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

C. Bentes, T. Bulhoes, H. Kramer, G. Marques, V. Nesello, A. Pessoa, M. Poss, A. Subramanian, M. Tanneau, R. Sadykov,

E. Uchoa, F. Vanderbeck

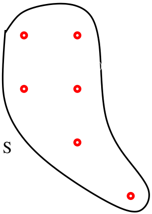
November 2019

Integer Programming Formulation

Combinatorial Optimization Problem

$$(CO) \equiv \min\{c(s) : \; s \in \mathbf{S}\}$$

where $\mathbf{S}$ is the **discrete** set of feasible solutions.



Integer Programming Formulation

Combinatorial Optimization Problem

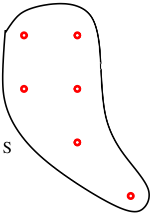
$$(CO) \equiv \min\{c(s) : \; s \in \mathbf{S}\}$$

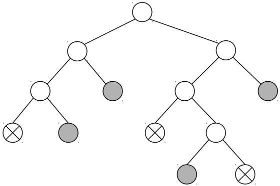
where $\mathbf{S}$ is the **discrete** set of feasible solutions.

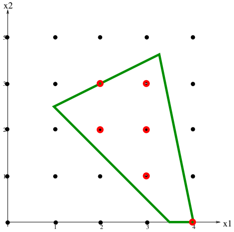
Integer Programming Formulation

A **polyhedron** $\mathbf{P} = \{x \in \mathbb{R}^n : Ax \geq a\}$ is a formulation for (CO) iff

$$\min\{c(s) : \; s \in \mathbf{S}\} \quad \equiv \quad \min\{cx : x \in \mathbf{P_I} = \mathbf{P} \cap \mathbb{N}^{\mathbf{n}}\}.$$







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**

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

Assume a bounded integer integer problem with structure:

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

Assume that subproblem

$$[\mathrm{SP}] \equiv \min\{c\,x : \; x \in X\} \tag{1}$$

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

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

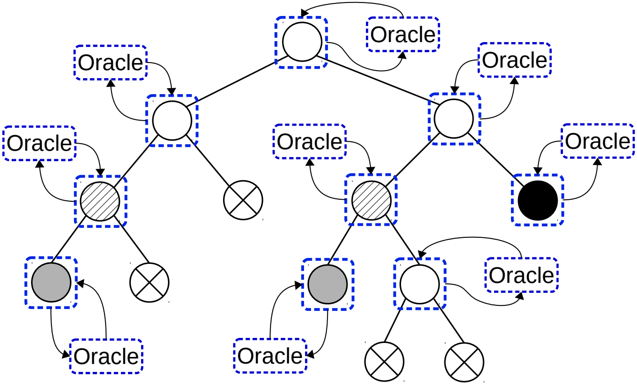
The Cutting Stock Problem (CSP)

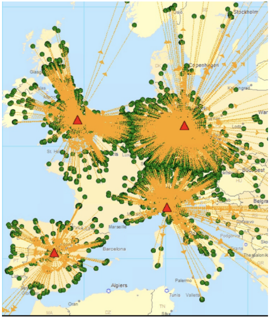
Number of distinct cutting patterns : 17
Number of rolls : 88
Total waste : 5090

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

$$
\begin{aligned}
\min & \sum_{q} \lambda_q \\
\sum_{q} x_i^q \, \lambda_q & \geq d_i \qquad i = 1, \cdots, n \\
\lambda_q & \geq 0 \qquad \forall q
\end{aligned}
$$

Solving decomposed problems





$$
\begin{array}{llllllllll}
\max & a\,x & + & b^1\,y^1 & + & b^2\,y^2 & + & \dots & + & b^K\,y^K \\
 & A\,x & + & B^1\,y^1 & & & & & & \\
 & A\,x & + & & & B^2\,y^2 & & & & \\
 & A\,x & + & & & & & \dots & & \\
 & A\,x & + & & & & & & & B^K\,y^K
\end{array}
$$

$$
\begin{array}{rcl}
\max\,\{\, a\,x + b\,y & : & \\
A\,x + B\,y & \leq & c\,, \\
x \in \mathbb{N}^n\,,\; y & \in & \mathbb{R}^p_+ \},
\end{array}
$$

