Coluna: An Open-Source Branch-Cut-and-Price Framework

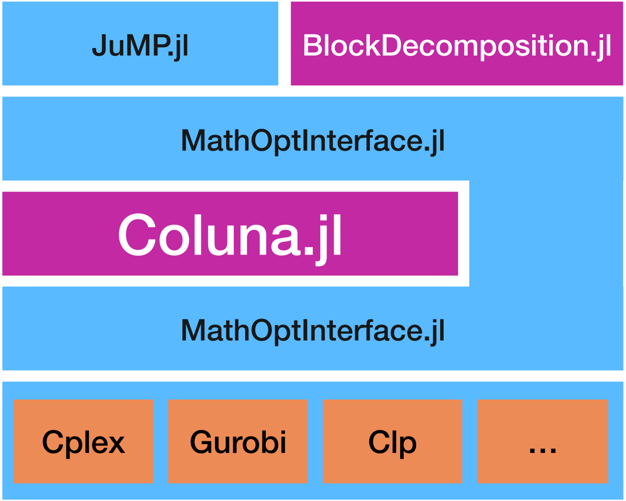
T. Bulhões, G. Marques, V. Nesello, A. Pessoa, E. Uchoa, R. Sadykov, I. Tahiri, F. Vanderbeck

JuMP-dev 2019, Santiago
March 2019

Contents

Contents

What is Coluna.jl?



Coluna.jl

- **What:**
  - A **Column-and-Row** generation code
  - Open-Source: **MPLicence**
  - Written in **Julia** (1.1)
  - Uses **JuMP.jl** (0.19), **MathOptInteface.jl** and **BlockDecomposition.jl**
- **Uses:**
  - **Dantzig-Wolfe & Benders decomposition**
  - **Robust & Stochastic Optimization**
  - **Machine Learning**
- **First release:** **Jan. ’19**
- **Support:** **Math Optimization Society (MOS)** & **JuliaOpt**
- **URL** `https://github.com/atoptima/Coluna.jl`

Contents

Decomposition Approaches

- On a **subset of constraints** (**Dantzig-Wolfe**): **multi-resources**

$$
\begin{array}{lccccccccl}
\min & c^1\, x^1 & + & c^2\, x^2 & + & \dots & + & c^K\, x^K & & \\
 & A\, x^1 & + & A\, x^2 & + & \dots & + & A\, x^K & \geq a \\
 & B^1\, x^1 & & & & & & & \geq b^1 \\
 & & & B^2\, x^2 & & & & & \geq b^2 \\
 & & & & & \dots & & & \geq \dots \\
 & & & & & & & B^K\, x^K & \geq b^K
\end{array}
$$

- On a **subset of variables** (**Benders**): **multi-decision-levels**

$$
\begin{array}{lccccccccl}
\min & a\, x & + & b^1\, y^1 & + & b^2\, y^2 & + & \dots & + & b^K\, y^K & \\
 & A\, x & + & B^1\, y^1 & & & & & & & \geq c^1 \\
 & A\, x & + & & & B^2\, y^2 & & & & & \geq c^2 \\
 & A\, x & + & & & & & \dots & & & \geq \dots \\
 & A\, x & + & & & & & & & B^K\, y^K & \geq c^K
\end{array}
$$

The Cutting Stock Problem (CSP): Block-diagonal structure

Number of distinct cutting patterns : 17
Number of rolls : 88
Total waste : 5090

$$
\begin{aligned}
\min \quad & \sum_{k=1}^{K} y_k \\
\text{s.t.} \quad & \sum_{k=1}^{K} x_{ik} \geq d_i \quad \forall i \\
& \sum_{i} w_i \, x_{ik} \leq W \, y_k \quad \forall k \\
& x_{ik} \in \mathbb{N} \quad \forall i, k \\
& y_k \in \{0, 1\} \quad \forall k
\end{aligned}
$$

Dantzig-Wolfe Decomposition

Assume a bounded integer integer problem with structure:

$$
\begin{array}{rrcll}
[\mathrm{F}] \equiv \min & c\,x & : & & \\
& A\,x & \geq & a & \\
x \in X = \{ & B\,x & \geq & b & \\
& x & \in & \mathbb{N}^n & \}
\end{array}
$$

Assume that subproblem

$$[\mathrm{SP}] \equiv \min\{c\,x : \; x \in X\}$$

is "relatively easy" to solve compared to problem [F].

