Coluna.jl: an open source framework for column generation and decomposition approaches in mixed integer programming

Guillaume Marques, Vitor Nesello, Francois Vanderbeck[1], Artur Pessoa[2] Ruslan Sadykov[3]

[1] Atoptima
coluna@atoptima.com
[2] Universidade Federal Fluminense (UFF)
[3] Inria Bordeaux

Key words: *Column Generation, Branch-and-Cut-and-Price, Dantzig and Benders Decomposition, Open Source, Julia, JuMP, BlockDecomposition*

1 Context

Many optimisation problems are well-structured; they decomposed into subproblems that are more tractable and widely studied in the scientific literature. The subproblem can be defined by a set of constraints or variables. Sometimes both. While the binding of these subproblems makes the optimisation very challenging and direct solution approaches generally do not scale. A classic divide and conquer technique is to break it down into these subproblems. When subproblems are linked by binding constraints, the Danzig-Wolfe decomposition is applied. This results in a formulation involving an exponential number of variables. When subproblems are related by first stage design decision variables, the Benders decomposition is applied. This results in a formulation involving an exponential number of cuts. The branch-cut-and-price algorithm is the exact optimisation approach of choice for such reformulation involving an exponential number of variables and constraints. The scope of the application domain is large, ranging from multi-ressource multi-period operation planning problems to dealing with deterministic equivalent formulation for stochastic problems or robust optimisation, and formulation naturally emerging in the learning techniques of artificial intelligence.

2 An overview of Coluna

Coluna is a framework to implement decomposition approaches to tackle block-structured mixed-integer programs (MIP). The platform offers a library of functionalities to develop a so-called Branch-and-Price-and-Cut algorithm and to handle extended formulation for combinatorial optimisation problems. It is intended to cover both Dantzig-Wolfe and Benders decomposition principle, their hybrisation, and their recursive application in a nested decomposition.

Coluna is developed on an open-source basis and hence can engage innovations and constant updates; it is made to serve as a solid socle to researchers building further algorithmic progress. This open source project is carried by Atoptima, a spin-off from Inria/CNRS/University of Bordeaux [1], in collaboration with academic partners.

Another important feature of Coluna.jl is that it is completely implemented in Julia, a recent dynamic language that provides high productivity and high performance. Choosing Julia to

[1] Aptoptima is a software editor designing cutting-edge cloud-based applications for all businesses requiring complex resource assignment optimisation.

develop Coluna.jl was motivated by striking the best trade-off between efficiency and ease to share co-developments under a simple syntax to facilitate updates and extension of the framework.

Coluna aims to be very modular and customisable providing full control for the user of the behaviour of his algorithm. The framework builds on JuMP enhanced with the BlockDecomposition package : model your optimisation problem as a Mixed Integer Program (MIP) using the Julia modelling tools of JuMP and add instructions about the decomposable structure through the add-ons of BlockDecomposition.Then, Coluna reformulates the original MIP using Dantzig and/or Benders reformulation schemes to provide a working formulation while offering tools to project the solutions of the reformulation onto the original problem formulation. The framework offers algorithmic strategies to solve the reformulated problem. The user can take control on the algorithm either through call-backs or through the parametrisation of algorithmic strategy including the passing-over of functions as parameters, or taking over the open source routines implementing high-level strategies.

3 In this Talk

In this presentation, we will introduce the main features of Coluna, using the Generalised Assignment Problem as a running example : Given a set of machines and a set of tasks, the purpose of this problem is to minimise the cost of assigning jobs to machines while ensuring, for each machine, that the sum of the weights of the assigned tasks does not exceed the capacity of that machine. We will illustrate how to model and solve a problem with the defaults algorithmic strategies and how to customise them such as by using a cut callback to separate valid inequalities or a specific algorithm to solve a subproblem.

References

[1] Diana A. Barros, Guillaume Marques, Vitor Nesello, Cristiana Bentes, and Francois Vanderbeck. Exploiting Parallelism in the Open-Source Coluna.jl Framework. *POMCO*, 2020.

[2] Coluna.jl *GitHub repository.* https://github.com/atoptima/Coluna.jl.