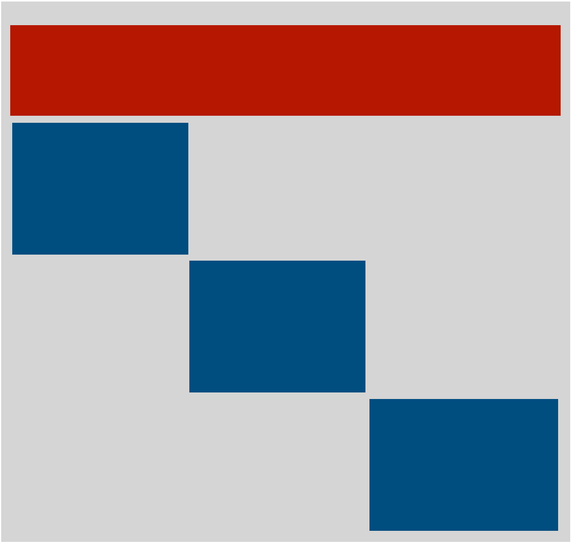
Coluna.jl: an open-source platform to implement your **creative algorithmic strategies** based on Dantzig-Wolfe and Benders decomposition

Guillaume Marques, Vitor Nesello, Artur Pessoa, Ruslan Sadykov, François Vanderbeck

Coluna.jl : in Julia with Mozilla Public Licence (MPL)

Open-source branch-and-cut-and-price framework

To optimize integer linear programs where the coefficient matrix is block structured.

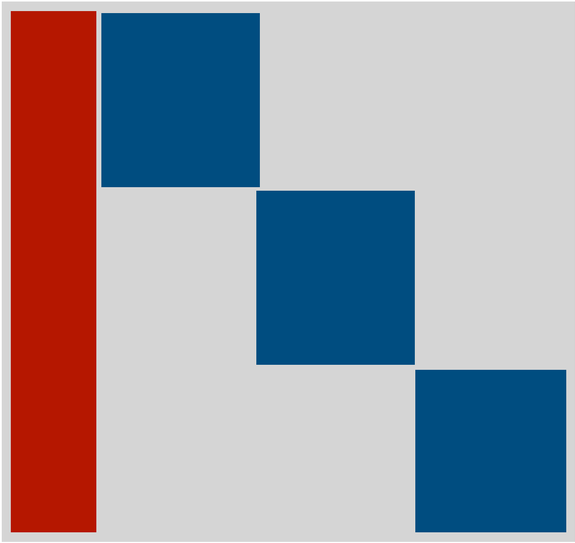


Linking constraints

Tractable subproblems

Dantzig-Wolfe decomposition

Linking variables



Benders decomposition

- Code available at https://github.com/atoptima/Coluna.jl

Model : Original formulation with JuMP , ex GAP

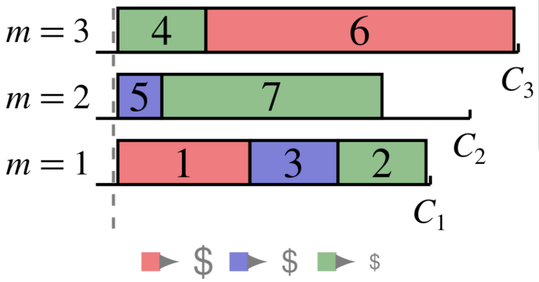
Original formulation

Each job assigned once

Knapsack on machine 1

Knapsack on machine 2

Knapsack on machine 3



Dantzig-Wolfe reformulation

Each job assigned once

$$\sum_p \lambda_p^m \leq 1 \quad m \in M$$

$$\lambda_p^m \in \{0,1\}$$

```
@variable(model, x[ j in Jobs, m in Machines], Bin)
@objective(model, Min,
        sum(cost[j,m] * x[j,m] for j in Jobs , m in Machines,))
@constraint(model, cov[j in Jobs],
        sum(x[j,m] for m in Machines) >= 1 )
@constraint(model, knp[m in Machines],
        sum(weight[j,m] * x[j,m] for j in Jobs) <= Capacity[m]) )
```

JuMP.jl

BlockDecomposition.jl

MathOptInterface.jl

Coluna.jl

Julia code, Rust code, C++, ...

