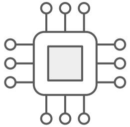
Combinatorial Optimization at Google
Tools, solvers, and applications

CO@Work 2024, Pawel Lichocki, 27.09.2024

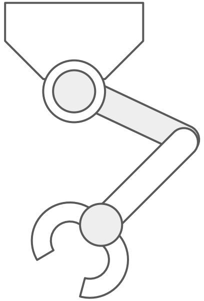
https://developers.google.com/optimization

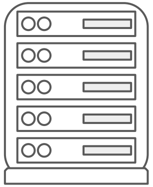
CO is everywhere



Hardware

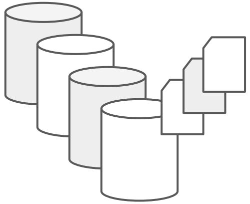
Supply chain

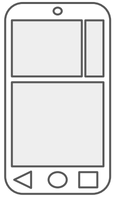




Infrastructure

Backends





Frontends

Outline

- OR-Tools
  - PDLP
  - CP-SAT
  - MathOpt
- Life of an optimization project
  - Modelling
  - Solving
  - Landing

Outline

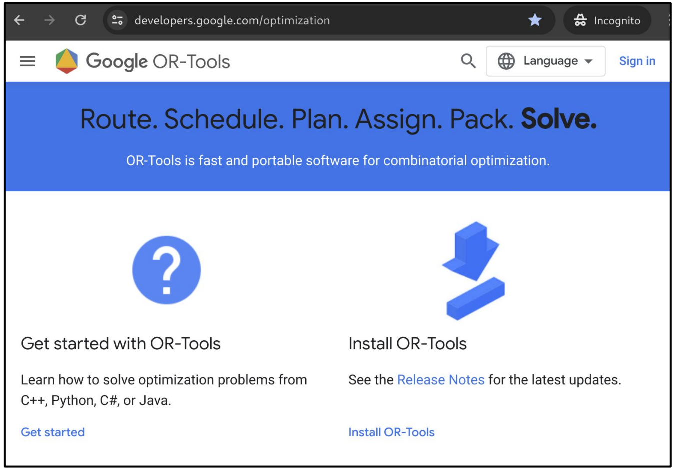
OR-Tools

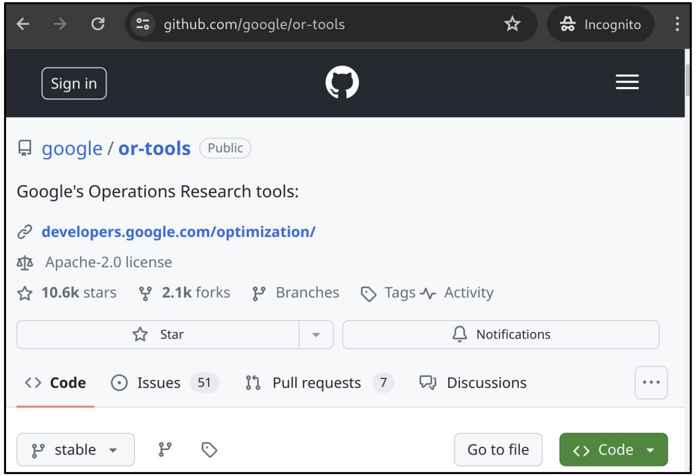
- PDLP
- CP-SAT
- MathOpt

Life of an optimization project

- Modelling
- Solving
- Landing

OR-Tools



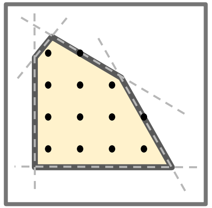
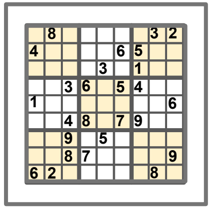
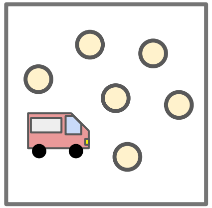
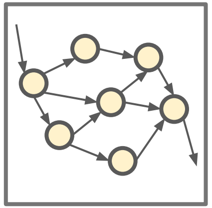


https://developers.google.com/optimization

https://github.com/google/or-tools

OR-Tools

A mature open-source to model and solve LP / MIP, CP, VRP, and graph problems

Multilingual API

C++, Python, C#, Java

Many solvers

GLOP	PDLP	Graph algos
CP-SAT	VRP	...

Easy wrappers

Math Opt

MP Model

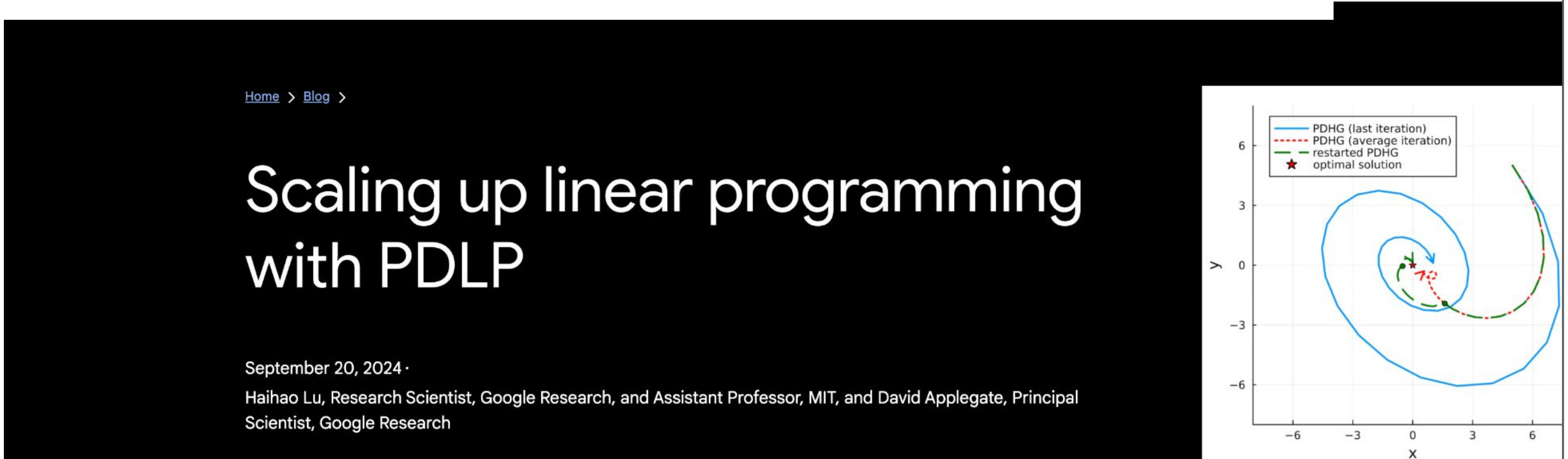
PDLP

Matrix-vector multiplication instead of matrix factorization

2024 Beale — Orchard-Hays Prize for

Excellence in Computational Mathematical Programming

https://research.google/blog/scaling-up-linear-programming-with-pdlp/



PDLP... a large scale LP solver

PDLP has been used to solve real-world problems with as many as 12B non-zeros (and an internal distributed version scaled to 92B non-zeros).

