & \boldsymbol{z}'\boldsymbol{Q}\,\boldsymbol{z} + \boldsymbol{\ell}'\boldsymbol{z} + c \\
\text{s.t.} \quad & \boldsymbol{z} = [\boldsymbol{x}; \boldsymbol{y}] \\
& \boldsymbol{x} \in \{0, 1\}^m \\
& \boldsymbol{y} \in [-1, +1]^n
\end{aligned}
$$



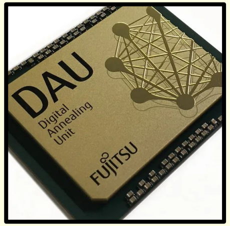



$$
\begin{aligned}
\min\ & \boldsymbol{z}'\boldsymbol{Q}\,\boldsymbol{z} + \boldsymbol{\ell}'\boldsymbol{z} + c \\
\text{s.t.}\ & \boldsymbol{z} = [\boldsymbol{x}; \boldsymbol{y}] \\
& \boldsymbol{x} \in \{0, 1\}^m \\
& \boldsymbol{y} \in [-1, +1]^n
\end{aligned}
$$

# Integrating an heterogeneous Solver Landscape

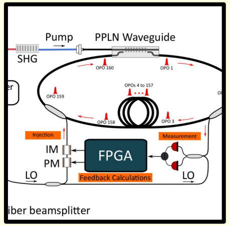


D-Wave

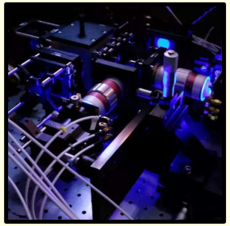



Fujitsu

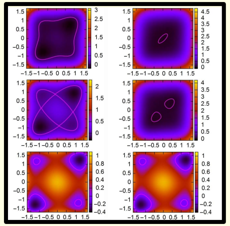
IBM




P. L. McMahon et al., 2016

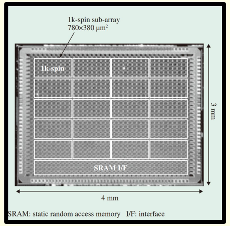


Microsoft Research



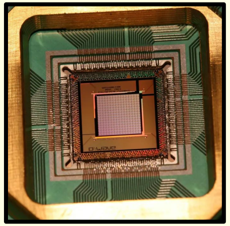
Toshiba, Goto et al., 2019

Julia Mathematical Programming

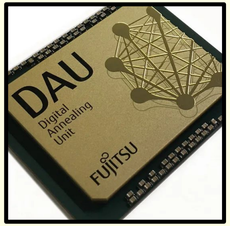



Hitachi, Yamaoka et al.

# Integrating an heterogeneous Solver Landscape

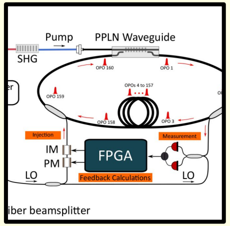


D-Wave

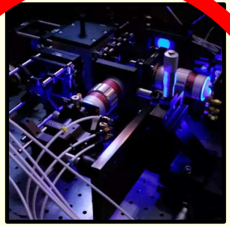



Fujitsu

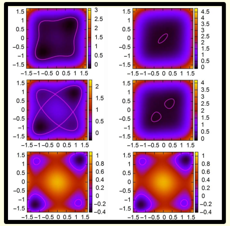
IBM



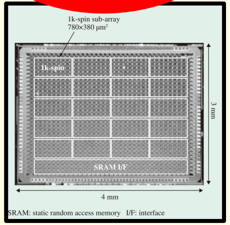

P. L. McMahon et al., 2016



Microsoft Research



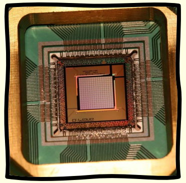
Toshiba, Goto et al., 2019




Hitachi, Yamaoka et al.