MathOptInterface.jl

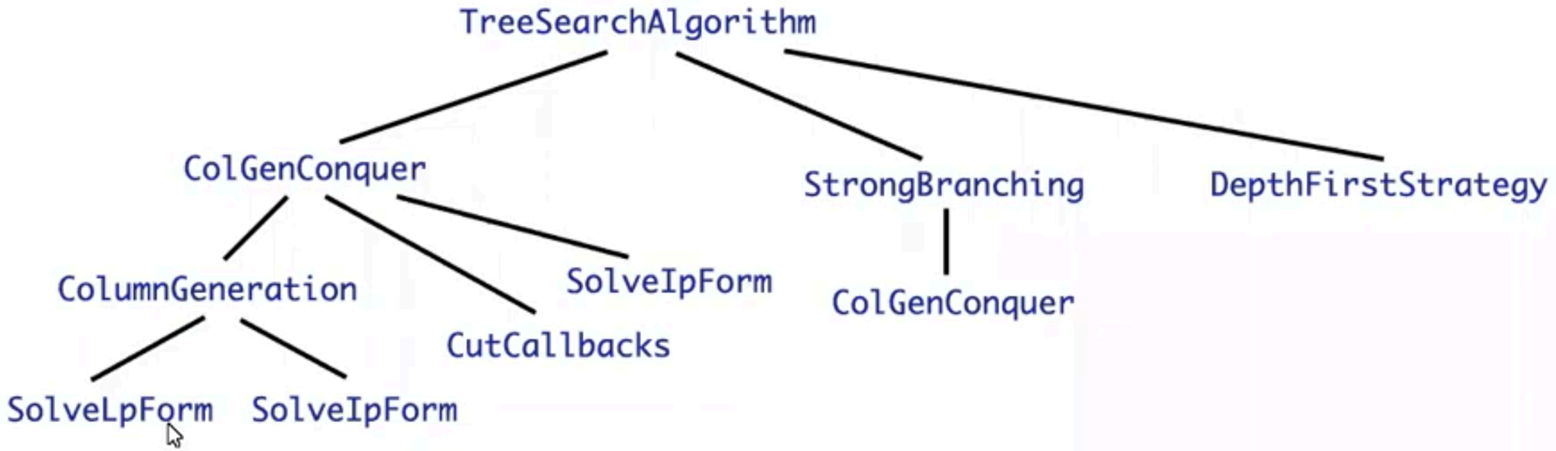
Cplex.jl

Gurobi.jl

...

Algorithms & Strategies

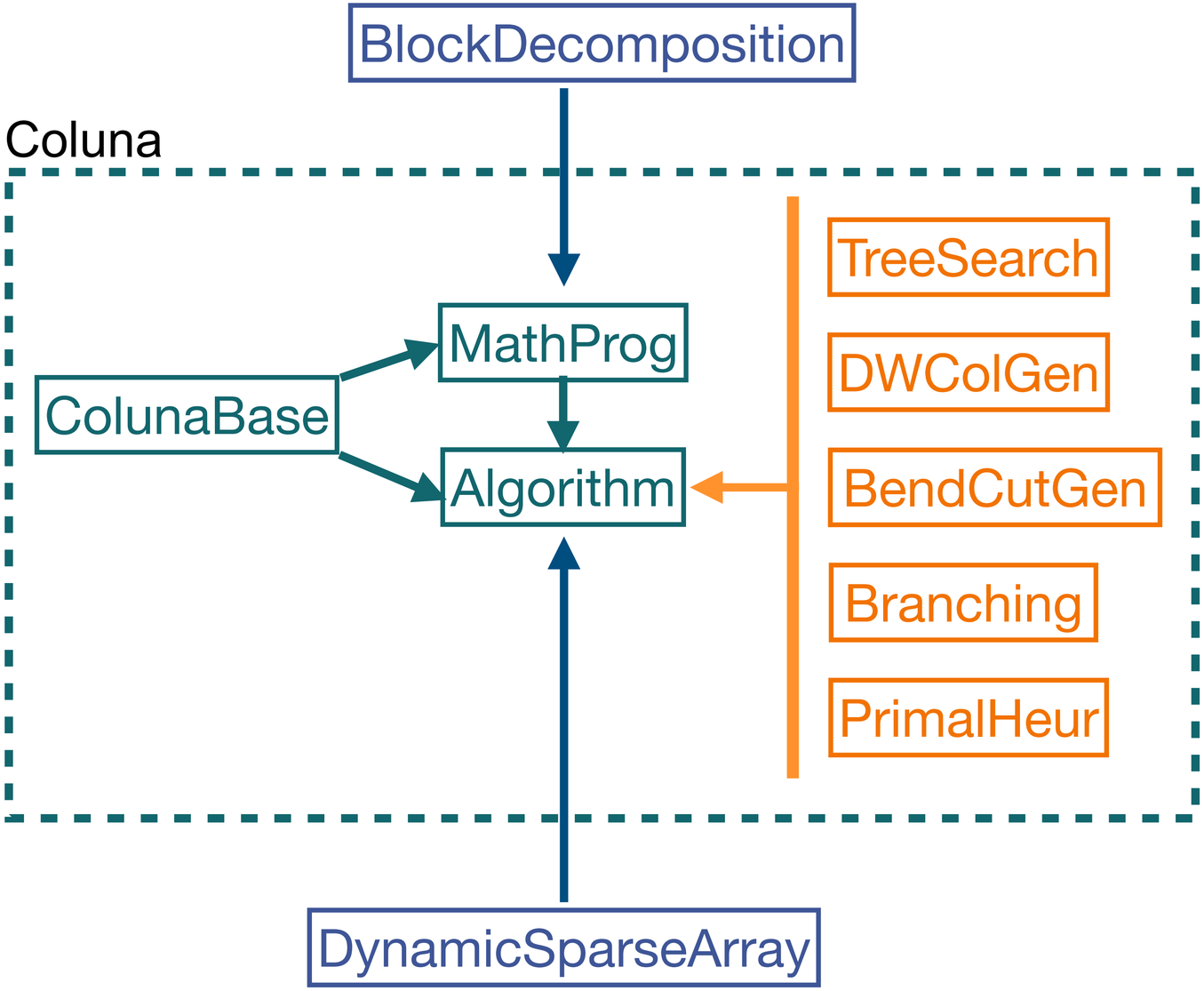
- « Text book » implementation + call-backs , f.i.
  - ➡ Pricing callback
  - ➡ Cut callback
  - ➡ Branching callback
- Customisation
  - ➡ Level-0: **callbacks**
  - ➡ Level-1: **re-implement** a method
  - ➡ Level-2: control method via « **context** » parameter
  - ➡ Level-3: re-implement an algorithm using « **composition** »




```
coluna = optimizer_with_attributes(
    Coluna.Optimizer,
    "params" => Coluna.Params(
        solver=Coluna.Algorithm.TreeSearchAlgorithm( ## default branch-and-bound of Coluna
            maxnumnodes=100,
            conqueralg=Coluna.ColCutGenConquer() ## default column and cut generation of Coluna
        ) ## default branch-cut-and-price
    ),
    "default_optimizer" => GLPK.Optimizer # GLPK for the master & the subproblems
);
```