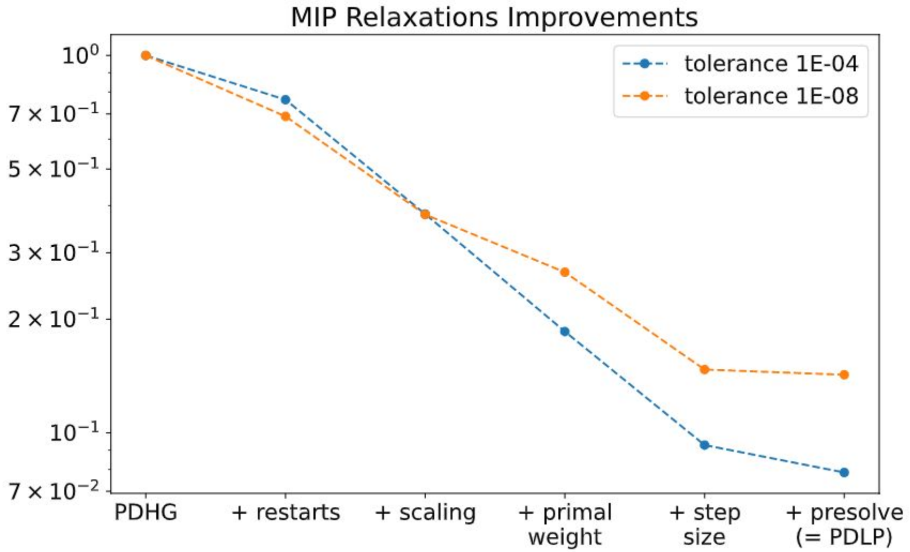
Instance	variables	constraints	nonzeros	iterations	time (h)
heat-source-recovery	6,886,808	3,375,000	27,000,000	3,688,640	43.9
supply-chain	1,005,000,100	11,050,100	2,015,000,100	125,280	272.3
synthetic-design-match	10,000,000	5,500,135	690,000,000	15,068	1.2
gaia100m	1,184,557,727	162,934,799	6,337,834,450	1,979	10.2

Preliminary results, 1e-2 optimality gap, 1e-8 primal/dual feasibility tolerance

PDLP... theoretical advancements and top engineering

Method	PDHG	PDLP
Solved to 10^{-4} (relative err.)	**37%**	**91%**
Solved to 10^{-8} (relative err.)	**13%**	**75%**

~400 LP relaxations from MIPLIP 2017
~100k iterations



CP-SAT... constraints

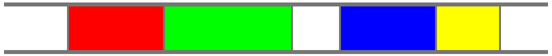
Linear and non-linear constraints

Only integer variables

Always exact

CP-SAT... constraints

NoOverlapConstraint



Intervals cannot overlap

CP-SAT... constraints

NoOverlapConstraint
CumulativeConstraint
NoOverlap2D
Boolean constraints
ElementConstraint
CircuitConstraint
LinearConstraint (with enforcement)

...

3 <= x + 2y + z <= 19
B => (x <= 2)

CP-SAT... the best of all worlds

(Max)Sat

- Core based search
- Model reductions
- Clause Learning

Constraint Programming

- Rich modeling layer (structure is not lost in the solver)
- Advanced deduction algorithms

Linear Integer Programming

- Linear Relaxation (via Glop)
- Cuts
- Presolve

Meta-heuristics

- Large Neighborhood Search
- Violation based Local Search

CP-SAT... the best of all CP solvers

Medals

MiniZinc Challenge 2024

Category	Gold	Silver	Bronze
Fixed	OR-Tools CP-SAT	Choco-solver CP-SAT	SICStus Prolog
Free	OR-Tools CP-SAT	PicatSAT	iZplus
Parallel	OR-Tools CP-SAT	PicatSAT	Choco-solver CP
Open	OR-Tools CP-SAT	PicatSAT	Choco-solver CP
Local Search	OR-Tools CP-SAT LS	Yuck	

MiniZinc Challenge 2023

Category	Gold	Silver	Bronze
Fixed	OR-Tools	SICStus Prolog	Choco 4
Free	OR-Tools	PicatSAT	iZplus
Parallel	OR-Tools	PicatSAT	Choco 4
Local Search	Yuck		

MiniZinc Challenge 2022

Category	Gold	Silver	Bronze
Fixed	OR-Tools	SICStus Prolog	JaCoP
Free	OR-Tools	PicatSAT	Choco 4
Parallel	OR-Tools	PicatSAT	Geas
Local Search	Yuck		

MiniZinc Challenge 2021

Category	Gold	Silver	Bronze
Fixed	OR-Tools	JaCoP	SICStus Prolog
Free	OR-Tools	PicatSAT	iZplus
Parallel	OR-Tools	PicatSAT	iZplus + Choco 4
Open	OR-Tools	PicatSAT	iZplus + Choco 4
Local Search	Yuck	OscaR/CBLS	

MiniZinc Challenge 2020

Category	Gold	Silver	Bronze
Fixed	SICStus Prolog	JaCoP	Choco 4
Free	OR-Tools	PicatSAT	Mistral 2.0
Parallel	OR-Tools	PicatSAT	Mistral 2.0
Open	OR-Tools	sunny-cp—	PicatSAT
Local Search	Yuck	OscaR/CBLS	

MiniZinc Challenge 2019

Category	Gold	Silver	Bronze
Fixed	OR-Tools	JaCoP	SICStus Prolog
Free	OR-Tools	iZplus	Picat SAT
Parallel	OR-Tools	iZplus	Choco 4
Open	OR-Tools	sunny-cp—	iZplus
Local Search	iZplus	Yuck	OscaR/CBLS

MiniZinc Challenge 2018

Category	Gold	Silver	Bronze
Fixed	OR-Tools	JaCoP	Choco 4
Free	OR-Tools	Picat SAT	Choco 4 + HaifaCSP
Parallel	OR-Tools	Choco 4	Picat SAT
Open	OR-Tools	sunny-cp—	Choco 4
Local Search	iZplus	Yuck	OscaR/CBLS

MiniZinc Challenge 2017

Category	Gold	Silver	Bronze
Fixed	OR-Tools LCG	JaCoP	Choco 4
Free	iZplus	OR-Tools LCG	Picat SAT
Parallel	Choco 4	iZplus	OR-Tools LCG
Open	sunny-cp—	Choco 4	OR-Tools LCG
Local Search	iZplus	Yuck	OscaR/CBLS

MiniZinc Challenge 2016

Category	Gold	Silver	Bronze
Fixed	OR-Tools	JaCoP	Choco
Free	HaifaCSP	Picat SAT	iZplus
Parallel	HaifaCSP	Picat SAT	iZplus
Open	sunny-cp——	HaifaCSP	Picat SAT
Local Search	OscaR/CBLS		

MiniZinc Challenge 2015

Category	Gold	Silver	Bronze
Fixed	Opturion CPX	OR-Tools	JaCoP
Free	Opturion CPX	iZplus	OR-Tools
Parallel	OR-Tools	Opturion CPX	Choco
Open	sunny-cp——	OR-Tools	Opturion CPX

MiniZinc Challenge 2014

Category	Gold	Silver	Bronze
Fixed	OR-Tools	Opturion CPX	SICStus Prolog
Free	iZplus	Opturion CPX	Choco
Parallel	OR-Tools	Opturion CPX	Choco
Open	OR-Tools	Opturion CPX	Choco

MiniZinc Challenge 2013

Category	Gold	Silver	Bronze
Fixed	Opturion/CPX	OR-Tools	Gecode
Free	Opturion/CPX	OR-Tools	izplus
Parallel	OR-Tools	Choco	Opturion/CPX
Open	OR-Tools	Choco	Opturion/CPX

MiniZinc Challenge 2012

Category	Gold	Silver	Bronze
Fixed	Gecode	JaCoP	OR-Tools
Free	Gecode	Fzn2smt	izplus

CP-SAT... and a strong linear integer solver

Closed open MIPLIB 2017 problems

amaze22012-07-04i (31s)

neos-3209462-rhin (87s)

l2p2i (16s)

neos-3214367-sovi (341s, vs. solved in 21 days with ParaXpress)

stoch-vrpvrp-s5v2c8vrp-v2c8i (30s)

MathOpt

```
$ pip install ortools
$ python3

>>> from ortools.math_opt.python import mathopt
```

MathOpt... separates modeling and solving

```python
from ortools.math_opt.python import mathopt

# Build the model
model = mathopt.Model()
x = model.add_binary_variable(name='x')
y = model.add_variable(lb=0.0, ub=1.0, name='y')
model.maximize(x + 3.0 * y)
model.add_linear_constraint(x + y <= 1.5)

# Set parameters, e.g. turn on logging.
p = mathopt.SolveParameters(enable_output=True)

# Solve the model, ensure the solver found the solution.
result = mathopt.solve(model, mathopt.SolverType.GSCIP, params=p)
assert result.termination.reason == mathopt.TerminationReason.OPTIMAL

# Print the optimal variable values.
print(result.variable_values([x, y]))
```