Dantzig-Wolfe Decomposition

Assume a bounded integer integer problem with structure:

$$
\begin{array}{rrcll}
[\mathrm{F}] \equiv \min & c\,x & : & & \\
 & A\,x & \geq & a & \\
x \in X = \{ & B\,x & \geq & b & \\
 & x & \in & \mathbb{N}^n & \}
\end{array}
$$

Assume that subproblem

$$[\mathrm{SP}] \equiv \min\{c\,x : \; x \in X\}$$

is <u>“relatively easy”</u> to solve compared to problem [F]. Then,

$$X \quad = \quad \{x^q\}_{q \in Q}$$

Dantzig-Wolfe Decomposition

Assume a bounded integer integer problem with structure:

$$
\begin{array}{rrcll}
[\mathrm{F}] \equiv \min & c\,x & : & & \\
 & A\,x & \geq & a & \\
x \in X = \{ & B\,x & \geq & b & \\
 & x & \in & \mathbb{N}^n & \}
\end{array}
$$

Assume that subproblem

$$[\mathrm{SP}] \equiv \min\{c\,x : \; x \in X\}$$

is "relatively easy" to solve compared to problem [F]. Then,

$$
\begin{array}{rcl}
X & = & \{x^q\}_{q \in Q} \\
\mathrm{conv}(X) & = & \{x \in \mathbb{R}^n_+ : x = \sum_{q \in Q} x^q \lambda_q, \; \sum_{q \in Q} \lambda_q = 1, \; \lambda_q \geq 0 \; q \in Q\}
\end{array}
$$

Dantzig-Wolfe Decomposition

Assume a bounded integer integer problem with structure:

$$\begin{array}{rrcl} [F] \equiv \min & c\,x & : & \\ & A\,x & \geq & a \\ x \in X = \{ & B\,x & \geq & b \\ & x & \in & \mathbb{N}^n \quad \} \end{array}$$

Assume that subproblem

$$[SP] \equiv \min\{c\,x : \; x \in X\}$$

is "relatively easy" to solve compared to problem [F]. Then,

$$\begin{array}{rcl} X & = & \{x^q\}_{q \in Q} \\ \mathrm{conv}(X) & = & \{x \in \mathbb{R}^n_+ : x = \sum_{q \in Q} x^q \lambda_q, \; \sum_{q \in Q} \lambda_q = 1, \; \lambda_q \geq 0 \; q \in Q\} \\ [F] & \equiv & \min\{\sum_{q \in Q} c x^q \lambda_q : \; \sum_{q \in Q} A x^q \lambda_q \geq a, \; \sum_{q \in Q} \lambda_q = 1, \; \sum_q x^q \lambda_q \in \mathbb{N}^n\} \end{array}$$