Packages & API

- Coluna
  - ➡ ColunaBase : Data Structures
  - ➡ MathProg : managing formulations and reformulations
  - ➡ Algorithm : solver implementation
- BlockDecomposition
  - ➡ Modelling language extensions (to JuMP) to define your decomposition
- API
  - ➡ Modular methods
  - ➡ « Must-implement » interface



Algorithms API : ColGen

function run_colgen_iteration!(context, phase, stage, env, ip_primal_sol)

```
master = get_master(context)
mast_result = optimize_master_lp_problem!(master, context, env)
mast_dual_sol = get_dual_sol(mast_result)

pricing_strategy = get_pricing_strategy(context, phase)
sp_to_solve_it = pricing_strategy_iterate(pricing_strategy)

while !isnothing(sp_to_solve_it)
    (sp_id, sp_to_solve), state = sp_to_solve_it
    optimizer = get_pricing_subprob_optimizer(stage, sp_to_solve)
    pricing_result = optimize_pricing_problem!(context, sp_to_solve, env, optimizer, mast_dual_sol)

    for primal_sol in primal_sols # multi column generation support.
        if push_in_set!(context, generated_columns, primal_sol)
            nb_cols_pushed += 1
        end
    end

    sp_db = get_dual_bound(pricing_result)

    sp_to_solve_it = pricing_strategy_iterate(pricing_strategy, state)
end

col_ids = insert_columns!(get_reform(context), context, phase, generated_columns)

valid_db = compute_dual_bound(context, phase, master_lp_obj_val, sps_db, mast_dual_sol)
```

TUTORIAL: Location routing

Notations

Let i denote a customer

Let j denote a facility

Let k denote a route index number in $1, \ldots, K$

Let (u, v) denote an arc between locations

Let p denote a position in the route

Data

Let f_j be the fixed cost for openeing a facility in j.

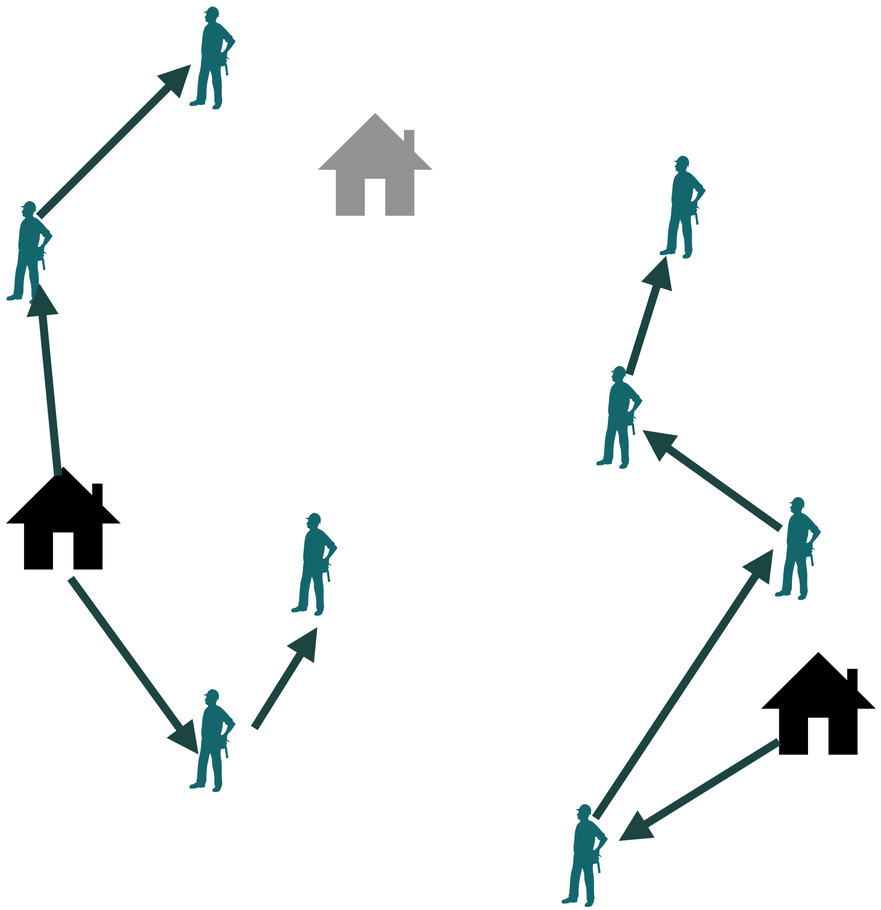
Let c_{uv} be the cost for travelling from u to v.