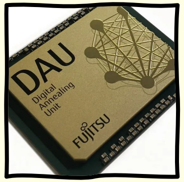
JuMP

Julia Mathematical Programming

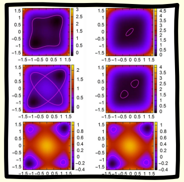


*D-Wave*

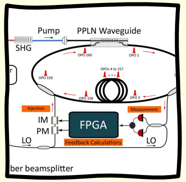
*IBM*



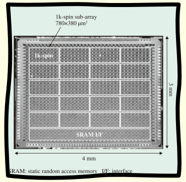

*Fujitsu*



*Toshiba, Goto et al., 2019*

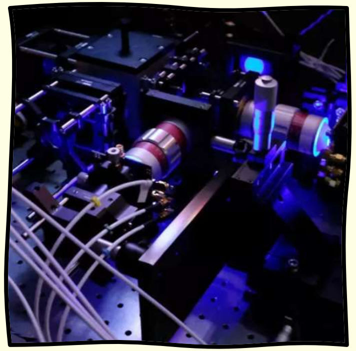



*P. L. McMahon et al., 2016*



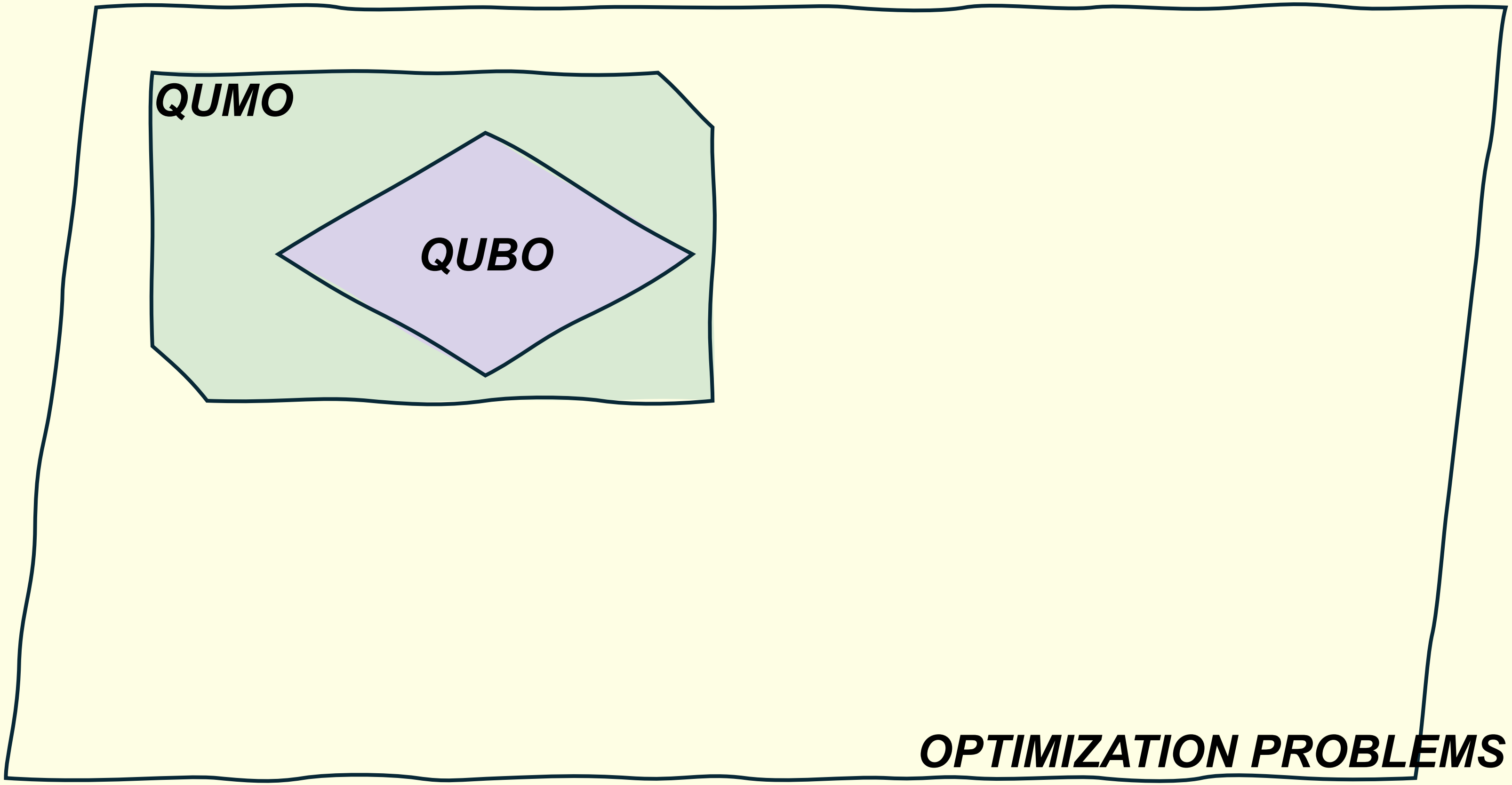

*Hitachi, Yamaoka et al.*

$$(\text{QUBO}) \quad \min_{\mathbf{x}} \; \mathbf{x}'\mathbf{Q}\,\mathbf{x} + \boldsymbol{\ell}'\mathbf{x} + c \quad \text{s.t.} \; \mathbf{x} \in \{0,1\}^n$$



***Microsoft Research AOC***

$$(\text{QUMO}) \quad \min_{\mathbf{x}} \; \mathbf{z}'\mathbf{Q}\,\mathbf{z} + \boldsymbol{\ell}'\mathbf{z} + c \quad \text{s.t.} \; \mathbf{z} = [\mathbf{x};\mathbf{y}], \; \mathbf{x} \in \{0,1\}^m, \; \mathbf{y} \in [-1,+1]^n$$