The Cutting Stock Problem (CSP): Decomposition

Number of distinct cutting patterns : 17
Number of rolls : 88
Total waste : 5090

$$\min \sum_{k=1}^{K} y_k$$

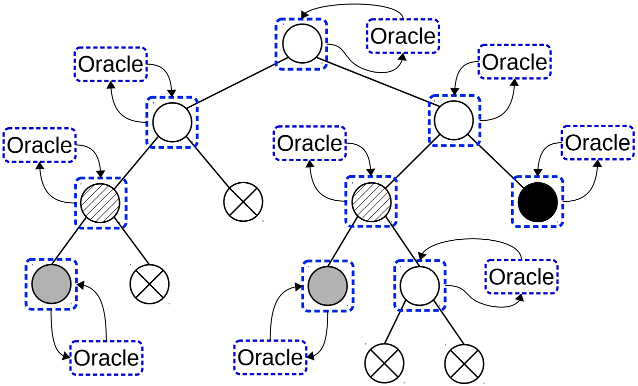
$$\begin{aligned}
\text{s.t.} \quad \sum_{k=1}^{K} x_{ik} &\geq d_i \quad \forall i \\
\sum_{i} w_i \, x_{ik} &\leq W \, y_k \quad \forall k \\
x_{ik} &\in \mathbb{N} \quad \forall i, k \\
y_k &\in \{0, 1\} \quad \forall k
\end{aligned}$$

$$\min \sum_{q} \lambda_q$$

$$\begin{aligned}
\sum_{q} x_i^q \, \lambda_q &\geq d_i \qquad i = 1, \cdots, n \\
\lambda_q &\in \mathbb{N} \qquad \forall q
\end{aligned}$$

Solving decomposed problems

- Choose node
- Solve chosen node
  - Solve master lp
  - Check if converged
  - Generate row/columns
  - Repeat
- Create children nodes



Coluna log : Solving a node

```
************************************************************
Preparing node 6 for treatment. Parent is 5.
Elapsed time: 174.96061992645264 seconds.
Current primal bound is 1931.999999999998
Subtree dual bound is 1929.8571428571377
Branching constraint:  x[3,76] >= 1.0
2 open nodes. Treating node 6.
Current best known bounds : [ 1929.8571428571377 , 1931.999999999998 ]
************************************************************
<it=1> <et=175.0> <mst= 0.021> <sp= 0.206> <cols=5> <mlp=1930.4839> <DB=1921.7871> <PB=1932.0>
<it=2> <et=175.0> <mst= 0.015> <sp= 0.234> <cols=5> <mlp=1930.25> <DB=1925.4306> <PB=1932.0>
<it=3> <et=176.0> <mst= 0.015> <sp= 0.264> <cols=5> <mlp=1930.25> <DB=1925.8393> <PB=1932.0>
<it=4> <et=176.0> <mst= 0.015> <sp= 0.242> <cols=5> <mlp=1930.25> <DB=1926.4267> <PB=1932.0>
<it=5> <et=176.0> <mst= 0.011> <sp= 0.27> <cols=5> <mlp=1930.25> <DB=1926.7257> <PB=1932.0>
<it=6> <et=176.0> <mst= 0.014> <sp= 0.206> <cols=5> <mlp=1930.25> <DB=1927.9531> <PB=1932.0>
<it=7> <et=177.0> <mst= 0.014> <sp= 0.266> <cols=5> <mlp=1930.25> <DB=1928.8538> <PB=1932.0>
                                          . . .
<it=14> <et=179.0> <mst= 0.017> <sp= 0.288> <cols=5> <mlp=1930.25> <DB=1929.7722> <PB=1932.0>
<it=15> <et=179.0> <mst= 0.014> <sp= 0.325> <cols=5> <mlp=1930.25> <DB=1930.0> <PB=1932.0>
<it=16> <et=180.0> <mst= 0.01> <sp= 0.347> <cols=5> <mlp=1930.25> <DB=1930.125> <PB=1932.0>
<it=17> <et=180.0> <mst= 0.014> <sp= 0.376> <cols=5> <mlp=1930.25> <DB=1930.25> <PB=1932.0>
<Coluna._AlgToPrimalHeurByRestrictedMip> <mlp=1930.2499999999993> <PB=1960.0000000000005>
[ Info: Generated 2 child nodes.
```

