ORTools.jl

Bringing Google's Optimization Power to Julia

Ochibobo Warren
linkedin.com/wochibo
Google SWE, Android Growth

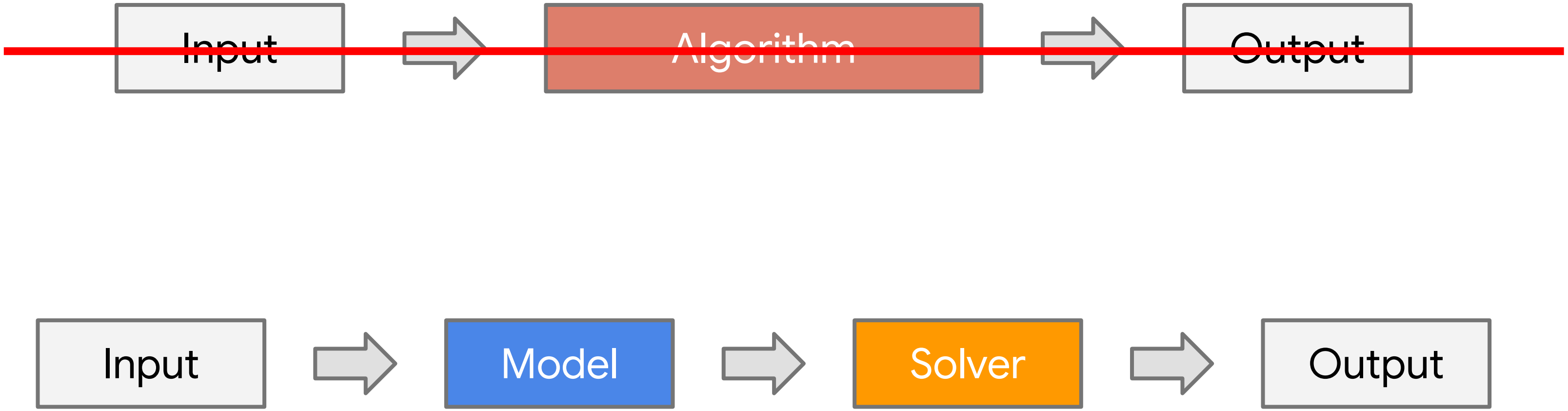
Joint work with:
Thibaut Cuvelier
Google SWE, Operations Research

Agenda

1. **The Big Picture:** What is Mathematical Optimization?
2. **The Engine:** A brief on Google's MathOpt
3. **The Bridge:** The ORTools.jl Package
4. **In Practice:** A Simple Coding Example
5. **Real-World Impact:** Use Cases & Applications
6. **Q&A**

Classic Computer Science vs Operations Research

There are different ways to solve a problem.



What is Mathematical Optimization?

In simple terms, it's the science of **making the best possible decisions** given a set of choices and limitations.

Think of it as finding the "highest point on a mountain" or the "lowest point in a valley" while staying on the marked trails within the shortest time (minimization).

- **Goal:** Find the best outcome (shortest time).
- **Choices:** Variables you can control (paths you can choose to use).
- **Limitations:** Rules or constraints you must follow (staying on marked trails).

Core Components of an Optimization Problem

Every optimization problem has three key parts:

1. **Objective Function:**
   - The value you want to **maximize** or **minimize**.
   - *Examples: Maximize profit, minimize cost, minimize travel time.*
2. **Decision Variables:**
   - The "knobs" you can turn; the choices you need to make.
   - *Examples: How many products to manufacture, which routes to take, where to invest money.*
3. **Constraints:**
   - The rules of the game; the limitations you must respect.
   - *Examples: Budget limits, resource availability, delivery deadlines.*

Model

The model is then fed into a tool called a **solver**, a standalone library that solves a mathematical optimization problem (or at least, it tries).

My Mini-Opt Journey...

Used to work in a logistics startup..

Authored LearnieCP, a constraint programming solver pedagogical purposes. It works with integer variables and deterministic problems.

It's based on the MiniCP paper.

Can be found here.

