Warm Starting Series of Mixed-Integer Linear Programs

Sean Kelley [a]

[a] Department of Industrial and Systems Engineering, Lehigh University

January 3, 2022

Abstract

When one solves a mixed-integer linear programming (MILP) problem with today's state-of-the-art solvers, the solvers employ a Branch and Cut algorithm that returns an optimal solution while collecting a variety of additional information embedded within a binary tree. The information in this tree includes other feasible solutions, valid inequalities derived from cut generation, and a disjunction of the variables, and it is collected to close the primal-dual gap, the terminating condition of the algorithm. When one solves another MILP problem that differs from the one before only by objective coefficients or bounds on the constraints, some attributes of the previous problem's tree remain valid in the new one or are easily translated. By reusing this information, either directly or by using it to generate valid inequalities on the initial LP relaxation, we can tighten the primal and dual bounds more effectively than by the solver's primal heuristics alone, leading to a quicker algorithm execution. Such conditions to reuse a previous tree may seem restrictive, but several well-known classes of mixed-integer programming (MIP) problems rely on finding solutions by solving a series of sufficiently structured problems. In this work, we explore different means of reusing information from previous MILP solves and explore the effectiveness of doing so in improving the efficiency of solving more complex classes of MIP problems.

1 Introduction

1.1 Motivation

Mixed-Integer Programming (MIP) is a class of problems in which one minimizes an objective subject to a set of constraints and the restriction that some variables must take on integer values. MIP has a deeply studied theory and has been adopted in many successful industrial applications, such as production scheduling, vehicle routing, and facility location. Much of this theory and many of these applications are focused on the case in which a single Mixed-Integer Linear Programming (MILP) instance, which is a MIP instance with linear constraints and objective, is solved. In recent decades, this theory has been built upon to enrich the space of solvable MIP instances. Some examples include solving decomposable MILP's with many constraints or variables, MILP's with multiple objectives, MIP's with nonlinear objective or constraints (MINLP), and MILP's with "real-time" run time restrictions. With this growth in theory, MIP is more relevant to industry than ever, specifically in the aforementioned applications.

A major bottleneck to implementing applications of more recently developed subclasses of MIP, like those mentioned above, is solving them in a reasonable amount of time. The computational difficultly in the above examples stems from the fact that each relies on solving a series of many MILP instances, where solving a single instance is already an NP-hard problem. However, in these cases, each MILP instance in the series shares a similar structure with the others in that they share coefficient matrices and differ only in right hand sides (RHS's) or objectives. In this paper, we show how this structure can be taken advantage of to greatly improve the run time when solving many such instances in a row.

1.2 Approach

In this work, we improve the run time of each MILP solve in such closely related series (referred to here on out as a "family") by "warm starting" its solver, which means that we provide the solver with information beyond the model formulation that shrinks the space in which it will have to search for a solution. Specifically, this means we provide the solver with information that reduces the primal-dual gap, which we do in the following four ways:

1. We provide the solver with feasible solutions found by previous MILP solves in the family. When only the objective differs among the instances, we can set the primal bound of the current instance to the objective value of the best previously found solution in terms of the current objective.

2. We provide the solver with a dual function generated by previous MILP solves in the family. When only the RHS differs among the instances, we can set the dual bound of each LP relaxation in the current instance's branch and bound tree to the evaluation of the dual function at the relaxation's right hand side.

3. We provide the solver with valid inequalities generated by previous MILP solves in the family. When only the objective differs amongst the instances, we can add all of the previously valid inequalities to the current instance as they all share the same feasible region. When the RHS's change amongst the instances, we can re-evaluate the methods for generating each previously valid inequality at the current RHS to yield inequalities valid for the current instance.

4. We provide the solver with a disjunction generated by previous MILP solves in the family. We can input this disjunction into a cut generating linear program (CGLP) that will generate valid cuts during each cut-generating routine of branch and cut, which refines the feasible region near the optimal solution.

By reducing the primal-dual gap via the above four methods, we reduce the number of nodes in the branch and bound tree that we must bound before finding the optimal solution. We demonstrate in this paper the effectiveness of doing so against a sample of families in each of the seven following problem types: restarting single MILP instances, dual decomposition for stochastic MILP, Branch and Price, Multi-Objective MILP, MINLP, real-time MILP, and serialized Local Search primal heuristic.

1.3 Outline

This paper is organized as follows. In section 2 we cover preliminaries on MILP, and in section 3 we cover the details on what each numerical experiment will have in common. We dedicate sections 4 to 10 to each of the six problem types above. Each of those sections covers how the given problem type builds upon MILP, including what is the MILP family from which many similar instances will be solved, how we incorporate the four aforementioned warm starting methods, and how our warm started approach compares to solving each MILP. We conclude in section 11 with our thoughts on where warm starting MILP instances is most appropriate and further research directions.

1.4 Literature Review

This work is largely an extension of [5], which warm-starts MILP's differing only by objective or constraint bounds. In that work, warm-starting is achieved by reusing the Branch-and-Cut tree and cuts generated during the solve process. The main drawback to that approach, however, is that the tree can be large, requiring the solver to explore many possible paths to the optimal solution. Warm starting by only adding strong cuts for the new MILP instance to the root relaxation circumvents this issue, preserving a refined relaxation while leaving only one node to search instead of many. The common approach to find these cuts is through Lift-and-Project [2]. Until recently, such an approach was considered intractible when using a Branch-and-Cut tree with more than a few disjunctive terms, but [1] proposes a new formulation for the problem, enabling its application at scale. Our contribution is to generate cuts in this fashion from a previously solved MILP Branch-and-Cut tree, using them alone to warm start a new MILP instance and gaining the aforementioned advantages.

Before continuing, it is worth comparing our approach to the learning-based approaches to warm starting, which have soared in popularity over the last few years. Learning-based approaches train neural networks to predict primal heuristic and branching decisions made during branch and bound algorithms [4], [3]. When put into practice, such approaches enable solvers to more quickly navigate the branch and bound tree for new instances. This differs from our approach as we improve the solve time of a MILP instance by reducing the space in which a solver must search for a solution. One advantage of learning-based warm-starts is that they give the flexibility that not every problem learned on must have as similar structure as the problems we are considering for predictions to be accurate. For example, the problems could semantically be the same but differ in number of constraints and variables. A disadvantage, however, is that learning-based approaches require many computational and human resources to generate warm-starts, which must be repeated when applied to a new problem. On the other hand, the framework we propose for sufficiently structured problems requires relatively little computational overhead and can be quickly retrofitted to new applications. It is worth noting, though, that because both approaches work in different ways to improve the run time of MILP solves, it could be worth exploring the effectiveness of using both in tandem in the future.

References

[1] Balas and Kazachov. V-polyhedral disjunctive cuts. *N/A*, 2021.

[2] Sebastián Ceria Egon Balas and Gérard Cornuéjols. A lift-and-project cutting plane algorithm for mixed 0-1 programs. *Math. Program.*, 1993.

[3] Lodi et. al. Learning to schedule heuristics in branch-and-bound. *ArXiv*, 2019. `doi:https://arxiv.org/abs/2103.10294`.

[4] Nair et. al. Solving mixed integer programs using neural networks. *ArXiv*, 2020. `doi:https://arxiv.org/pdf/math/0411298v1.pdf`.

[5] Ralphs. Warm starting mixed integer programs. *COR@L*, 2006. `doi:https://coral.ise.lehigh.edu/~ted/files/papers/DMII06.pdf`.