Interest

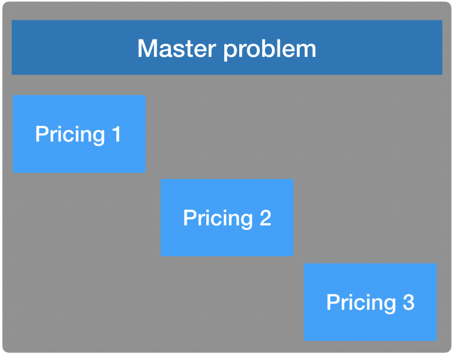
Integer Programming Decomposition

- A powerful **way** to exploit the combinatorial structure.
- **Benders & Dantzig Wolfe decomposition** are generic schemes to derive & handle **strong** reformulations
- **Size** can be coped with using **dynamic generation**: a **small %** of **variables** and **constraints** are needed; hence it scales up to real-life applications.
- With efficiency enhancement features, these approaches can be highly **competitive**
- Implementation can be generic, with tools such as **Coluna.jl**

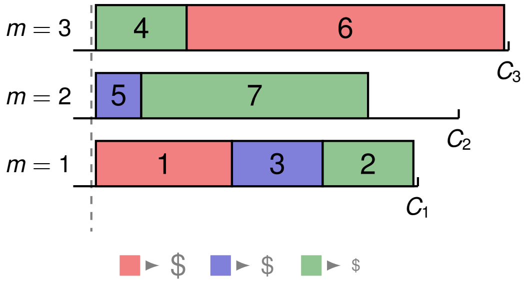
Contents

General idea

- Write a model using JuMP
- Use variable and constraint indices to define the decomposition
- Store extra information in the indices -> `axis` macro
- Create a decomposition tree based on the memberships defined in the extra information -> `dantzig_wolfe_decomposition` macro



Generalized Assignment Problem : The simple case



$$\min \sum_{jm} c_{jm}\, x_{jm}$$

$$\text{s.t.} \quad \sum_{m=1}^{M} x_{jm} \geq 1 \quad \forall j$$

$$\sum_{j} w_{jm}\, x_{jm} \leq C_m \quad \forall m$$

$$x_{jm} \in \{0,1\} \quad \forall j, m$$

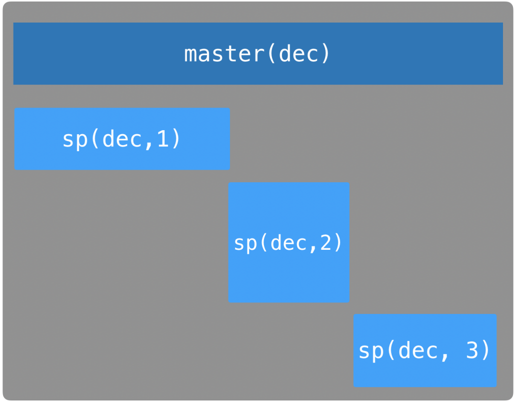
$$\min \sum_{q} c_q\, \lambda_q$$

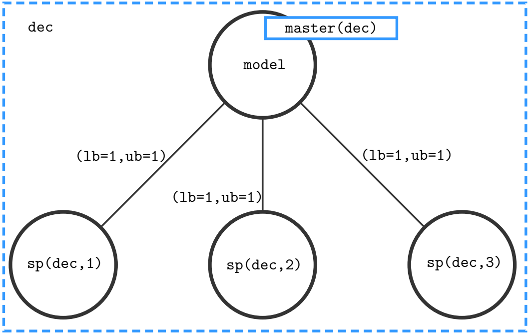
$$\sum_{m \in M} \sum_{q \in Q^m} x_j^q\, \lambda_q \geq 1 \quad \forall j$$

$$\sum_{q \in Q^m} \lambda_q = 1 \quad \forall m$$

$$\lambda_q \in \{0,1\} \quad \forall q$$

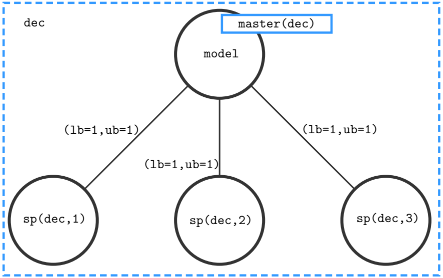
Generalized Assignment Problem : The simple case