Variables

Let $x_{ijpk} = 1$ if customer i is in position p in the kth route starting from facility j .

Let $y_j = 1$ if facility j is open.

Let $z_{uv} = 1$ if a vehicle traverse arc (u, v).



TUTORIAL: Location routing

Original Formulation

Objective: $\min \sum_j f_j\, y_j + \sum_{u,v} c_{uv}\, z_{uv}$

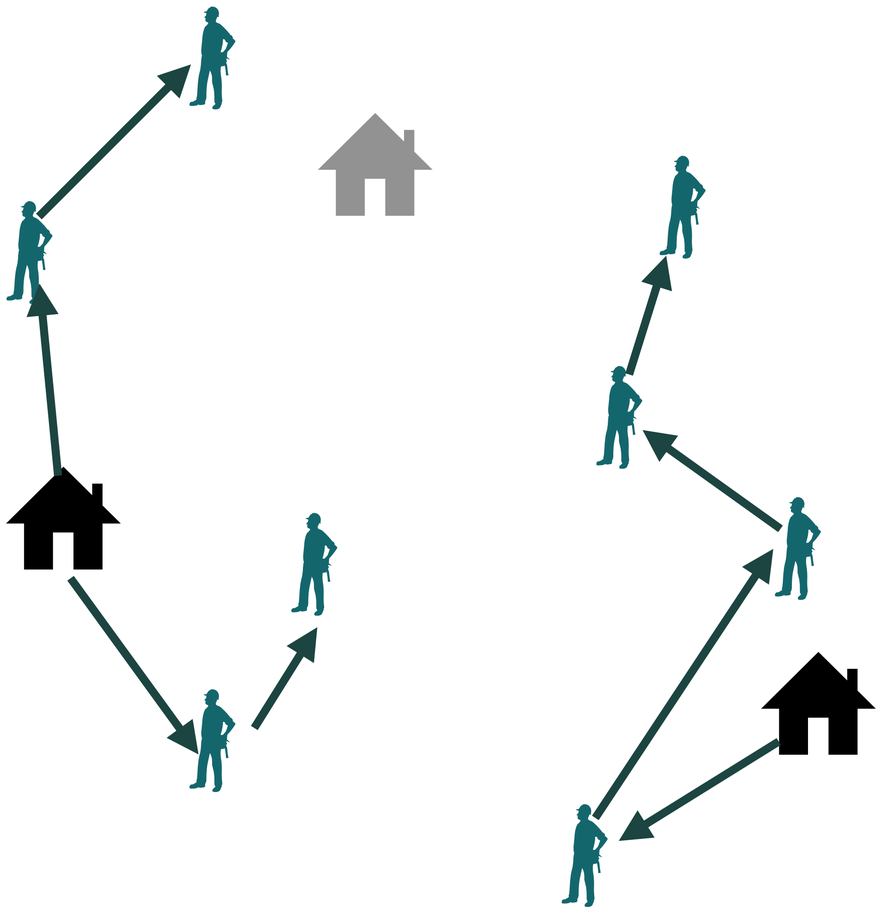
Covering: $\sum_{j,p,k} x_{ijpk} \geq 1 \qquad \forall i$

Opening: $\sum_i x_{ij1k} \leq y_j \qquad \forall j, k$

Sequence: $\sum_i x_{ijpk} \leq \sum_i x_{ijp-1k} \qquad \forall j, k, p > 1$

StartingArc: $z_{ji} \geq x_{ij1k} \qquad \forall i, j, k$

ArcSetup: $z_{uv} \geq x_{ujp-1k} + x_{vjpk} - 1 \qquad \forall u, v, k, p > 1$



https://atoptima.github.io/Coluna.jl/stable/start/advanced_demo/

TUTORIAL: Location routing

Original Formulation

```
# y[j] equals 1 if facility j is open; 0 otherwise.
@variable(model, y[j in facilities], Bin)

# z[u,v] equals 1 if a vehicle travels from u to v; 0 otherwise
@variable(model, z[u in locations, v in locations], Bin)

# x[i,j,k,p] equals 1 if customer i is delivered from facility j at position p of route k; 0 oth
@variable(model, x[i in customers, j in facilities, k in routes_per_facility, p in positions], B
```