Benders Decomposition & Cut Generation Reformulation

$$\max \{ a\,x + b\,y : \; A\,x + B\,y \le c, \; x \in \mathbb{N}^n, \; y \in \mathbb{R}^p_+ \},$$

$$\phi(x) := \max \{ b\,y : B\,y \le c - A\,x, \; y \in \mathbb{R}^p_+ \}$$

First stage:

$$\max_{x,\eta} \{ \eta : \; \eta \le a\,x + \phi(x), \; x \in \mathbb{N}^n, \eta \in \mathbb{R} \}, \quad \Leftrightarrow \quad u_0^q\,(\eta - a\,x) + u^q\,(A\,x - c) \le 0 \;\; \forall q$$

Second stage:

$$\begin{aligned} \min\ \nu &\le 0\ ? \\ b\,y + \nu &\ge (\eta - a\,x) \\ -B\,y + 1\,\nu &\ge (A\,x - c) \\ (y, \nu) &\in \mathbb{R}^{p+1}_+ \end{aligned} \qquad \begin{aligned} \max\ u_0\,(\eta - a\,x) \;+\; u\,(A\,x - c) &\le 0\ ? \\ u_0 b - u\,B &\le 0 \\ u_0 + 1x &\le 1 \\ (u_0, u) &\in \mathbb{R}^{1+m_2}_+ \end{aligned}$$

$\{u^1, \ldots, u^T\}$ are its **extreme dual solutions**.

Integer Programming Decomposition & Reformulations

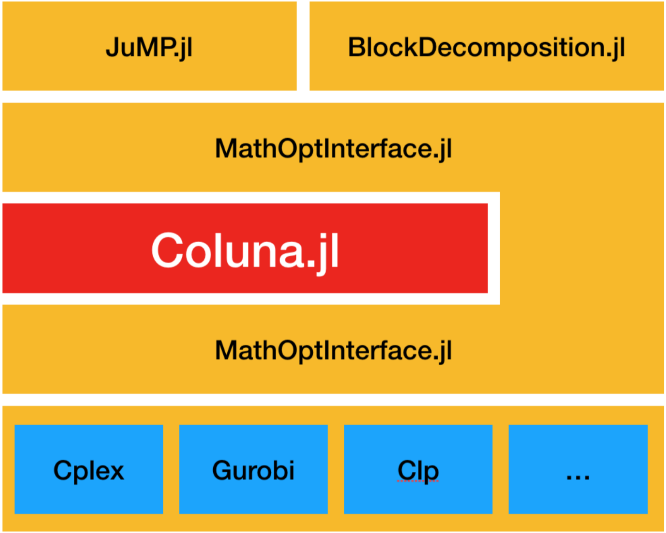
- A powerful **way** to exploit the combinatorial structure.
- **Benders & Dantzig Wolfe decomposition** are generic schemes to derive & handle **quality** reformulations
- **Size** can be coped with using **dynamic generation**: a **small %** of **variables** and **constraints** are needed; hence it 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**

Coluna.jl 0.2

- **What:**
  - a **Column-and-Row** generation code
  - Open-Source: **Mozilla Public Licence - MPL**
  - in **Julia 1.2**
  - Builds on **JuMP-JuliaOpt** extending it with **BlockDecomposition.jl**
- **Uses:**
  - **Dantzig-Wolfe & Benders decomposition**
  - **Extended Formulation Approaches**
  - **Robust & Stochastic Optimization**
  - **Machine Learning**
- **Support: Math Optimization Society (MOS)** & **JuliaOpt**
- **URL**

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

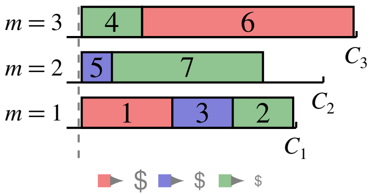
Coluna Environment



OUTLINE

- **Modeling**
- **Data Structure**
- **Algorithms**
- **Current Features**
- **Features to come**
- **Installation**