QUMO
QUBO
OPTIMIZATION PROBLEMS

# BREAKING

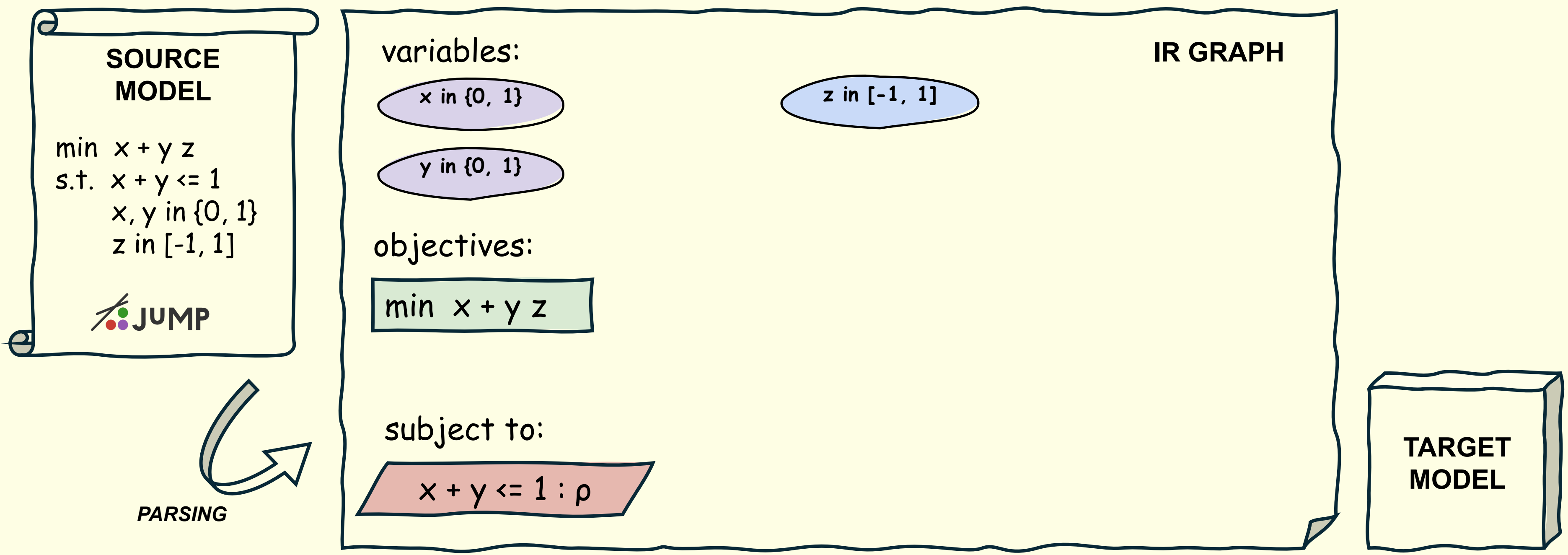
OPTIMIZATION PROBLEMS

**BREAKING**

**PSEUDO-BOOLEAN THEORY IS OVER**

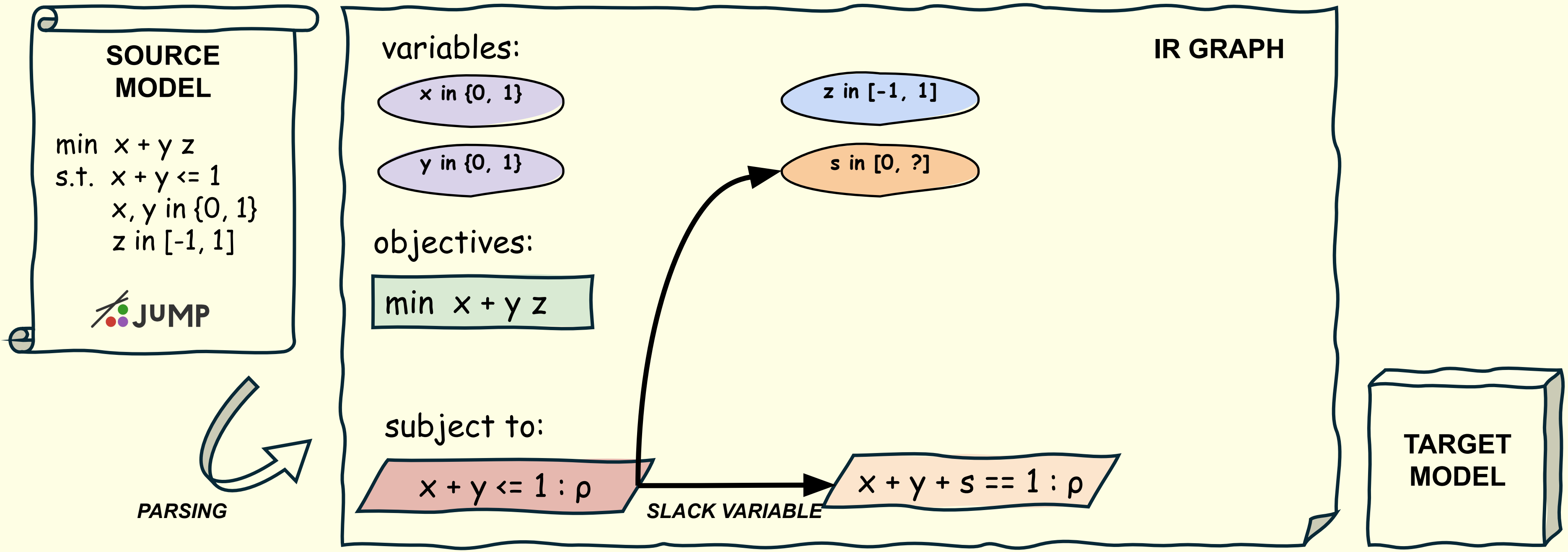