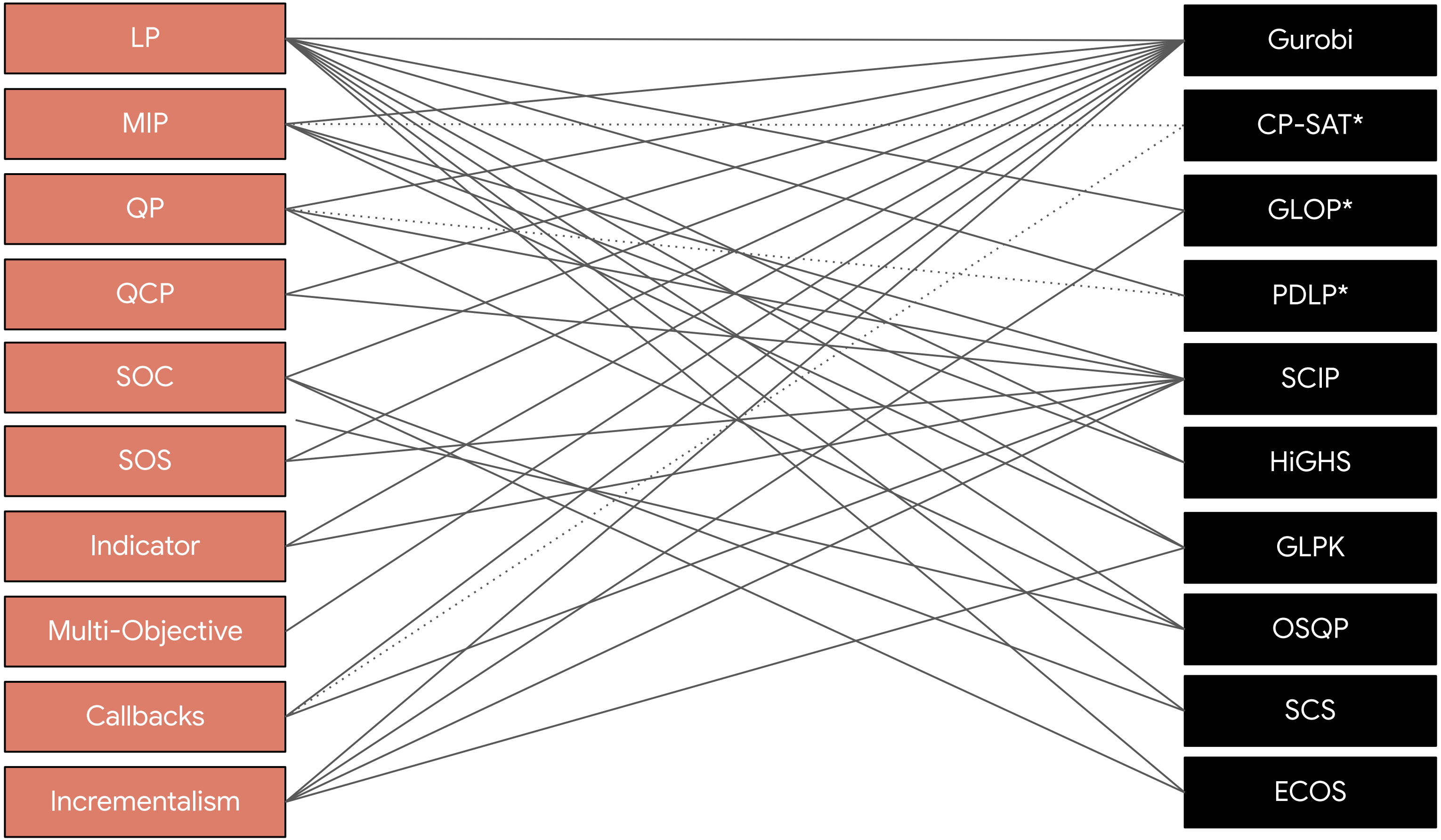
$\max\ x + 3y$

$x + y \leq 1.5$

$x \in \{0, 1\}$

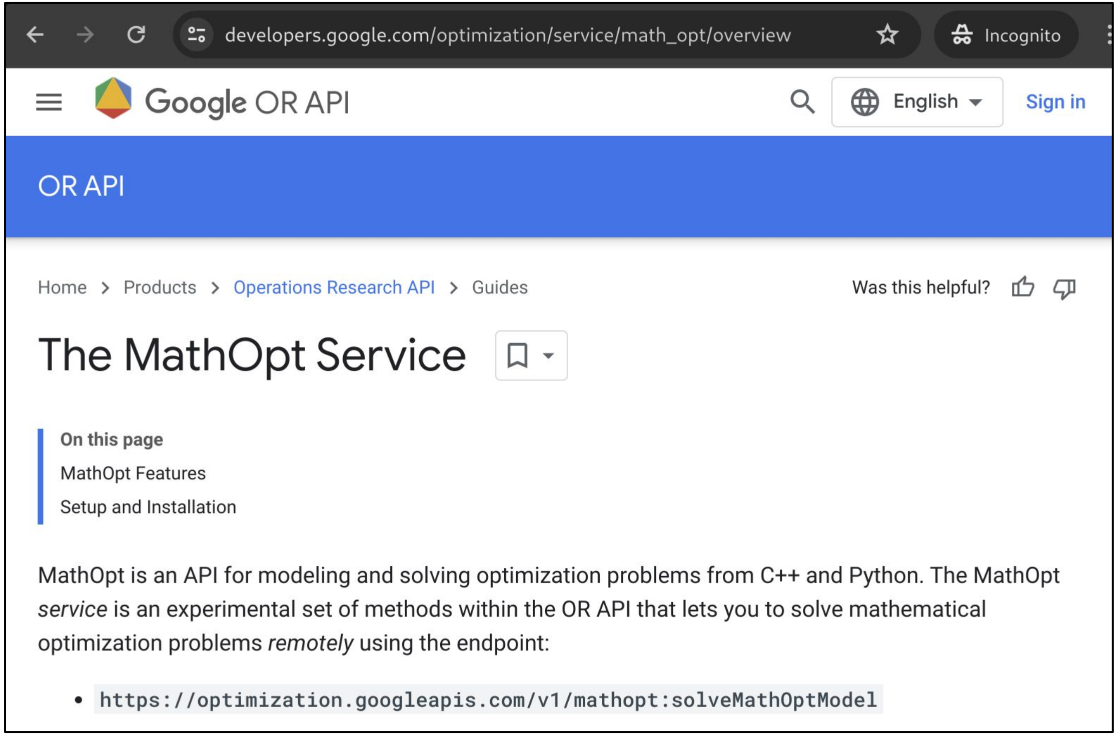
$y \in [0, 1]$

MathOpt... many features, many solvers



* Made by Google

MathOpt... solve remotely for free!



https://developers.google.com/optimization/service/math_opt/overview

MathOpt... solve remotely for free!

```
from ortools.math_opt.python import mathopt
from ortools.math_opt.python.ipc import remote_http_solve

model = mathopt.Model()
x = model.add_binary_variable(name='x')
y = model.add_variable(lb=0.0, ub=1.0, name='y')
model.maximize(x + 3.0 * y)
model.add_linear_constraint(x + y <= 1.5)

try
  result, logs = remote_http_solve.remote_http_solve(
      model,
      mathopt.SolverType.GSCIP,
      mathopt.SolveParameters(enable_output=True),
      api_key=api_key,
  )
  print("Objective value: ", result.objective_value())
  print("\n".join(logs))
except remote_http_solve.OptimizationServiceError as err:
  print(err)
```