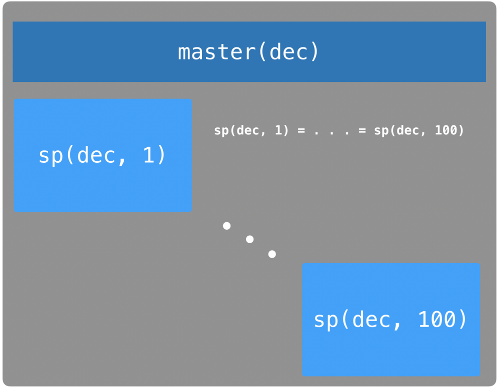


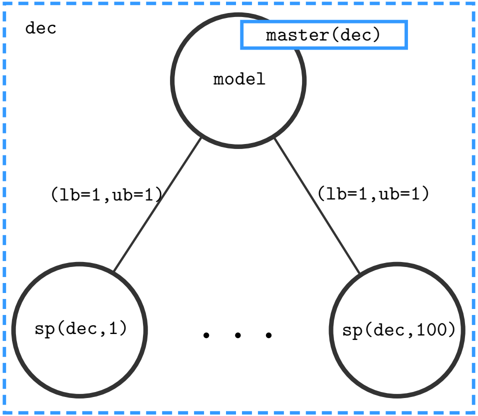
Generalized Assignment Problem : The simple case

```julia
function generalized_assignment()
    @axis(Machines, 1:3)

    model = BlockModel() # Defines a JuMP model with a hook
    @variable(model, x[j in jobs, m in Machines], Bin)
    @constraint(
        model, cov[j in jobs],
        sum(x[j, m] for m in Machines) >= 1
    )
    @constraint(
        model, knp[m in Machines],
        sum(weights[j, m] * x[j, m] for j in jobs)
        <= capacities[m]
    )
    @objective(
        model, Min,
        sum(costs[j, m] * x[j, m] for j in jobs, m in Machines)
    )

    @dantzig_wolfe_decomposition(model, dec, Machines)
end
```