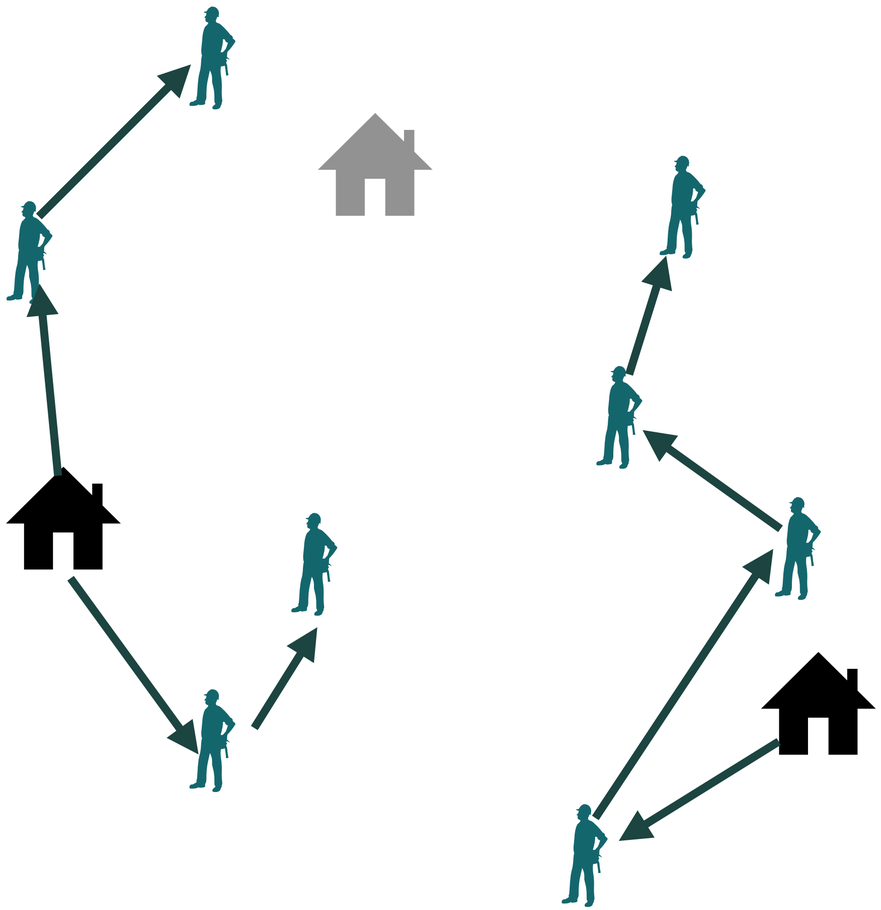
```
# each customer visited once
@constraint(model, cov[i in customers],
    sum(x[i, j, k, p] for j in facilities, k in routes_per_facility, p in positions) == 1)

# each facility is open if there is a route starting from it
@constraint(model, setup[j in facilities, k in routes_per_facility],
    sum(x[i, j, k, 1] for i in customers) <= y[j])

# flow conservation
@constraint(model, flow_conservation[j in facilities, k in routes_per_facility, p in positions;
    sum(x[i, j, k, p] for i in customers) <= sum(x[i, j, k, p-1] for i in customers))

# there is an arc between two customers whose demand is satisfied by the same route at consecut
@constraint(model, route_arc[i in customers, l in customers, j in facilities, k in routes_per_f
    z[i, l] >= x[l, j, k, p] + x[i, j, k, p-1] - 1)

# there is an arc between the facility `j` and the first customer visited by the route `k` from
@constraint(model, start_arc[i in customers, j in facilities, k in routes_per_facility],
    z[j, i] >= x[i, j, k, 1]);
```



TUTORIAL: Location routing

Original Formulation

Objective: $\min \sum_j f_j\, y_j + \sum_{u,v} c_{uv}\, z_{uv}$

Covering: $\sum_{j,p,k} x_{ijpk} \geq 1 \qquad \forall i$

Opening: $\sum_i x_{ij1k} \leq y_j \qquad \forall j, k$

Sequence: $\sum_i x_{ijpk} \leq \sum_i x_{ijp-1k} \qquad \forall j, k, p > 1$

StartingArc: $z_{ji} \geq x_{ij1k} \qquad \forall i, j, k$

ArcSetup: $z_{uv} \geq x_{ujp-1k} + x_{vjpk} - 1 \qquad \forall u, v, k, p > 1$

Dantzig-Wolfe Reformulation

Objective: $\min \sum_j f_j\, y_j + \sum_{j,q \in Q_j} (\sum_{u,v} c_{uv}\, z^q_{uv}) \lambda_{qj}$

Covering: $\sum_{j,q \in Q_j} (\sum_{j,p,k} x^q_{ijpk}) \lambda_{qj} \geq 1 \qquad \forall i$