Enter Google's MathOpt 🛠️

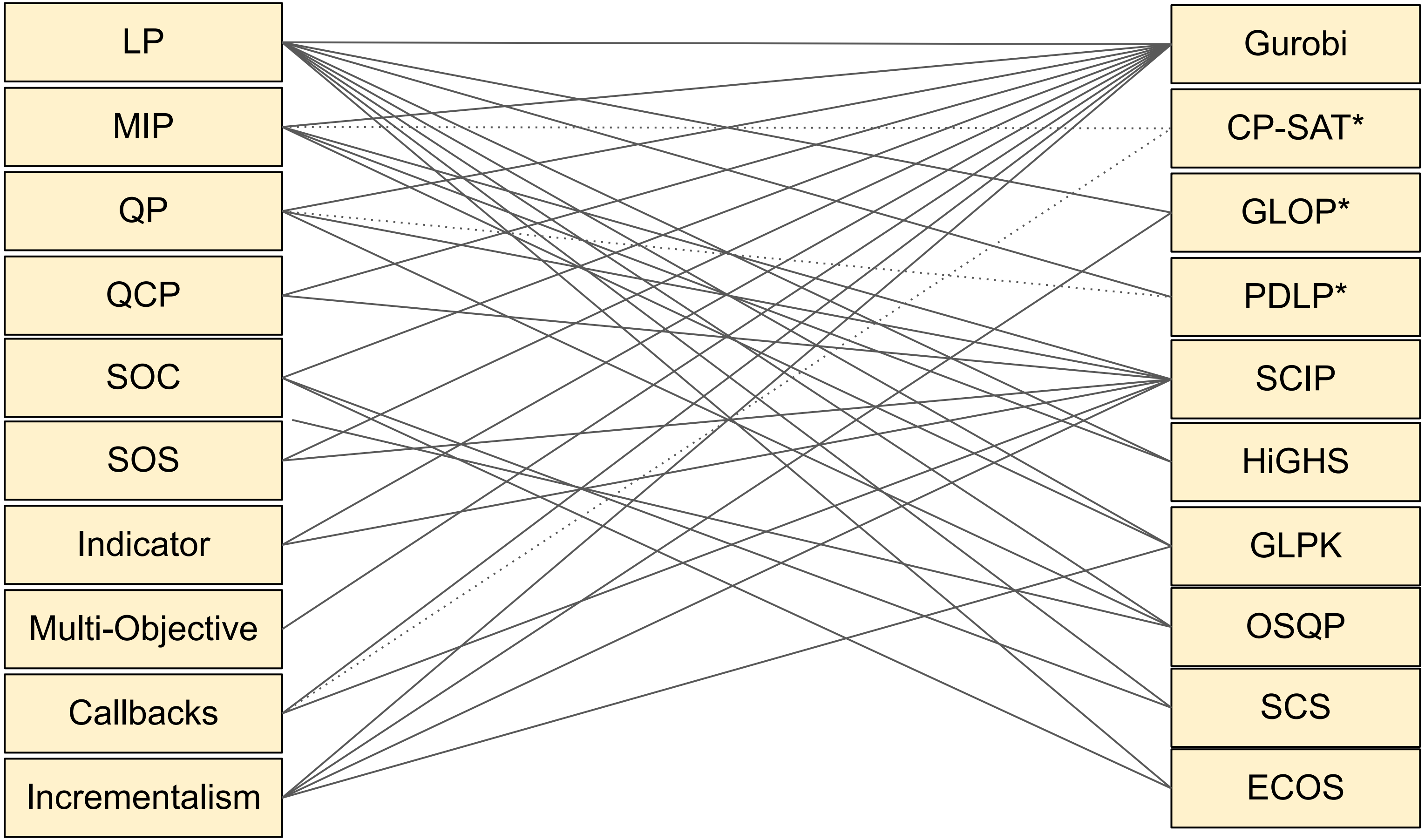
MathOpt is a software library developed by Google for solver-independent optimization modeling that allowing users to switch between different solvers when modelling and solving problems. It provides a generic way of accessing mathematical optimization solvers.

In summary, MathOpt is:

- A feature rich library for solver independent optimization modeling
- Model building in multiple languages: C++, Python, Java & **now Julia!!**
- Execution independent solving: locally, in a subprocess, or remotely
- Built for production use at Google, free for everyone in **Google's OR-Tools**

It exposes a set of solvers; Google owned and third-party solvers, open source & commercially available solvers.