*...TIMIZATI...N PROBLEMS*

# BRIDGING THE GAP



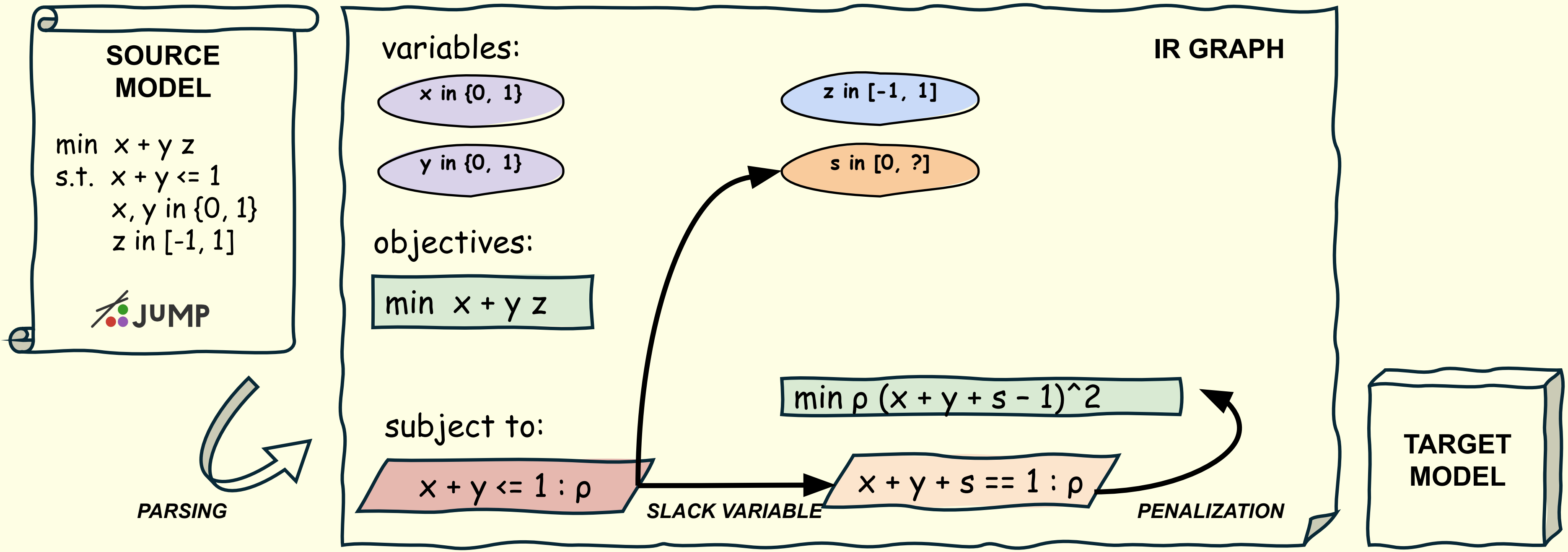

**MOI Bridges:** [2002.03447] MathOptInterface: a data structure for mathematical optimization