Convexity: $\sum_{q \in Q_j} \lambda_{qj} \leq K\, y_j \qquad \forall j$

Dantzig-Wolfe Subproblem for j

Opening: $\sum_i x_{ij1} = 1$

Sequence: $\sum_i x_{ijp} \leq \sum_i x_{ijp-1} \qquad \forall p > 1$

StartingArc: $z_{ji} \geq x_{ij1} \qquad \forall i$

ArcSetup: $z_{uv} \geq x_{ujp-1} + x_{vjp} - 1 \qquad \forall u, v, p > 1$

TUTORIAL: Location routing

```
@axis(facilities_axis, collect(facilities))

# We declare a `BlockModel` instead of `Model`.
model = BlockModel(optimizer)
```

$$x_{ij} = \sum_p x_{ijp}$$

```
@variable(model, x[i in customers, j in facilities_axis], Bin)
```

```
@dantzig_wolfe_decomposition(model, dec, facilities_axis)
```

```
subproblems = BlockDecomposition.getsubproblems(dec)
specify!.(subproblems, lower_multiplicity=0, upper_mu

# We define `z` are a subproblem variable common to a
subproblemrepresentative.(z, Ref(subproblems))
```

Dantzig-Wolfe Reformulation

Objective: $\min \sum_j f_j\, y_j + \sum_{j,q \in Q_j} (\sum_{u,v} c_{uv}\, z^q_{uv}) \lambda_{qj}$

Covering: $\sum_{j,q \in Q_j} (\sum_{j,p,k} x^q_{ijpk}) \lambda_{qj} \geq 1 \qquad \forall i$

Convexity: $\sum_{q \in Q_j} \lambda_{qj} \leq K\, y_j \qquad \forall j$

Dantzig-Wolfe Subproblem for j

Opening: $\sum_i x_{ij1} = 1$

Sequence: $\sum_i x_{ijp} \leq \sum_i x_{ijp-1} \qquad \forall p > 1$

StartingArc: $z_{ji} \geq x_{ij1} \qquad \forall i$

ArcSetup: $z_{uv} \geq x_{ujp-1} + x_{vjp} - 1 \qquad \forall u, v, p > 1$

TUTORIAL: Location routing

Pricing callback

```
mutable struct Route
    length::Int # record
    path::Vector{Int} #
end;
```

```
routes_per_facility = Dict(
    j => best_route_forall_cust_subsets(arc_costs, customers, j, nb_positions) for j
)
```

```
function pricing_callback(cbdata)
    # Get the id of the facility.
    j = BlockDecomposition.indice(BlockDecomposition.callback_spid(cbdata, model))
```

```
# Keep route with minimum reduced cost.
red_costs_j = map(r -> (
        r,
        x_contribution(r, j, x_red_costs) + z_contribution(r, z_red_costs)
    ), routes_per_facility[j]
)
min_index = argmin([x for (_, x) in red_costs_j])
(best_route, min_reduced_cost) = red_costs_j[min_index]
```

```
# Submit the solution of the subproblem to Coluna.
MOI.submit(model, BlockDecomposition.PricingSolution(cbdata), sol_cost, sol_vars, sol_vals)
```

Dantzig-Wolfe Reformulation

Objective: $\min \sum_j f_j\, y_j + \sum_{j,q \in Q_j} (\sum_{u,v} c_{uv}\, z^q_{uv}) \lambda_{qj}$

Covering: $\sum_{j,q \in Q_j} (\sum_{j,p,k} x^q_{ijpk}) \lambda_{qj} \geq 1 \qquad \forall i$

Convexity: $\sum_{q \in Q_j} \lambda_{qj} \leq K\, y_j \qquad \forall j$

Dantzig-Wolfe Subproblem for j

Opening: $\sum_i x_{ij1} = 1$

Sequence: $\sum_i x_{ijp} \leq \sum_i x_{ijp-1} \qquad \forall p > 1$

StartingArc: $z_{ji} \geq x_{ij1} \qquad \forall i$

ArcSetup: $z_{uv} \geq x_{ujp-1} + x_{vjp} - 1 \qquad \forall u, v, p > 1$

TUTORIAL: Location routing

"robust" cuts $x_{ij} \leq y_j \; \forall i \in I, \forall j \in J$

```
struct OpenFacilityInequality
    facility_id::Int
    customer_id::Int
end
```

```
function valid_inequalities_callback(cbdata)
```

```
for j in facilities
    y_j = y_vals["y_$(j)"]
    for i in customers
        x_i_j = x_vals["x_$(i)_$(j)"]
        if x_i_j > y_j
            push!(inequalities, OpenFacilityInequality(j, i))
        end
    end
end
```

```
# Add the valid inequalities to the model.
for ineq in inequalities
    constr = JuMP.@build_constraint(x[ineq.customer_id, ineq.facility_id] <= y[ineq.facility
    MOI.submit(model, MOI.UserCut(cbdata), constr)
end
```

Dantzig-Wolfe Reformulation

Objective: $\min \sum_j f_j \, y_j + \sum_{j,q \in Q_j} (\sum_{u,v} c_{uv} \, z^q_{uv}) \lambda_{qj}$

Covering: $\sum_{j,q \in Q_j} (\sum_{j,p,k} x^q_{ijpk}) \lambda_{qj} \geq 1 \qquad \forall i$