Outline

OR-Tools

PDLP - *The largest-scale LP solver in the world*

CP-SAT - *The best CP solver in the world*

MathOpt - *One wrapper, many features, many solvers*

Life of an optimization project

Modelling

Solving

Landing

Outline

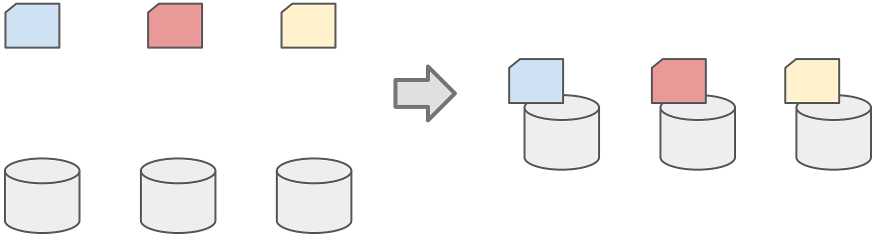
OR-Tools

- PDLP - *The largest-scale LP solver in the world*
- CP-SAT - *The best CP solver in the world*
- MathOpt - *One wrapper, many features, many solvers*

Life of an optimization project

- Modelling
- Solving
- Landing

Modelling... placing items in containers



Indices

Item i = 1..I

Bin b = 1..B

Resource r = 1..R

Variables

place(i, b) ∈ {0, 1}

Constants

double Required(i, r)

double Available(b, r)

Constraints

Objective

Indices

Item i = 1..I

Bin b = 1..B

Resource r = 1..R

Variables

place(i, b) ∈ {0, 1}

Constants

double Required(i, r)

double Available(b, r)

Constraints

$\forall_{i} \quad \sum_{b} place(i, b) = 1$

Objective

Indices

Item i = 1..I

Bin b = 1..B

Resource r = 1..R

Variables

place(i, b) ∈ {0, 1}

Constants

double Required(i, r)

double Available(b, r)

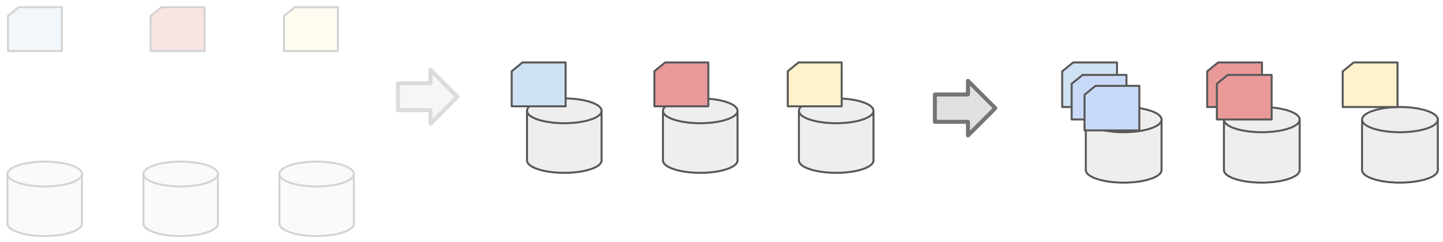
Constraints

$$\forall_i \quad \sum_b \text{place}(i, b) = 1$$

$$\forall_r \ \forall_b \quad \sum_i \text{Required}(i, r) \cdot \text{place}(i, b) \leq \text{Available}(b, r)$$

Objective

Modelling... redundancy



Indices

Item i = 1..I

Bin b = 1..B

Resource r = 1..R

Variables

place(i, b) ∈ [0..Copies(i)]

Constants

int Copies(i)

double Required(i, r)

double Available(b, r)

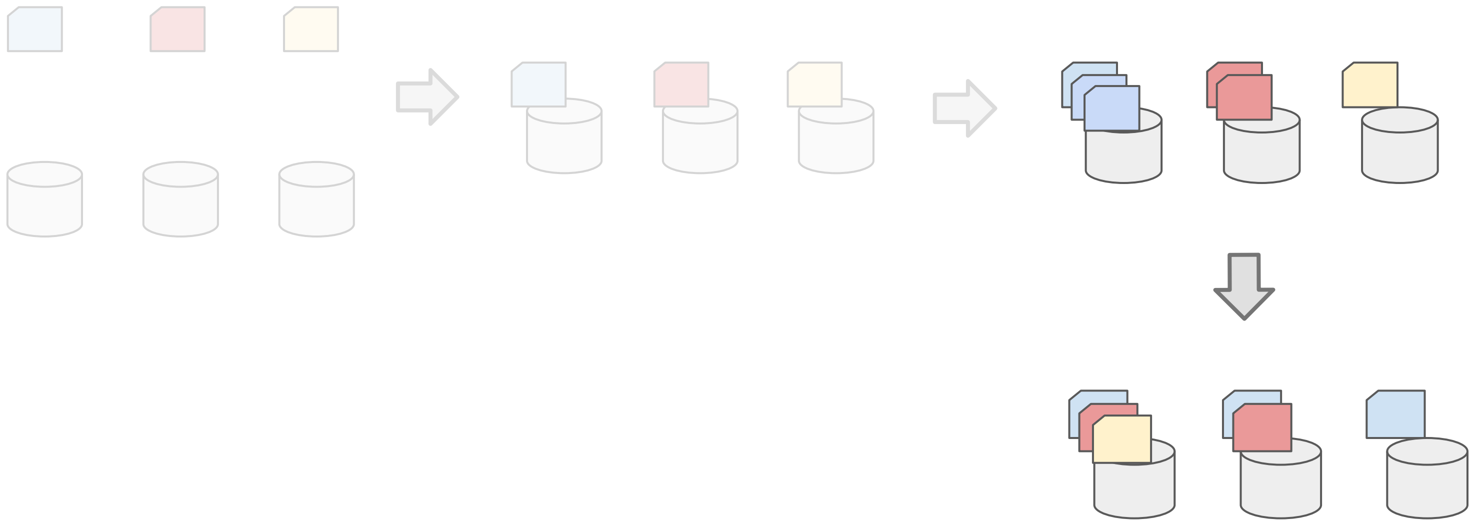
Constraints

$\forall_i \quad \sum_b \text{place}(i, b) = \text{Copies}(i)$

$\forall_r \ \forall_b \quad \sum_i \text{Required}(i, r) \cdot \text{place}(i, b) \leq \text{Available}(b, r)$

Objective

Modelling... fault tolerance



Indices

Item i = 1..I

Bin b = 1..B

Resource r = 1..R

Variables

place(i, b) ∈ {0, 1}

Constants

int Copies(i)

double Required(i, r)

double Available(b, r)

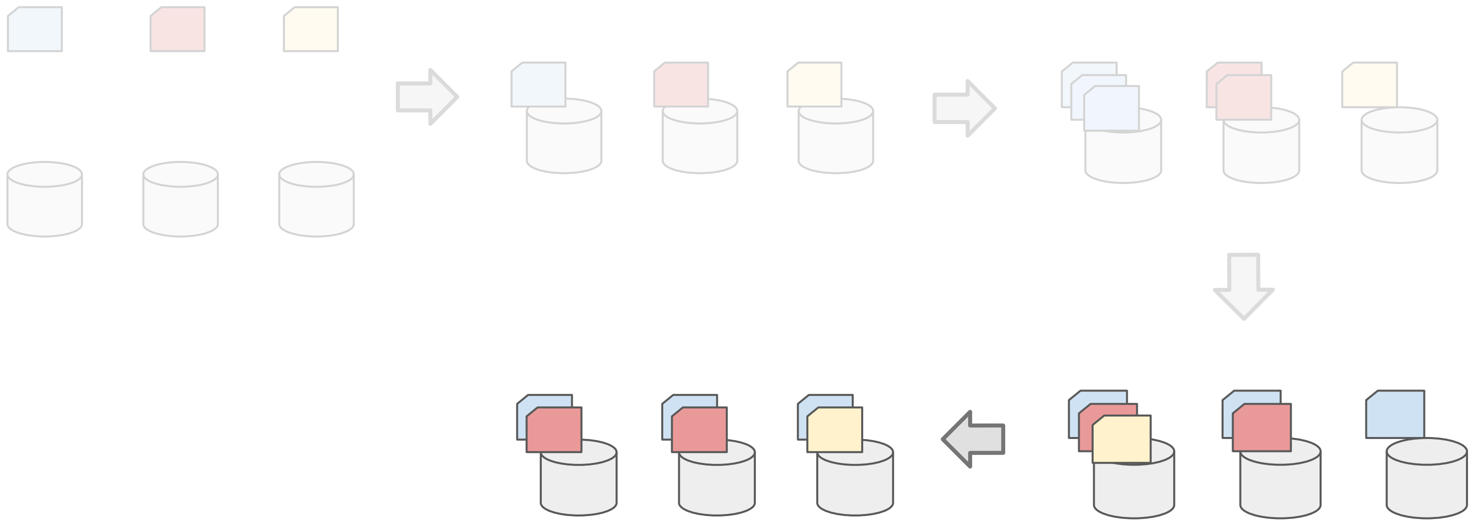
Constraints

$\forall_i \quad \sum_b \text{place}(i, b) = \text{Copies}(i)$

$\forall_r \ \forall_b \quad \sum_i \text{Required}(i, r) \cdot \text{place}(i, b) \le \text{Available}(b, r)$