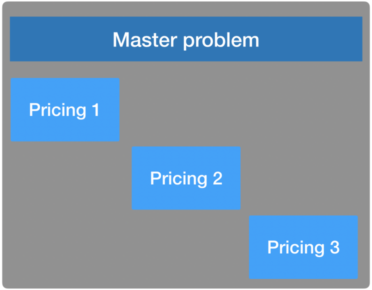
Exemple : Generalized Assignment

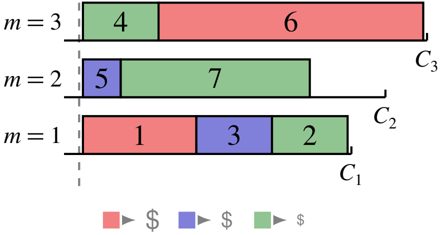



```
Machines = 1:3
@axis(Machines)
model = BlockModel()

@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]) )

@DantzigWolfe_decomposition(model, dec, Machines)
```

Modeling using JuMP + BlockDecomposition.jl

- Write a model of your problem using JuMP
- Use variable and constraint indices to define the decomposition
- Store extra information in the indices -> `axis` macro
- Create a decomposition tree -> `dantzig_wolfe_decomposition` macro



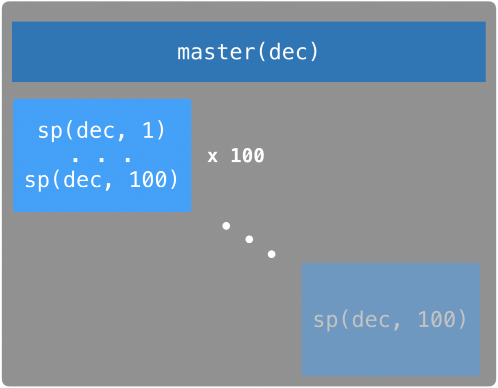


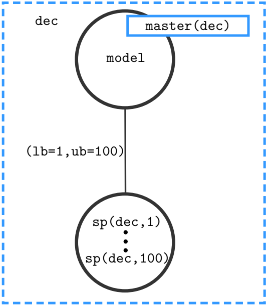
$$
\begin{array}{lrcll}
\min & \sum_{jm} c_{jm}\, x_{jm} & & & \\
\text{s.t.} & \sum_{m=1}^{M} x_{jm} & \geq & 1 & \forall j \\
 & \sum_{j} w_{jm}\, x_{jm} & \leq & C_m & \forall m \\
 & x_{jm} & \in & \{0,1\} & \forall j, m
\end{array}
$$

$$
\begin{array}{lrcll}
\min \sum_{q} c_q\, \lambda_q & & & & \\
\sum_{m \in M} \sum_{q \in Q^m} x_j^q\, \lambda_q & & \geq & 1 & \forall j \\
\sum_{q \in Q^m} \mathbf{y^q} \lambda_q & & = & 1 & \forall m \\
\lambda_q & & \in & \{0,1\} & \forall q
\end{array}
$$

where $\mathbf{y}$ is an added **setup-variable** in the sub-problem.

Cutting Stock : Identical subproblems





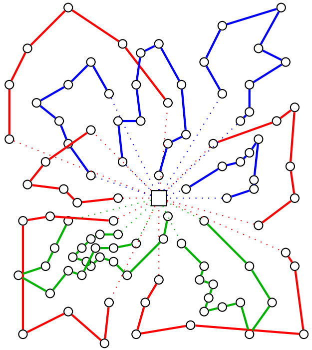
Cutting Stock : Identical subproblems

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

```
SheetTypes = 1:1
Orders = 1:100

model = BlockModel()
@axis(SheetTypes)
@variable(model, x[i in Orders, k in SheetTypes], Int)
@variable(model, y[k in SheetTypes], Int)
@objective(model, Min,
    sum(cost[k] * y[k] for k in SheetTypes))
@constraint(model, cov[i in Orders],
    sum(x[i,k] for k in SheetTypes) >= 1 )
@constraint(model, knp[k in SheetTypes],
    sum(weight[i] * x[i,k] for i in Orders) <= Capacity[k]) )

@DantzigWolfe_decomposition(model, dec, SheetTypes)
```