Convexity: $\sum_{q \in Q_j} \lambda_{qj} \leq K \, y_j \qquad \forall j$

Dantzig-Wolfe Subproblem for j

Opening: $\sum_i x_{ij1} = 1$

Sequence: $\sum_i x_{ijp} \leq \sum_i x_{ijp-1} \qquad \forall p > 1$

StartingArc: $z_{ji} \geq x_{ij1} \qquad \forall i$

tup: $z_{uv} \geq x_{ujp-1} + x_{vjp} - 1 \qquad \forall u, v, p > 1$

TUTORIAL: Location routing

"non-robust" cuts $\sum_{k \in K} \lfloor \sum_{i \in C} \alpha_c \tilde{x}^k_{i,j} \lambda_k \rfloor \leq \lfloor \sum_{i \in C} \alpha_c \rfloor, \; C \subseteq I$

($|C|$ = 3) $\alpha = (0.5, 0.5, 0.5)$.

$$\sum_{k \in K} \tilde{\alpha}(C, k) \lambda_k \leq 1 \; C \subseteq I, |C| = 3$$

```
struct R1cVarData <: BlockDecomposition.AbstractCustomData
    visited_locations::Vector{Int}
end

struct R1cCutData <: BlockDecomposition.AbstractCustomData
    cov_constrs::Vector{Int}
end
```

```
MOI.submit(
    model, BlockDecomposition.PricingSolution(cbdata), sol_cost, sol_vars, sol_vals,
    R1cVarData(best_route.path)
```

```
function Coluna.MathProg.computecoeff(
    ::Coluna.MathProg.Variable, var_custom_data::R1cVarData,
    ::Coluna.MathProg.Constraint, constr_custom_data::R1cCutData
)
    return floor(1 / 2 * length(var_custom_data.visited_locations ∩ constr_custom_data.cov_const
end
```

```
function r1c_contrib(route::Route, custduals)
    cost = 0
    if !isempty(custduals)
        for (r1c_cov_constrs, dual) in custduals
            coeff = floor(1 / 2 * length(route.path ∩ r1c_cov_constrs))
            cost += coeff * dual
        end
    end
    return cost
end;
```

TUTORIAL: Location routing

rank-one cut callback:

```
lambdas = Tuple{Float64,Coluna.MathProg.Variable}[]
for (var_id, val) in original_sol
    if Coluna.MathProg.getduty(var_id) <= Coluna.MathProg.MasterCol
        push!(lambdas, (val, Coluna.MathProg.getvar(cbdata.form, var_id)))
    end
end
```

```
for cov_constr_subset in collect(combinations(cov_constrs, 3))
    lhs = 0
    for lambda in lambdas
        (val, var) = lambda
        if !isnothing(var.custom_data)
            coeff = floor(1 / 2 * length(var.custom_data.visited_locations ∩ cov_constr_sub
            lhs += coeff * val
        end
    end
    if lhs > 1
        # Create the constraint and add it to the model.
        MOI.submit(model,
            MOI.UserCut(cbdata),
            JuMP.ScalarConstraint(JuMP.AffExpr(0.0), MOI.LessThan(1.0)),
            R1cCutData(cov_constr_subset)
        )
    end
```

TUTORIAL: Location routing

Original Formulation

Objective: $\min \sum_j f_j\, y_j + \sum_{u,v} c_{uv}\, z_{uv}$

Covering: $\sum_{j,p,k} x_{ijpk} \geq 1 \qquad \forall i$

Opening: $\sum_i x_{ij1k} \leq y_j \qquad \forall j, k$

Sequence: $\sum_i x_{ijpk} \leq \sum_i x_{ijp-1k} \qquad \forall j, k, p > 1$

StartingArc: $z_{ji} \geq x_{ij1k} \qquad \forall i, j, k$

ArcSetup: $z_{uv} \geq x_{ujp-1k} + x_{vjpk} - 1 \qquad \forall u, v, k, p > 1$

Benders Reformulation

Objective: $\min \sum_j f_j\, y_j + \eta$

Benders-Cut: $\sum_j \pi_j y_j + \pi_0 \eta \geq 1 \qquad \forall \pi$

Benders Convexified Subproblem

SecondStageCost: $\eta \geq \sum_{j,q \in Q_j} (\sum_{u,v} c_{uv}\, z^q_{uv}) \lambda_{qj}$

Covering: $\sum_{j,q \in Q_j} (\sum_{j,p,k} x^q_{ijpk}) \lambda_{qj} \geq 1 \qquad \forall i$

Convexity: $\sum_{q \in Q_j} \lambda_{qj} \leq K\, y_j \qquad \forall j$

TUTORIAL: Location routing

```
fake = 1
@axis(axis, collect(fake:fake))

coluna = JuMP.optimizer_with_attributes(
    Coluna.Optimizer,
    "params" => Coluna.Params(solver=Coluna.Algorithm.BendersCutGeneration()
    ),
    "default_optimizer" => GLPK.Optimizer
)

model = BlockModel(coluna);
```

```
@benders_decomposition(model, dec, axis)
```