Objective

Modelling... load balancing



Indices

Item i = 1..I
Bin b = 1..B
Resource r = 1..R

Variables

place(i, b) ∈ {0, 1}
surplus(b) ∈ [0, +inf)

Constants

int Copies(i)
double Required(i, r)
double Available(b, r)

Constraints

$\forall_i \quad \sum_b \text{place}(i, b) = \text{Copies}(i)$

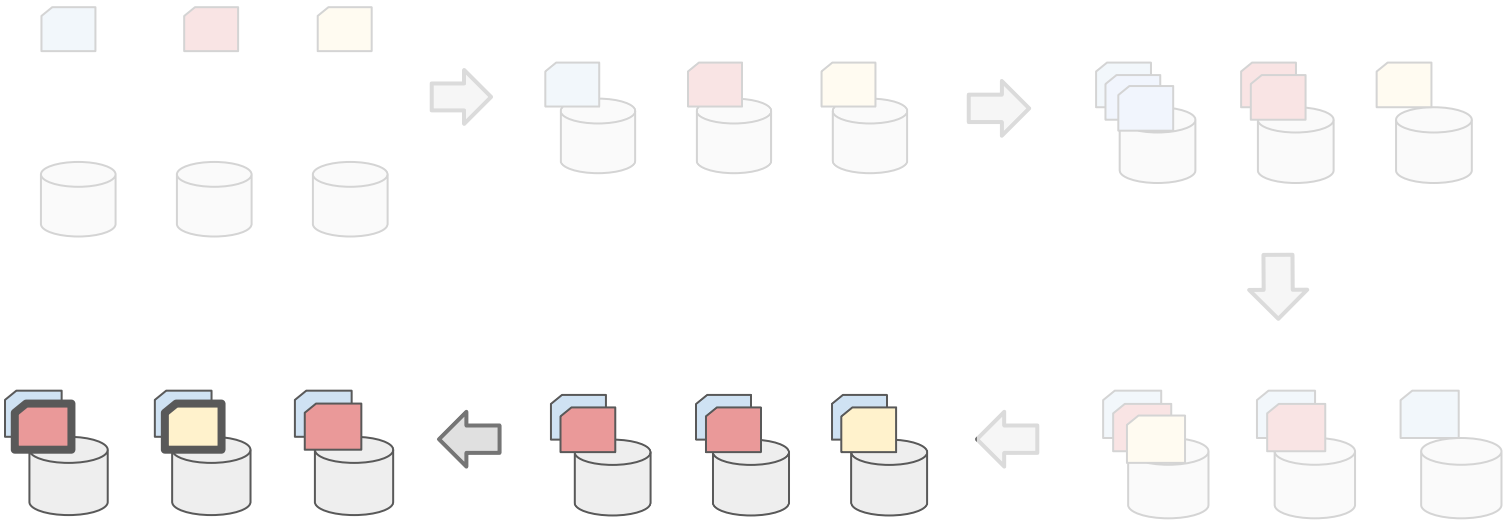
$\forall_r \ \forall_b \quad \sum_i \text{Required}(i, r) \cdot \text{place}(i, b) \leq \text{Available}(b, r)$

$\forall_b \quad \sum_i \text{place}(i, b) - \sum_i \text{Copies}(i) / B \leq \text{surplus}(b)$

Objective

$\min \sum_b \text{surplus}(b)$

Modelling... churn reduction



Indices

Item $i = 1..I$

Bin $b = 1..B$

Resource $r = 1..R$

Variables

$place(i, b) \in \{0, 1\}$

$surplus(b) \in [0, +inf)$

Constants

int $Copies(i)$

double $Required(i, r)$

double $Available(b, r)$

int MaxChange

bool $Placed(i, b)$

Constraints

$$\forall_i \quad \sum_b place(i, b) = Copies(i)$$

$$\forall_r \ \forall_b \quad \sum_i Required(i, r) \cdot place(i, b) \leq Available(b, r)$$

$$\forall_b \quad \sum_i place(i, b) - \sum_i Copies(i) / B \leq surplus(b)$$

$$\sum_b \sum_i Placed(i, b) \cdot (1 - place(i, b)) \leq MaxChange$$

Objective

$$\min \sum_b surplus(b)$$

Indices

Item $i = 1..I$

Bin $b = 1..B$

Resource $r = 1..R$

Variables

$place(i, b) \in \{0, 1\}$

$surplus(b) \in [0, +inf)$

Constants

int Copies(i)

double Required(i, r)

double Available(b, r)

int MaxChange

bool Placed(i, b)

Constraints

$$\forall_i \quad \sum_b place(i, b) = Copies(i)$$

$$\forall_r \ \forall_b \quad \sum_i Required(i, r) \cdot place(i, b) \le Available(b, r)$$

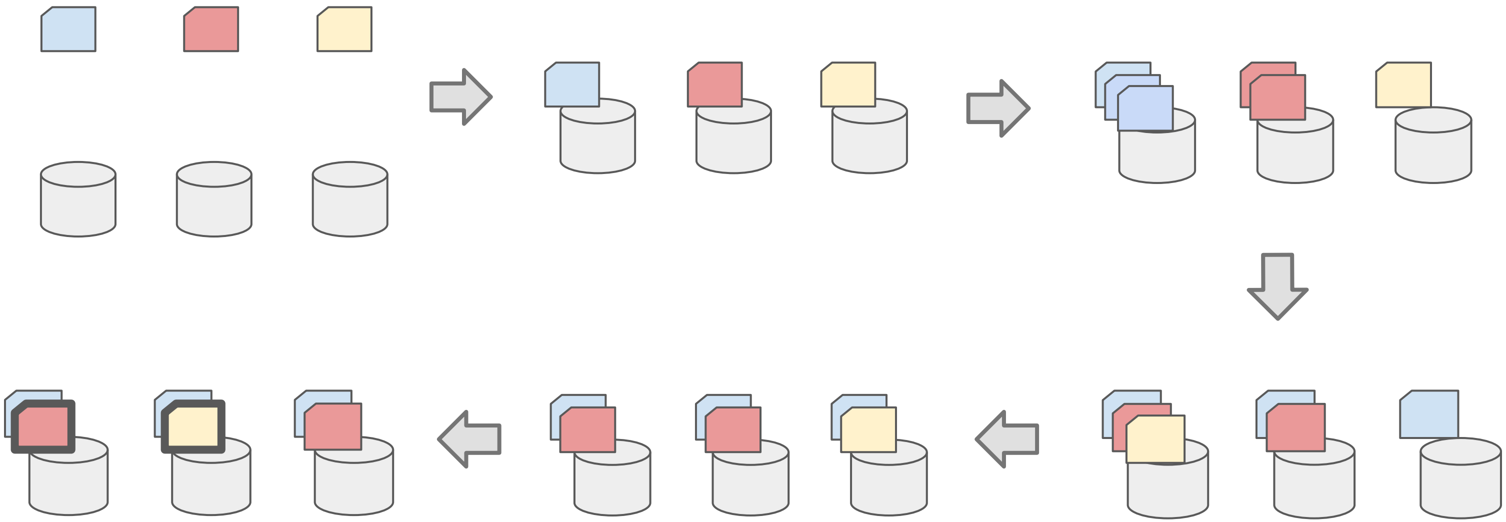
$$\forall_b \quad \sum_i place(i, b) - \sum_i Copies(i) / B \le surplus(b)$$