problems (arxiv.org)

# BRIDGING THE GAP



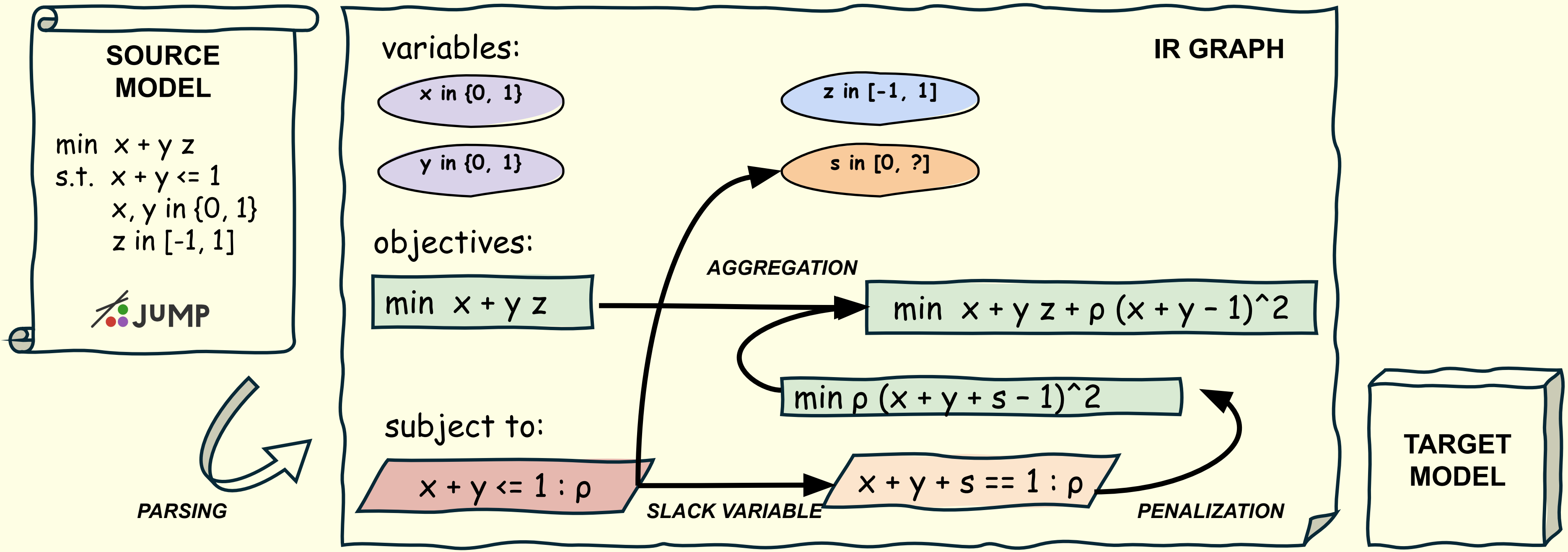

**MOI Bridges:** [2002.03447] MathOptInterface: a data structure for mathematical optimization problems (arxiv.org)

# BRIDGING THE GAP



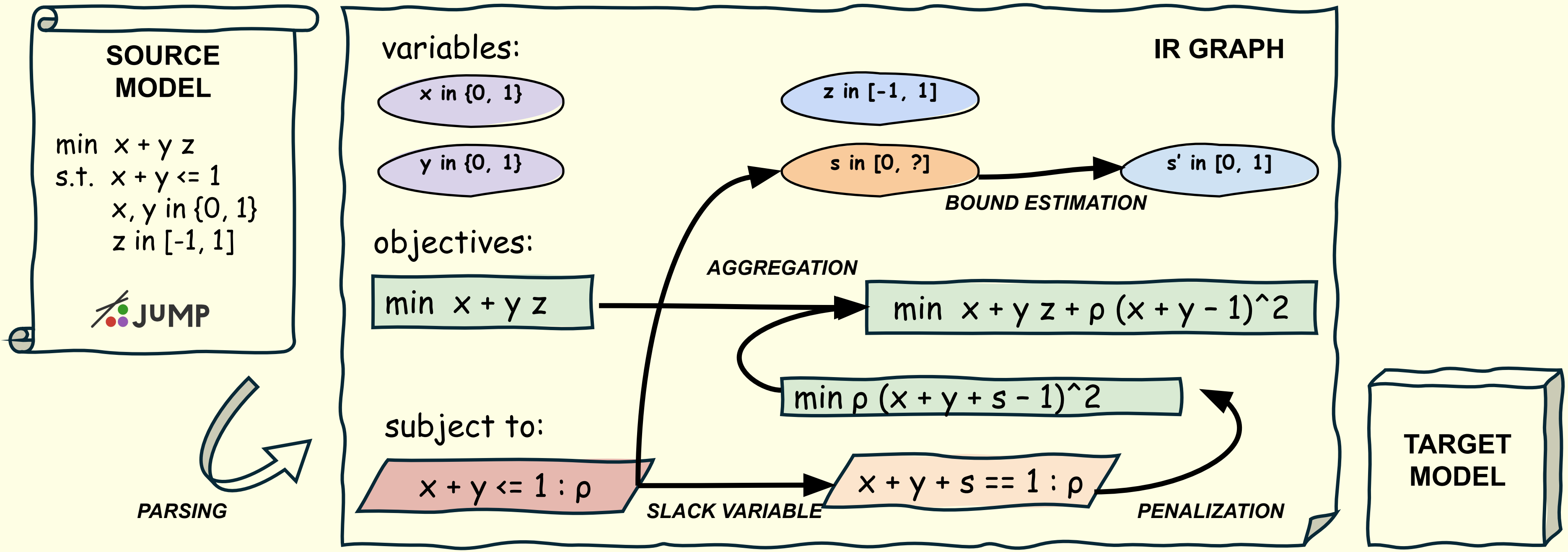

MOI Bridges: [2002.03447] MathOptInterface: a data structure for mathematical optimization problems (arxiv.org)