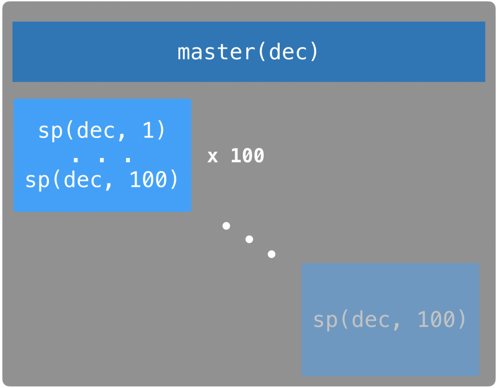


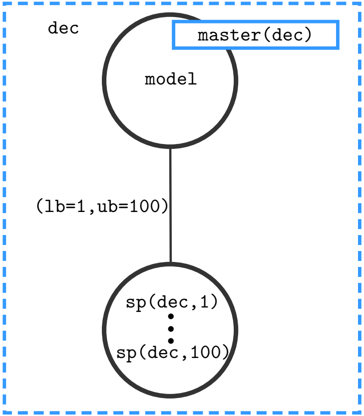
Cutting Stock : Different subproblems





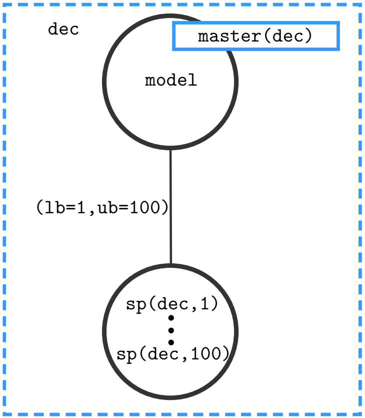
Cutting Stock : Different subproblems





Cutting Stock : Identical subproblems

```
function cutting_stock()
    @axis(Patterns, 1:100, Identical)

    model = BlockModel()
    @variable(
        model, 0 <=
        x[i in items, s in Patterns] <= demands[i], Int
    )
    @variable(model, y[s in Patterns], Bin)
    @constraint(
        model, cov[i in items],
        sum(x[i, s] for s in Patterns)
        >= demands[i]
    )
    @constraint(
        model, knp[s in Patterns],
        sum(widths[i] * x[i, s] for i in items)
        <= sheets_sizes[1] * y[s]
    )
    @objective(model, Min, sum(y[s] for s in Patterns))

    @dantzig_wolfe_decomposition(model, dec, Patterns)
end
```

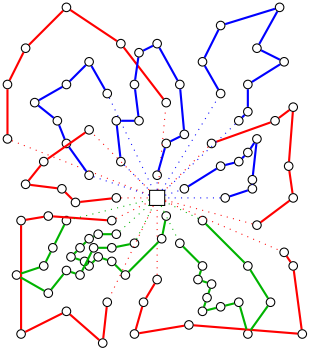


Cutting Stock : Getting the solution

```julia
function cutting_stock()
    @axis(Patterns, 1:100, Identical)

    model = BlockModel()
    @variable(
        model, 0 <=
        x[i in items, s in Patterns] <= demands[i], Int
    )
    @variable(model, y[s in Patterns], Bin)
    @constraint(
        model, cov[i in items],
        sum(x[i, s] for s in Patterns)
        >= demands[i]
    )
    @constraint(
        model, knp[s in Patterns],
        sum(widths[i] * x[i, s] for i in items)
        <= sheets_sizes[1] * y[s]
    )
    @objective(model, Min, sum(y[s] for s in Patterns))

    @dantzig_wolfe_decomposition(model, dec, Patterns)
end
```

```julia
for s in active_indices(Patterns)
    for i in items
        @show value(x[s, i])
    end
end
```

CVRP : Expression & ad-hoc pricing solver



$$
\begin{array}{llccl}
\min & \sum_{e,r} c_e x_e^r & & & \\
\text{s.t.} & \sum_{e \in \delta(i), r} x_e^r & \geq & 2 & \forall i \\
& x^r & \in & X^r & \forall r
\end{array}
$$

$$
\begin{array}{llccl}
\min & \sum_{e,r,q \in Q^r} c_e x_e^q \lambda_q & & & \\
& \sum_{e \in \delta(i), r, q \in Q^r} x_e^q \lambda_q & \geq & 2 & \forall i \\
& \sum_{q \in Q^r} \lambda_q & \leq & U^r & \forall r \\
& \lambda_q & \in & \{0, 1\} & \forall r, q \in Q^r
\end{array}
$$

CVRP : Ad-hoc pricing solver

```
function cvrp()
    G = GraphOfInstance()

    @axis(model, Routes, 1:max_routes, Identical)

    @variable(model, x[r in Routes, e in edges(G)] >= 0)
    @expression(model, y[e in edges(G)], sum(x[r,e] for r in Routes), Int)
    @constraint(model, deg[i in nodes(G)], sum(x[r,e] for e in incidents(G, i), r in Routes) == 2.0)
    @objective(model, Min, sum(c[e] * y[e] for e in edges(G)))

    @dantzig_wolfe_decomposition(model, dec, Routes)

    build_network(model, G, sp(dec, 1))
    add_pricing_callback(model, dec, PricingCallBack::Function)
end
```