$$\sum_b \sum_i Placed(i, b) \cdot (1 - place(i, b)) \le MaxChange$$

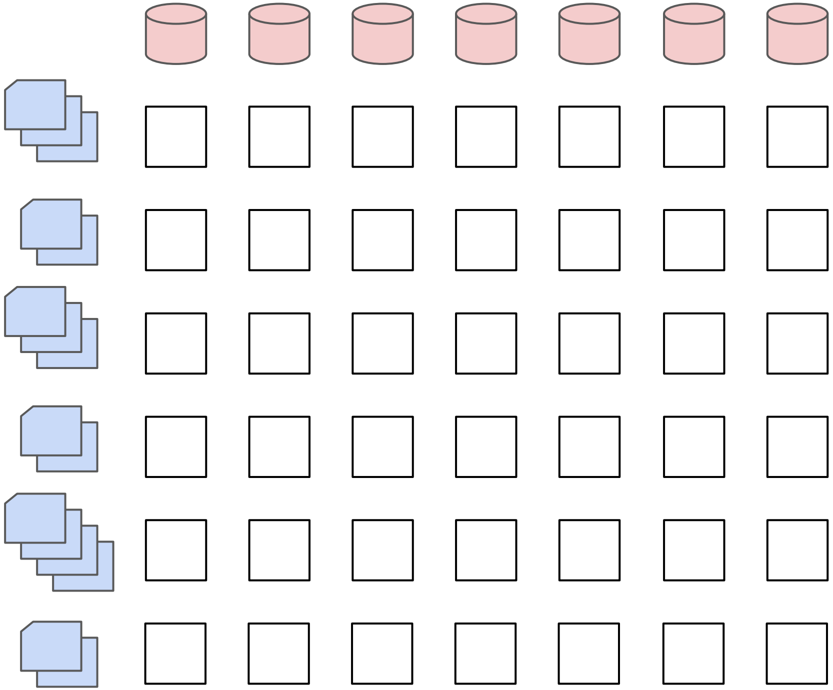
Objective

$$\min \sum_b surplus(b)$$

Modelling... multi-dimensional multi-packing with redundancy, fault tolerance, load balancing, and churn reduction

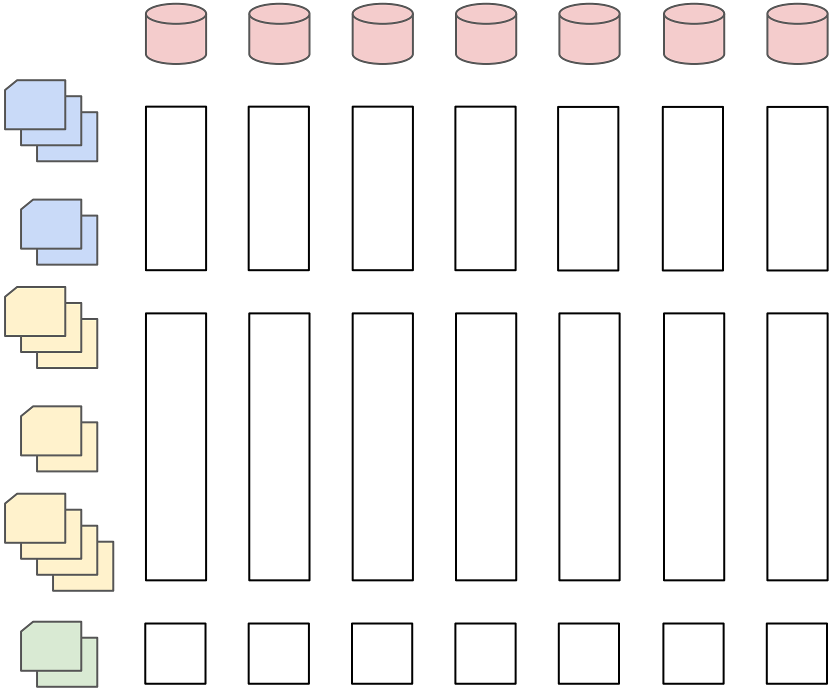


Solving...



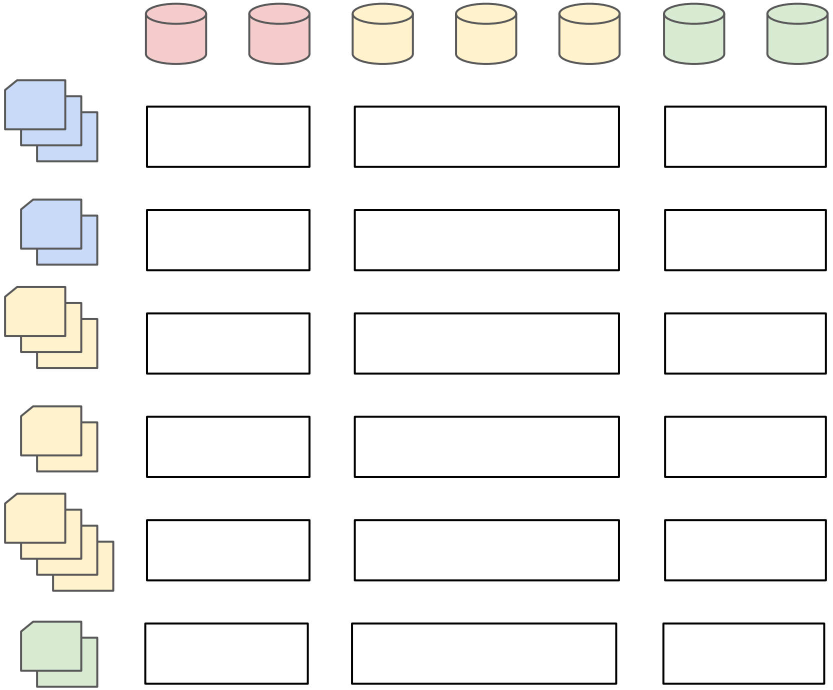
Too large?