Benders Reformulation

Objective: $\min \sum_j f_j\, y_j + \eta$

Benders-Cut: $\sum_j \pi_j y_j + \pi_0 \eta \geq 1 \qquad \forall \pi$

Benders Convexified Subproblem

SecondStageCost: $\eta \geq \sum_{j,q \in Q_j} (\sum_{u,v} c_{uv}\, z^q_{uv}) \lambda_{qj}$

Covering: $\sum_{j,q \in Q_j} (\sum_{j,p,k} x^q_{ijpk}) \lambda_{qj} \geq 1 \qquad \forall i$

Convexity: $\sum_{q \in Q_j} \lambda_{qj} \leq K\, y_j \qquad \forall j$

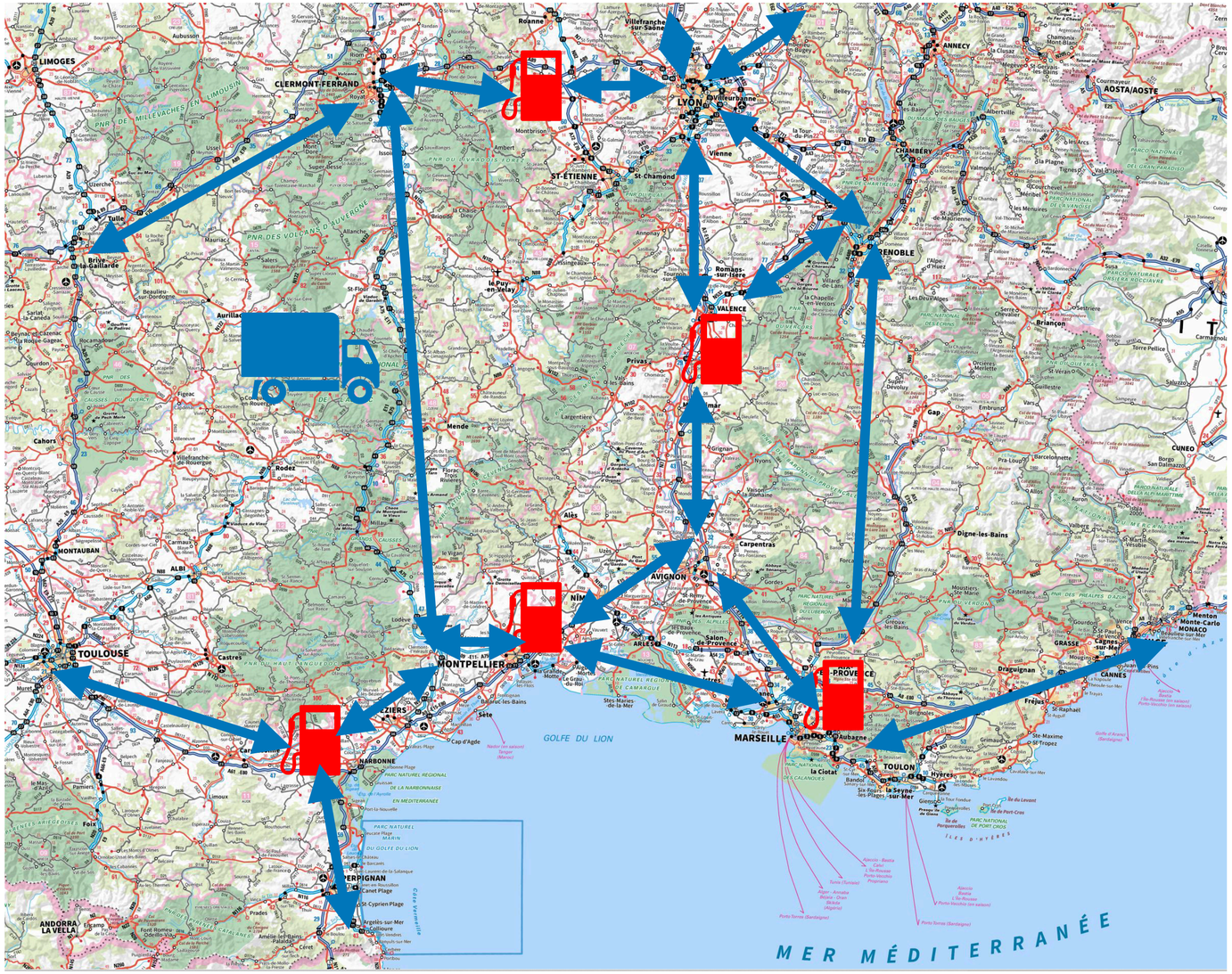
```
@variable(model, 0 <= y[j in facilities] <= 1) ## 1st stage
@variable(model, 0 <= λ[f in axis, j in facilities, k in 1:length(routes_per_facility[j])] <= 1)
```

```
# Linking constraints
@constraint(model, open[fake in axis, j in facilities, k in 1:length(routes_per_facility[j])],
    y[j] >= λ[fake, j, k])
```

Use Case of Location Routing

CMA CGM

1. Charging station location placement
2. Assigning electric vehicles to depots
3. Routing clean vehicles to cover transportation requests



atoptima
OPTIMIZATION INTELLIGENCE