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

```
master = getmaster(dec)

addbranchingpriority!(master, y, 1)

SP = getsubproblems(dec)

specify!(SP, lm = 0, um = U, setup_var = y)

optimize!(model)

for k in SheetTypes
    solutions = Coluna.getsolutions(model, SP[k])
    for sol in solutions
        multiplicity = getvalue(sol, y[k])
        x_val  = [getvalue(sol, x[i, k]) for i in Orders]
    end
end
```



$$\min \sum_{e,r} c_e x_e^r$$

$$\text{s.t.} \sum_{e \in \delta(i), r} x_e^r \geq 2 \quad \forall i$$

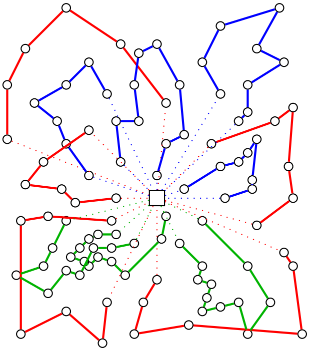
$$x^r \in X^r \quad \forall r$$

$$\min \sum_{e,r,q \in Q^r} c_e x_e^q \lambda_q$$

$$\sum_{e \in \delta(i), r, q \in Q^r} x_e^q \lambda_q \geq 2 \quad \forall i$$

$$\sum_{q \in Q^r} \lambda_q \leq U^r \quad \forall r$$

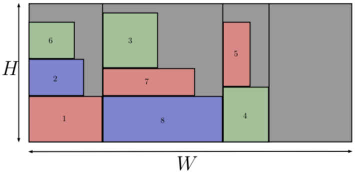
$$\lambda_q \in \{0,1\} \quad \forall r, q \in Q^r$$



```
model = BlockModel()
@axis(VehicleTypes)
@variable(model, x[e in Edge(G)], Int)
@variable(model, y[k in VehicleTypes], Bin)
@variable(model, z[a in Arc(G), k in VehicleTypes], Bin)
@objective(model, Min,
        sum(cost[e] * x[e] for e in Edge(G)))
@constraint(model, cov[i in Locations],
        sum(x[e] for e in Edge(G,i)) = 2 )
@expression(model, x[ (i,j) in Edge(G)],
        sum((z[(i,j),k] + z[(j,i),k]) for k in VehicleTypes) )
@DantzigWolfe_decomposition(model, dec, VehicleTypes)

function rcsp_fct(cb)
        k = getspidx(cb)
        red_costs = [getredcost(cb, x[e]) for e in Edge(G)]
        route = rcsp_pricing_solver(data, k, red_costs)
end
@oracle(model, sp_cb[k in VehicleTypes], rcsp_fct)
```

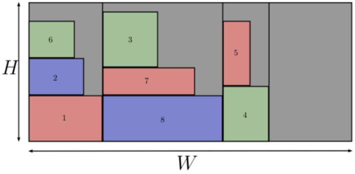
2D Cutting Stock : Original & master reformulations

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



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

2D Cutting Stock : Pricing problem and its reformulation



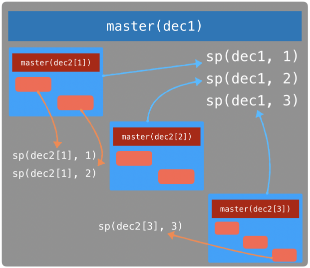
$$\min 1 - \left(\sum_j \pi_j \left(\sum_p x_{jp}\right)\right)$$