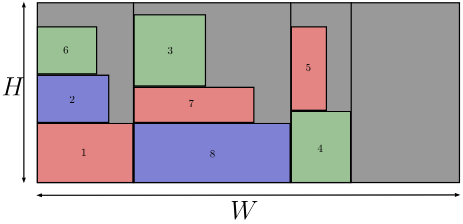
CVRP : Ad-hoc pricing solver

```
function build_network(subproblem)
    r = getsubproblemid(subproblem)
    net = Network(nb_nodes, source = 1, sink = nb_nodes)
    capacity_res = addmainresource!(net, max_capacity, stepsize = 1)
    for e in edges(G)
        edge_id = add_edge!(net, e, var = x[r,e])
            edgeconsumption!(net, edge_id, capacity_res, value = conso[e])
    end
    return net
end
```

CVRP : Ad-hoc pricing solver

```julia
function pricing_callback(cb)
    r = getsubproblemid(cb)
    reduced_costs = [getreducedcost(cb, x[r, e]) for e in edges(G)]
    solution_vectors = compute_shortest_qroute(G, reduced_costs)
    for solution in solution_vectors
        solution_reduced_cost = getreducedcost(cb, getsubproblem(dec, r))
        path_x_vector = zeros(Int, length(edges(G)))
        for edge in solution
            path_x_vector[edge] += 1
            solution_reduced_cost += reduced_costs[edge]
        end
        if solution_reduced_cost < - 0.001
            for e in edges(G)
                x[r, e] = path_x_vector[e]
            end
                recordsubproblemsolution(cb, getsubproblem(dec, r), x[r, :])
        end
    end
end
```

2D Cutting Stock : Nested decomposition



$$
\begin{array}{lrcll}
\min & \sum_{s \in S} z_s & & & \\
\text{s.t.} & \sum_{s \in S, p \in P} x_{jps} & \geq & 1 & \forall j \\
& \sum_{p} w_{ps} & \leq & W\, z_s & \forall s \\
& w_{ps} & \geq & w_j\, x_{jps} & \forall j, p, s \\
& \sum_{j} h_j\, x_{jps} & \leq & H\, y_{ps} & \forall p, s \\
& x_{jps} & \in & \{0, 1\} & \forall j, p, s \\
& y_{ps} & \in & \{0, 1\} & \forall p, s \\
& z_s & \in & \{0, 1\} & \forall s
\end{array}
$$

2D Cutting Stock : Original & master reformulations

$$
\begin{array}{rrcll}
\min & \sum_{s \in S} z_s & & & \\
\text{s.t.} & \sum_{s \in S, p \in P} x_{jps} & \geq & 1 & \forall j \\
& \sum_{p} w_{ps} & \leq & W\, z_s & \forall s \\
& w_{ps} & \geq & w_j\, x_{jps} & \forall j, p, s \\
& \sum_{j} h_j\, x_{jps} & \leq & H\, y_{ps} & \forall p, s \\
& x_{jps} & \in & \{0, 1\} & \forall j, p, s \\
& y_{ps} & \in & \{0, 1\} & \forall p, s \\
& z_s & \in & \{0, 1\} & \forall s
\end{array}
$$

$$
\begin{array}{rcll}
\min \sum_{s} \lambda_s & & & \\
\sum_{s} x_j^s\, \lambda_s & \geq & 1 & \forall j \\
\lambda_s & \in & \mathbb{N} & \forall s
\end{array}
$$

2D Cutting Stock : Pricing problem and its reformulation

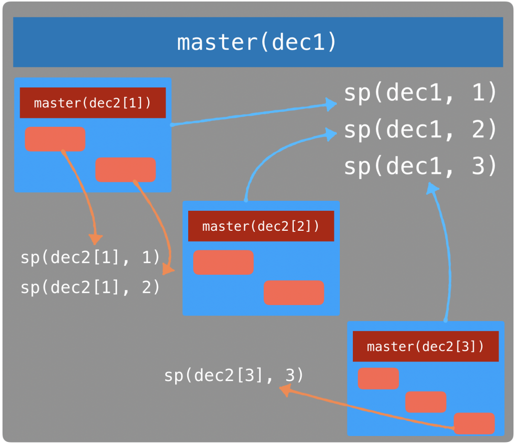
$$\min 1 - \left(\sum_j \pi_j \left(\sum_p x_{jp}\right)\right)$$

