Abstract

Large scale Mixed Integer Programs (MIP) can be solved by application of decomposition techniques such as Dantzig-Wolfe or Benders. The resulting reformulations can be solved by a generic Branch-and-Price-and-Cut algorithm where pricing and separation subproblems are solved using a MIP solver. The algorithm can be enhanced by user callbacks, primarily for the subproblems. Having a specialized exact or heuristic combinatorial algorithm for pricing or separation can improve substantially the performance of the generic algorithm.

JuMP.jl [1] offers a unified modeling language for solving MIP programs without decomposition. BlockDecomposition.jl extends the latter in order to model the information that describes a problem decomposition. It also extends JuMP callbacks with those useful in the scheme of decomposition-based algorithms.

The input is therefore split into two independent components. Using JuMP, the user specifies the set of constraints and variables of the MIP in its natural/compact formulation. Then, using simple functions of BlockDecomposition.jl, the user specifies which of these constraints and variables define the subsystems on which the decomposition is based (it is handy to test different decompositions). The validity of the decomposition is checked by BlockDecomposition.jl.

Several applications have been already implemented using JuMP/BlockDecomposition, and benchmarks were run using two underlying solvers which support decomposition techniques: BaPCod and Cplex [2] (with its Benders decomposition feature).

Références