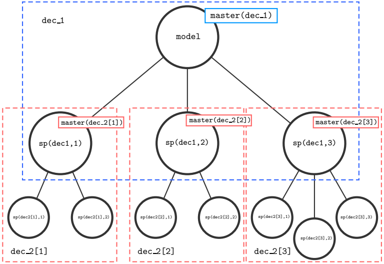
$$\begin{array}{rrcll}
\text{s.t.} & \sum_p x_{jp} & \leq & 1 & \forall j \\
& \sum_p w_p & \leq & W & \forall \\
& w_p & \geq & w_j \, x_{jp} & \forall j, p \\
& \sum_j h_j \, x_{jp} & \leq & H & \forall p \\
& x_{jp} & \in & \{0, 1\} & \forall j, p
\end{array}$$

$$\max \sum_p \left(\sum_j \pi_j x_j^p\right) \lambda_p$$

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

2D Cutting Stock : Nested Dantzig-W decomposition





@axis(**SheetTypes**)
@axis(**StripPatterns**)
@DantzigWolfe_decomposition(*model*, dec1, **SheetTypes**)
@DantzigWolfe_decomposition(dec1, dec2[k in **SheetTypes**], **StripPatterns**)

$$
\begin{array}{rrrcll}
 & \min \sum_k c^k \mathbf{x}^k & + \sum_{k,\omega} h^k_\omega \mathbf{y}^k_\omega & & & \\
\text{1th stage facility Linking C.} & \sum_k \mathbf{x}^k_t & & \geq & N_t & \forall t \\
\text{1th stage C. for each facility} & A^k \mathbf{x}^k & & \leq & a^k & \forall k \\
\text{link C. for each facility} & B^k \mathbf{x}^k & + T^k \mathbf{y}^k_\omega & \leq & b^k & \forall k, \omega \\
\text{?2d stage facility C.} & & \sum_k \mathbf{y}^k_{\omega t} & \geq & d_{\omega t} & \forall \omega, t \\
 & \mathbf{x}^k & \mathbf{y}^k_\omega & \in & \{0,1\}^n \times \mathbb{R}^m_+ & \forall k, \omega
\end{array}
$$

@axis(**K-facilities**)

@axis(**Ω-scenarios**)

@DantzigWolfe_decomposition

(*model*, dec1, **K-facilities**)

@Benders_decomposition

(*model*, dec2, **Ω-scenarios**)

Coluna's Data Structure

```
struct Problem <: AbstractProblem
    original_formulation::Union{Nothing, Formulation}
    re_formulation::Union{Nothing, Reformulation}
end

struct Formulation{Duty <: AbstractFormDuty} <: AbstractFormulation
    parent::Union{AbstractFormulation, Nothing}
    manager::FormulationManager
end

struct Reformulation  <: AbstractFormulation
    parent::Union{Nothing, AbstractFormulation}
    master::Union{Nothing, Formulation}
    dw_pricing_subprs::Vector{AbstractFormulation}
    benders_sep_subprs::Vector{AbstractFormulation}
end
```

Coluna's Data Structure

```
struct FormulationManager
    vars::VarDict
    constrs::ConstrDict
    coefficients::VarConstrMatrix
    expressions::VarVarMatrix
    primal_sols::VarVarMatrix
    dual_sols::ConstrConstrMatrix
end
struct MembersVector{IdT,ElemT,ValueT} <: AbstractMembersContainer
    elements
    records
end
struct MembersMatrix{IdT,ElemT,ValueT} <: AbstractMembersContainer
    cols
    rows
end
```