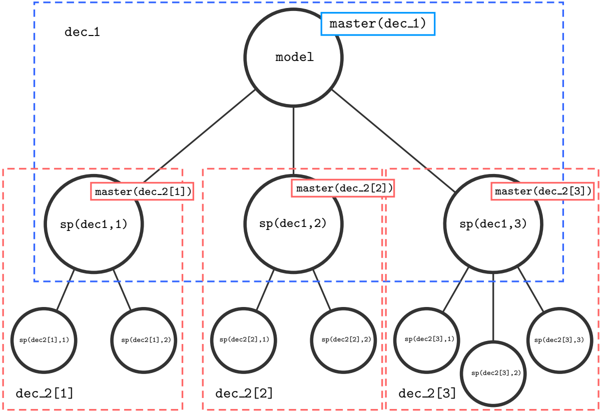
$$\begin{array}{lrcll}
\text{s.t.} & \sum_p x_{jp} & \leq & 1 & \forall j \\
& \sum_p w_p & \leq & W & \forall \\
& w_p & \geq & w_j\, x_{jp} & \forall j, p \\
& \sum_j h_j\, x_{jp} & \leq & H & \forall p \\
& x_{jp} & \in & \{0, 1\} & \forall j, p
\end{array}$$

$$\min \sum_p \lambda_p$$

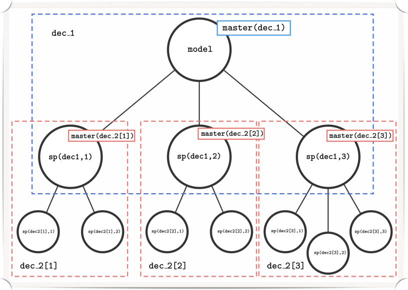
$$\begin{array}{rcll}
\sum_p x_j^p\, \lambda_p & \leq & 1 & \forall j \\
\sum_p w_p\, \lambda_p & \leq & W & \\
\lambda_p & \in & \mathbb{N} & \forall p
\end{array}$$

2D Cutting Stock : Nested decomposition





2D Cutting Stock : Nested decomposition




```
function 2D_cutting_stock()
    @axis(model, S, 1:3, Identical) # Defines Sheets
    @axis(model, P[s in S], 1:4, Identical) # Defines Patterns

    @variable(model, 0 <= z[s in S] <= 1, Bin)
    @variable(model, 0 <= y[s in S, p in P[s]] <= 1, Bin)

    @variable(model, 0 <= w[s in S, p in P[s]] <= 1)
    @variable(model, 0 <= x[j in items, p in P[s], s in S] <= 1, Bin)

    @constraint(model, cov[j in items],sum(x[j, p, s] for s in S, p in P[s]) >= 1)
    @constraint(model, sheet_cap[s in S], sum(w[s, p] for p in P[s]) <= z[s] * sheet_width)
    @constraint(model, strip_width[s in S, p in P[s], j in items], w[p, s] >= x[j, p, s] * items[j].width)
    @constraint(model, strip_cap[s in S, p in P[s]],
                sum(x[j, p, s] * items[j].height for j in items) <= y[p, s] * sheet_height)

    @objective(model, Min, sum(z[s] for s in S))

    @dantzig_wolfe_decomposition(model, dec1, S)
    @dantzig_wolfe_decomposition(sp(dec1, s), dec2[s in S], P[s])
end
```

Coluna: An Open-Source Branch-Cut-and-Price Framework

T. Bulhões, G. Marques, V. Nesello, A. Pessoa, E. Uchoa, R. Sadykov, I. Tahiri, F. Vanderbeck

JuMP-dev 2019, Santiago
March 2019

https://github.com/atoptima/Coluna.jl