MathOpt Feature/Solver Support



* Made by Google

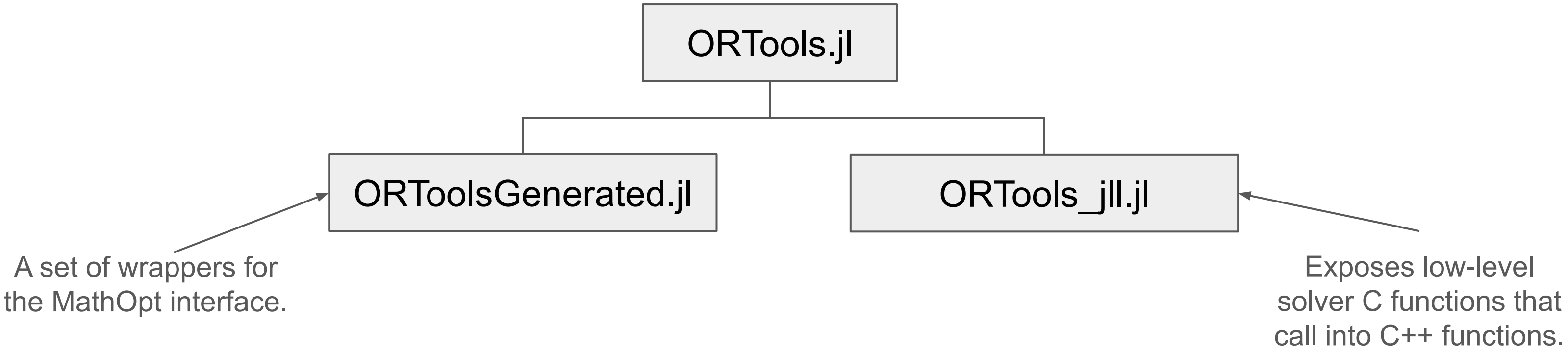
The Julia Connection 🤝

ORTools.jl is the "bridge" that exposes Google's MathOpt interface to the Julia community (written in Julia, of course) through implementing the MathOptInterface; an abstraction layer for mathematical optimization in Julia.

You get the best of both worlds:

- **Julia's** elegant syntax for defining your problem.
- **Google's** world-class solvers to find the solution.

Currently, ORTools.jl has the following architecture:



A Simple Example: The Fertilizer Blend Problem 🌱

A company produces a fertilizer by mixing two raw materials: Material A and Material B. The company wants to produce a blend that meets certain nutritional requirements at the **minimum possible cost**.:

- **Cost:**
  - **Material A** costs **$2 per kilogram.**
  - **Material B** costs **$4 per kilogram.**
- **Nutrient Content:**
  - **Nitrogen:** Material A contains 10% nitrogen, and Material B contains 5% nitrogen. The final blend must contain at least 20 kg of nitrogen.
  - **Phosphate:** Material A contains 6% phosphate, and Material B contains 10% phosphate. The final blend must contain at least 18 kg of phosphate.

How many kilograms of each raw material should be mixed to meet the requirements at minimum cost?

Model representation in LaTeX

$$
\begin{aligned}
&\text{Minimize} \quad && C = 2x + 4y \\
&\text{Subject to:} && \\
& && 0.10x + 0.05y \geq 20 \\
& && 0.06x + 0.10y \geq 18 \\
& && x, y \geq 0
\end{aligned}
$$

ORTools.jl in Action (The Code)

```
# import ORTools
using ORTools

# To use functions exposed by this interface
import MathOptInterface as MOI

# Model initialization
optimizer = ORTools.Optimizer()
```

Model Initialization

Variable Definition

```
# Variable coefficients in the objective function
c = [2.0, 4.0]

# Variable definition
# We have added 2 variables to our model. One representing x
& the other y.
x = MOI.add_variables(model, length(c))
```

ORTools.jl in Action (The Code)

```
# The nitrogen constraint (0.10x + 0.05y >= 20)
C1 = 20
MOI.add_constraint(
    model,
    MOI.ScalarAffineFunction(
        [MOI.ScalarAffineTerm(0.10, x[1]),
        MOI.ScalarAffineTerm(0.05, x[2])],
        0.0,
    ),
    MOI.LessThan(C1),
)
```