Solving... by finding patterns



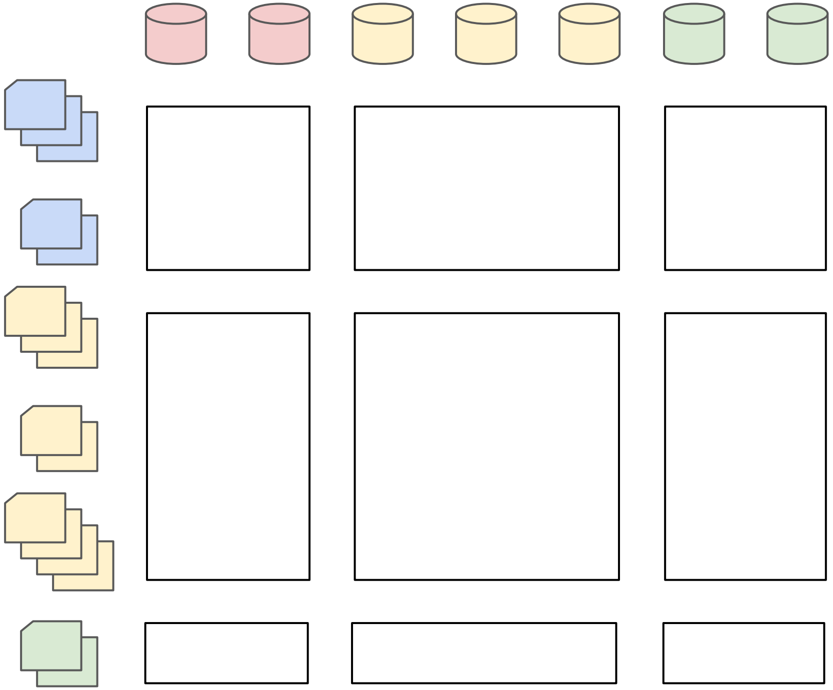
Merge items

Solving... by simplifying



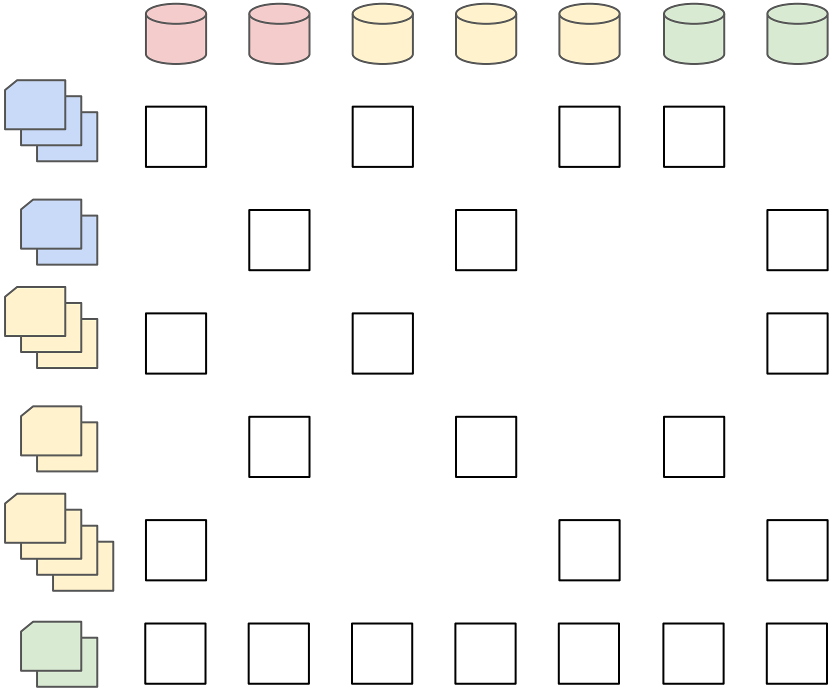
Merge bins

Solving... crudely



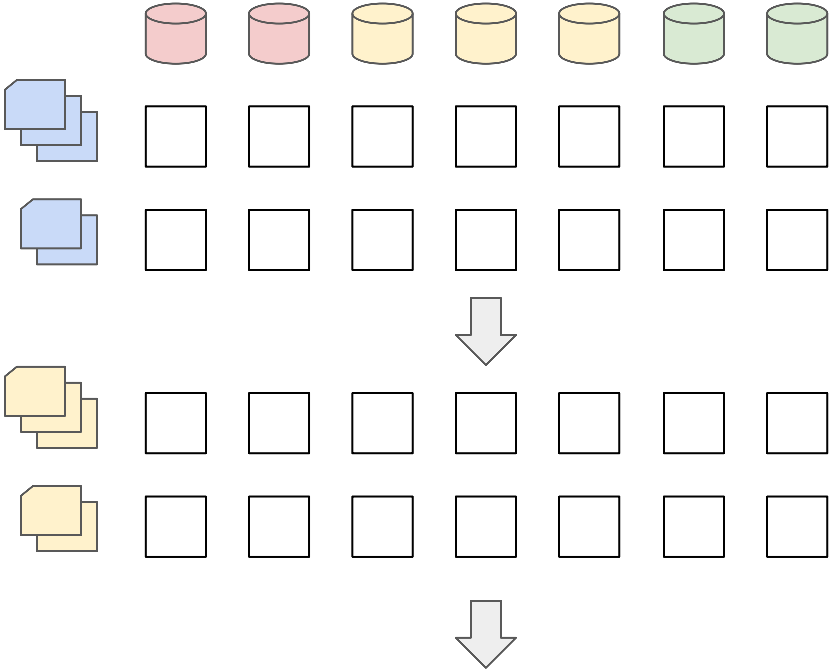
Merge both

Solving... heuristically



Skip some

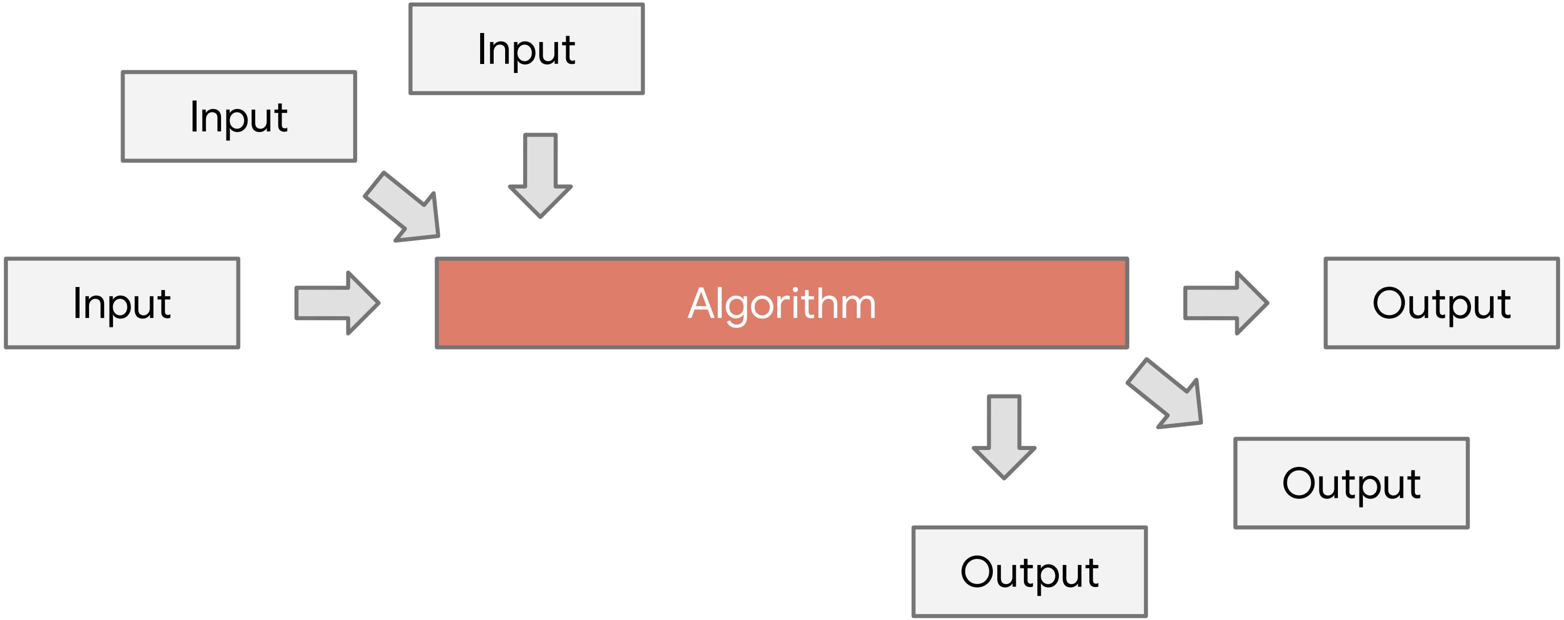
Solving... step by step



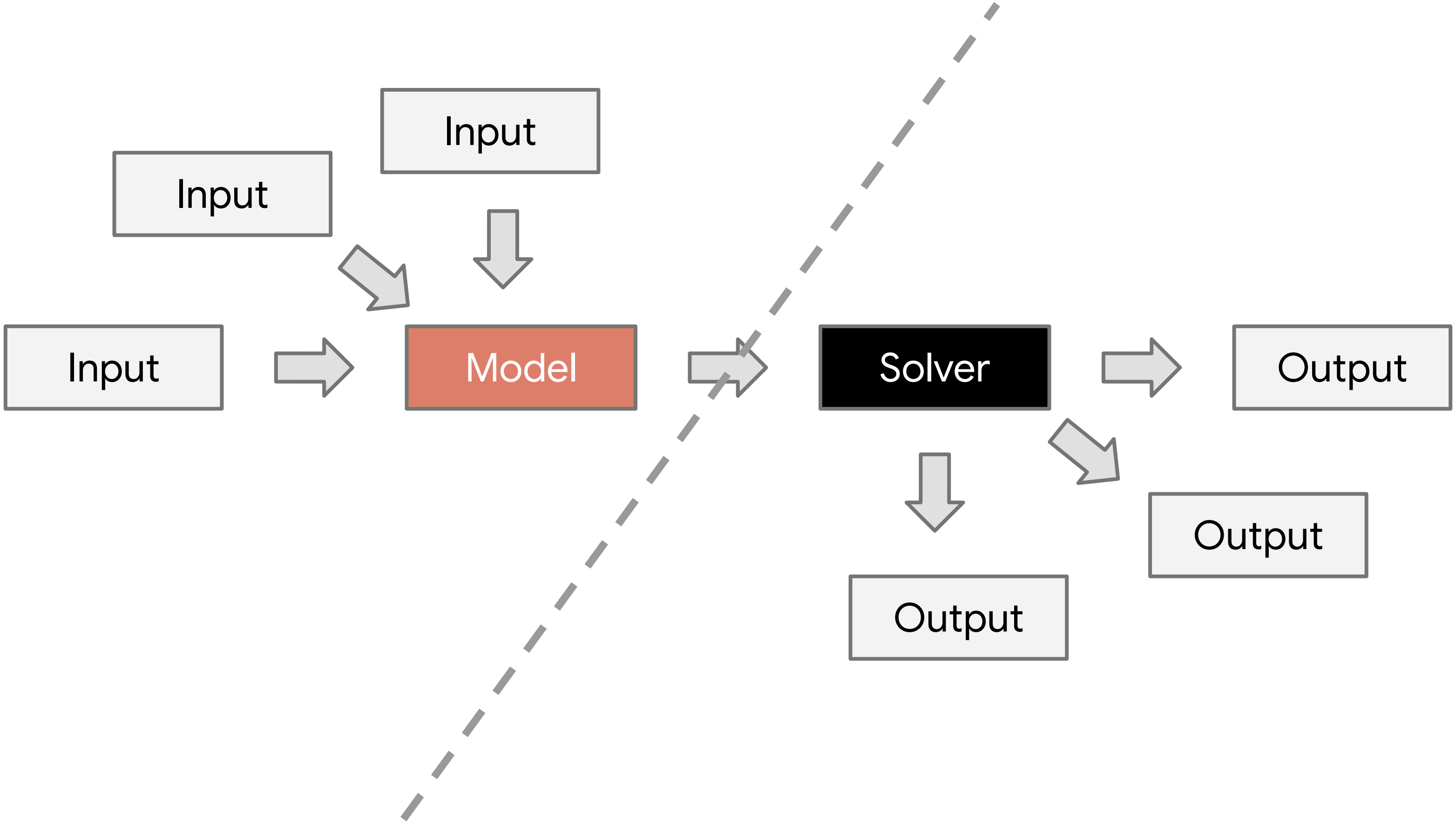
Be greedy

Landing...

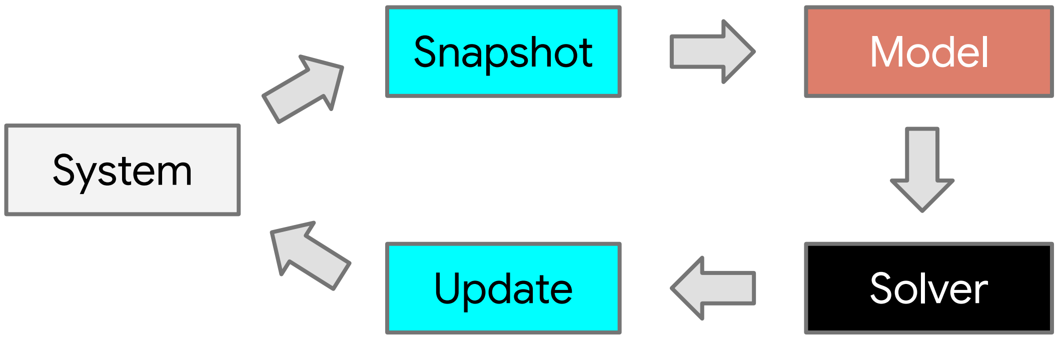
Landing... "spaghetti"



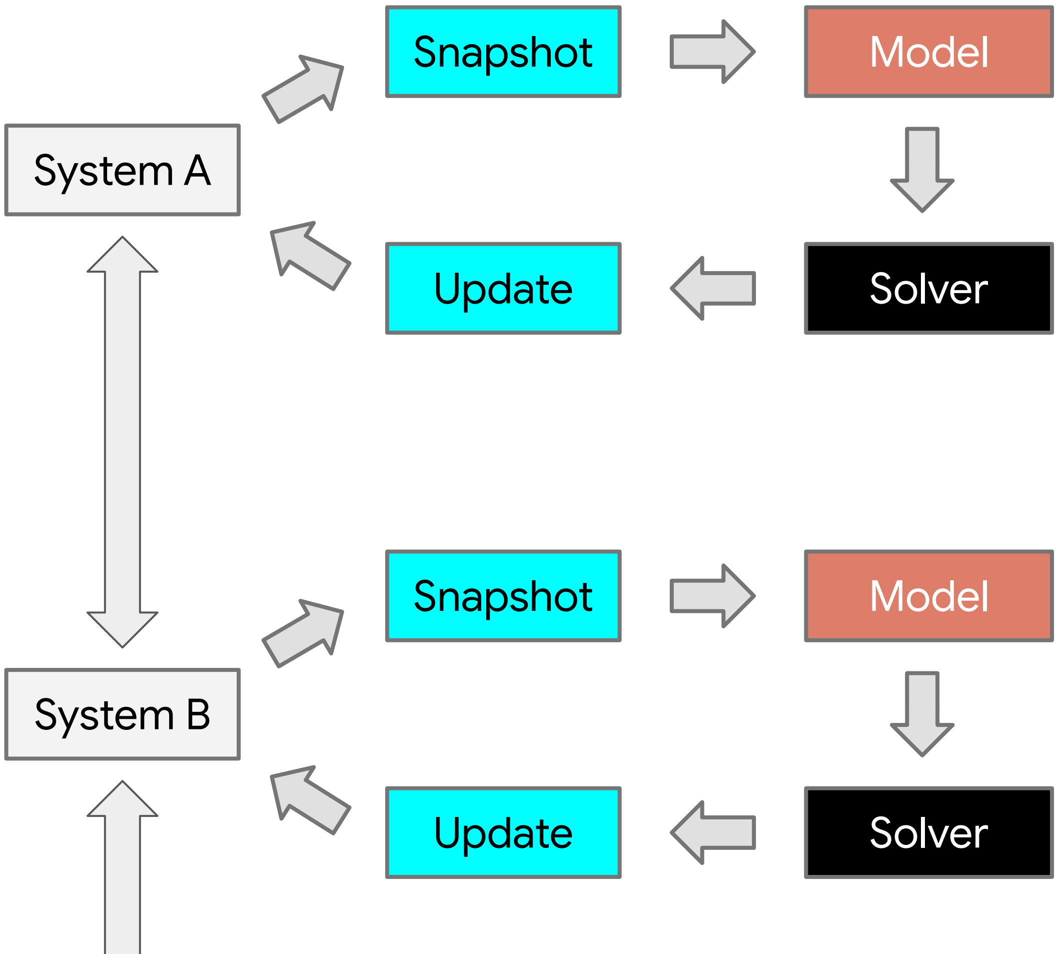
Landing... combinatorial optimization



Landing... combinatorial optimization in prod



Landing... combinatorial optimizations in prod



Thank you!

OR-Tools

PDLP - *The largest-scale LP solver in the world*

CP-SAT - *The best CP solver in the world*

MathOpt - *One wrapper, many features, many solvers*

Life of an optimization project

Modelling - *Explore and define the problem*

Solving - *Make things as simple as possible (but no simpler)*

Landing - *Leverage and maintain encapsulation*