# BRIDGING THE GAP



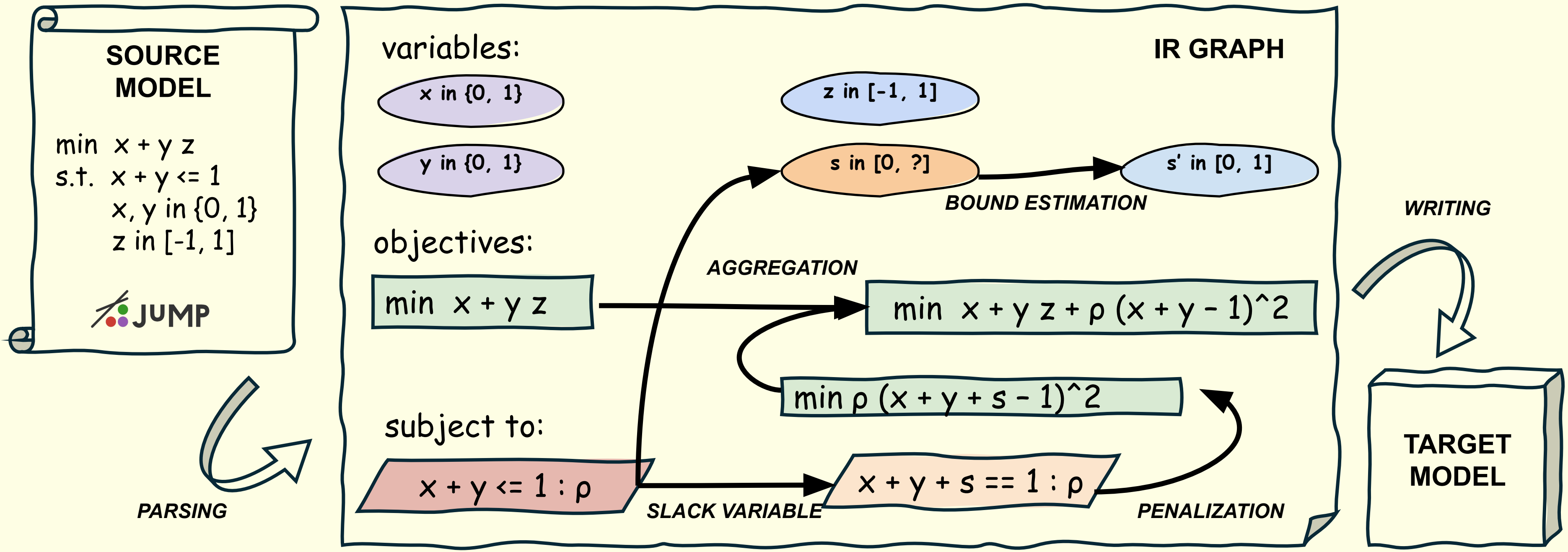

**MOI Bridges:** [2002.03447] MathOptInterface: a data structure for mathematical optimization problems (arxiv.org)

# BRIDGING THE GAP




MOI Bridges: [2002.03447] MathOptInterface: a data structure for mathematical optimization problems (arxiv.org)

# BRIDGING THE GAP




**MOI Bridges:** [2002.03447] MathOptInterface: a data structure for mathematical optimization problems (arxiv.org)

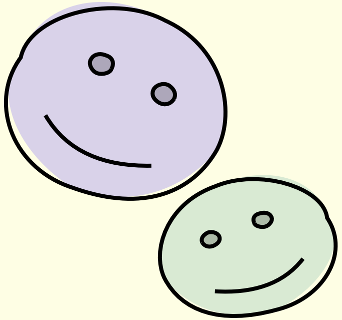
# What about MOI Bridges?

- In fact, it is good to use as many MOI Bridges as possible
- Some of the operations are too destructive
  - e.g. Penalization, Encoding Continuous Variables
- Solver-dependent reformulation path finding
- Fine-grained control over reformulation steps

# Next Steps

- ❏ Generalize the reformulation algorithm
- ❏ Expand constraint support
- ❏ Set up continuous benchmarking service
- ❏ Draw insights from benchmarking to guide reformulation

# ~~Wishlist~~ Food for Thought



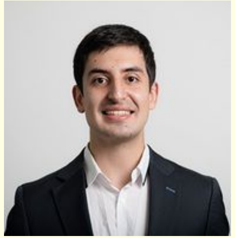
- ❑ Asynchronous `optimize!` call
- ❑ Communication protocol for client-server apps

# Acknowledgements


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

**Joaquim Dias Garcia**
*PSR*

**Nelson Maculan**
*UFRJ*



**David E. Bernal Neira**
*Purdue*

PSR

UFRJ
UNIVERSIDADE FEDERAL
DO RIO DE JANEIRO

PURDUE
UNIVERSITY

USRA
Universities Space Research Association

***THANKS FLATICON.COM FOR THE FIGURES***

# More Acknowledgements

Christos Gkantsidis
Microsoft Research UK

Kirill Kalinin
Microsoft Research UK

Microsoft

***THANKS FLATICON.COM FOR THE FIGURES***

# Thanks!

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

GitHub Repository

arXiv Preprint