Constraints Definition

```
# The phosphate constraint (0.06x + 0.10y >= 18)
C2 = 18
MOI.add_constraint(
    model,
    MOI.ScalarAffineFunction(
        [MOI.ScalarAffineTerm(0.06, x[1]),
MOI.ScalarAffineTerm(0.10, x[2])],
        0.0,
    ),
    MOI.LessThan(C2),
)
```

```
# The x variable lower bound constraint (x >= 0)
MOI.add_constraint(model,
      MOI.VariableIndex(1),
      MOI.GreaterThan(0.0))
```

```
# The y variable lower bound constraint (y >= 0)
MOI.add_constraint(model,
        MOI.VariableIndex(2),
        MOI.GreaterThan(0.0))
```

ORTools.jl in Action (The Code)

```
# Set the objective
MOI.set(
    model,
    MOI.ObjectiveFunction{MOI.ScalarAffineFunction{Float64}}(),
    MOI.ScalarAffineFunction(MOI.ScalarAffineTerm.(c, x), 0.0),
)

# We want to minimize our cost.
MOI.set(model, MOI.ObjectiveSense(), MOI.MIN_SENSE)

# Print out the model
print(model)
```

Objective Function

Solve the Model

```
# Solve!
MOI.optimize!(model)

# Print the solution
print(model.solve_result)
```

JuMP Interface

Given that ORTools.jl implements the MathOptInterface.jl, ORTools will be accessible through the JuMP interface; *a domain-specific modeling language for mathematical optimization embedded in Julia (from JuMP's docs).*

JuMP/ORTools Example

Model Initialization

```
using JuMP

import ORTools

model = Model(ORTools.Optimizer)
```

Variables

Constraints

Objective

```
@variable(model, x >= 0)
@variable(model, y >= 0)

@constraint(model, 0.10x + 0.05y >= 20)
@constraint(model, 0.06x + 0.10y <= 18)

@objective(model, Min, 2x + 4y)
```

JuMP & ORTools

```
# Solve!
optimize!(model)
```

Solve the Model

Result retrieval

```
# Get the objective value, the minimized cost.
println("The maximized profit is: ", objective_value(model))

# Values assigned to the variables
println("x: ", value(x))
println("y: ", value(y))
```

Examples of how Google solvers are used

- Workforce scheduling API (https://developers.google.com/optimization/service/scheduling/workforce_scheduling)
  - Take the example of a hospital: you have nurses, you have wards
  - Who works when?
  - Abide by social laws, like minimum rest time between two shifts
  - Have nurses with the right background (minimum number of experienced nurses, e.g.)
  - Minimise overtime
- Shipping network design API (https://developers.google.com/optimization/service/shipping/network_design)
  - What routes should ships take in the next year?

How Google uses solvers

- Resource and capacity planning in data centres
    - Optimise the supply chain of spare parts: how many spare RAM sticks do you need, knowing how often they break for the types of machines in a data centre?
    - Optimise server deployments: how do you deploy hundreds of new servers in an existing data centre while not consuming too much power? (Limit of power draw per rack!)
    - Optimise network routing, load balancing, etc.

Real-World Applications

Mathematical optimization and tools like ORTools.jl are used everywhere to solve billion-dollar problems:

- **Logistics & Supply Chain:**
    - Vehicle Routing (The "Traveling Salesperson Problem")
    - Warehouse placement and inventory management.
- **Scheduling:**
    - Employee shift scheduling.
    - Job-shop scheduling in manufacturing.
    - Airline fleet and crew assignment.
- **Finance:**
    - Portfolio optimization to balance risk and return.
- **Energy:**
    - Power plant dispatch and grid management.

And many more.....

Development Hurdles

- MathOptInterface.jl implementation is a hustle.
- ProtoBuf-Struct mutability support; opened a PR here.
- JLL wrapper (took a long time…)

Next Plans…

- Complete support for PDLP.
- Offer public documentation with examples.
- Complete support for CP-SAT.
- Add support for remote solve calls
- Add support for quadratic optimization problems
- Add support for multiple objectives.
- First major release in September 2025….

ORTools.jl

Thank you for listening.

Ochibobo Warren.
linkedin.com/wochibo