Coluna's Data Structure

```
struct Variable <: AbstractVarConstr
    id::Id{Variable}
    name::String
    duty::Type{<: AbstractVarDuty}
    perene_data::VarData
    cur_data::VarData
    moirecord::MoiVarRecord
end

struct VarData  <: AbstractVarData
    cost::Float64
    lb::Float64
    ub::Float64
    kind::VarKind
    sense::VarSense
    inc_val::Float64
    is_active::Bool
    is_explicit::Bool
end

struct Constraint <: AbstractVarConstr
    id::Id{Constraint}
    name::String
    duty::Type{<: AbstractConstrDuty}
    perene_data::ConstrData
    cur_data::ConstrData
    moirecord::MoiConstrRecord
end
```

```
struct OriginalVar <: AbstractOriginalVar end
struct OriginalExpression <: AbstractOriginalVar end

struct MasterPureVar <: AbstractOriginMasterVar end
struct MasterCol <: AbstractAddedMasterVar end
struct MasterArtVar <: AbstractAddedMasterVar end
struct MasterBendSecondStageCostVar <: AbstractAddedMasterVar end
struct MasterBendFirstStageVar <: AbstractOriginMasterVar end
struct MasterRepPricingVar <: AbstractMasterRepDwSpVar end
struct MasterRepPricingSetupVar <: AbstractMasterRepDwSpVar end

struct DwSpPricingVar <: AbstractDwSpVar end
struct DwSpSetupVar <: AbstractDwSpVar end
struct DwSpPureVar <: AbstractDwSpVar end

struct BendSpSepVar <: AbstractBendSpVar end
struct BendSpPureVar <: AbstractBendSpVar end
struct BendSpSlackFirstStageVar <: AbstractBendSpSlackMastVar end
struct BendSpSlackSecondStageCostVar <: AbstractBendSpSlackMastVar end

struct OriginalConstr <: OriginalConstr end

struct MasterPureConstr  <: AbstractMasterOriginConstrend
struct MasterMixedConstr  <: AbstractMasterOriginConstr end
struct MasterConvexityConstr  <: AbstractMasterAddedConstr end

struct DwSpPureConstr  <: bstractDwSpConstr end
struct DwSpRepMastBranchConstr  <: bstractDwSpConstr end
.
.
.
```

Coluna's Algorithms

Algorithm: is a routine involved in the optimisation process.

Solver: is a complete algorithmic strategy that we can solve a (re)formulation.

```
struct abstractConquerAlgorithm <: AbstractAlgorithm end
struct abstractDivideAlgorithm <: AbstractAlgorithm end
struct abstractExploreAlgorithm <: AbstractAlgorithm end

struct MySolver <: AbstractAlgorithm
    conquer_strategy::abstractConquerAlgorithm
    divide_strategy::abstractDivideAlgorithm
    explore_strategy::abstractExploreAlgorithm
end
```

Coluna's Feature

Current

- **Modeling**
  - JuMP Extension
  - Dantzig Decomposition
  - Benders Decomposition
- **Algorithms**
  - Column Generation
  - Benders cut Gen.
  - Branch-and-Bound
  - Branch-and-Price
  - Benders-Decomposition
  - Preprocessing
  - Primal Heuristics

Coluna's Feature

Current

- **Modeling**
  - JuMP Extension
  - Dantzig Decomposition
  - Benders Decomposition
- **Algorithms**
  - Column Generation
  - Benders cut Gen.
  - Branch-and-Bound
  - Branch-and-Price
  - Benders-Decomposition
  - Preprocessing
  - Primal Heuristics

Coming Soon

- **Efficiency**
  - Stabilization
  - Interior Point Approach
  - Strong Branching
  - Further Primal Heuristics
  - Pricing Strategies
  - **Parallelisation**
- **Cutomization**
  - Pricing Callback
  - Cutting Callback
  - Branching Callback
- **Functionality**
  - Template Cut Generation
  - Ad-hoc Branching
  - Nested / Hybrid Decomp.

Coluna Installation

- Install **Julia 1.2+**.
- Install **JuMP** & **BlockDecomposition**
- Install **GLPK** (or **CPLEX**, or **Gurobi**, or **Xpress** or **...**) as the default underlying MOI Optimizer for the master and the subproblems.
- Get **Coluna.jl** (using the package manager of Julia)
  - Go to the **Pkg-REPL-mode** :
    the Pkg REPL-mode is entered from the Julia REPL using the key]
  - The command to **install Coluna.jl** and its dependencies is

    pkg> add https://github.com/atoptima/Coluna.jl.git
  - Start **using Coluna** by doing :

    using Coluna

Welcome

- **β-users**
- **Integration**
  - Algorithms: Subgradient, Interior Point, Primal Heuristics
  - Oracles: Pricing, Separation
  - Applications: demo problems
- **Code review**
- **Co-development**

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

C. Bentes, T. Bulhoes, H. Kramer, G. Marques, V. Nesello, A. Pessoa, M. Poss, A. Subramanian, M. Tanneau, R. Sadykov, E. Uchoa, F. Vanderbeck

November 2019