Machine Learning for Integer Programming

Elias B. Khalil
School of Computational Science & Engineering
Georgia Institute of Technology
ekhalil3@gatech.edu

Abstract

Mixed Integer Programs (MIP) are solved exactly by tree-based branch-and-bound search. However, various components of the algorithm involve making decisions that are currently addressed heuristically. Instead, I propose to use machine learning (ML) approaches such as supervised ranking and multi-armed bandits to make better-informed, input-specific decisions during MIP branch-and-bound. My thesis aims at improving the overall performance of MIP solvers. To illustrate the potential for ML in MIP, I have so far tackled *branching variable selection*, a crucial component of the search procedure, showing that ML approaches for variable selection can outperform traditional heuristics.

1 Motivation

Recently, discrete optimization has been successfully leveraged to improve machine learning (ML) methodology. I will focus on the opposite direction of this fruitful cross-fertilization. I explore ways to harness ML approaches to improve the performance of branch-and-bound search for Mixed Integer Linear Programming (MIP). Although branch-and-bound solves MIP problems exactly, it is known that modern MIP solvers exhibit many heuristic decisions in the course of the algorithm [Lodi, 2013]. These decisions include:

- Cutting planes: various types of cuts may be valid, but only some of them have to be selected and added;
- Node selection: a node must be selected among active nodes and have its LP relaxation solved;
- Branching: a variable must be selected at each node in order to expand it into two child nodes.

To see how such decisions are currently handled, we refer to papers describing IBM CPLEX [Achterberg and Wunderling, 2013] and SCIP [Achterberg, 2009], the state-of-the-art commercial and non-commercial MIP solvers, respectively. In cut selection, a scoring system combining a handful of quality measures is used to score cuts, and the best among them are chosen [Achterberg, 2009]. However, both the quality measures and their weights in the scoring formula are chosen purely heuristically, based on the algorithm designer's intuition. In node selection, best-bound and best-estimate scoring heuristics are the default in CPLEX and SCIP, respectively. Again, both methods are heuristic in nature, and do not adapt to the input MIP instance in any way. Similar issues arise in branching, which we discuss in detail later.

My thesis is that data-driven decision-making in these components, via ML, can improve the overall performance of the MIP solver. For instance, most of the problems I just discussed involve *ranking* nodes, variables or cuts using a scoring function. It is natural to ask whether a superior function can be designed by collecting the appropriate data and *learning* a ranking function from it. Alternatively, one can use *online learning* approaches, such as multi-armed bandit, to model node/variable selection problems, after defining appropriate reward and/or transition functions.

This data-driven approach promises to create more flexible solvers that can adapt to new types of problems from new domains, without having to tune the strategies by manual engineering and trial-and-error.

Next, I will present the work I have already completed by the time of submission to the DC. The work is my own, under the supervision of my advisor, Prof. Bistra Dilkina.

2 Current Results

Preliminaries. Branching variable selection (or *branching*; see definition in section 1) is a main component of modern MIP solvers. Choosing good variables to branch on often leads to a dramatic reduction in terms of the number of nodes needed to solve an instance; see [Achterberg and Wunderling, 2013] for extensive experiments. Existing approaches for branching are based on either Strong Branching (SB) or Pseudocost branching (PC). However, both SB and PC suffer from serious limitations in terms of time-efficiency (for SB), node-efficiency, or reliance on expert manual tuning (for PC).

2.1 A Supervised Learning Approach

I have developed a novel framework for data-driven, on-the-fly design of variable selection strategies, see [Khalil *et al.*, 2016]. By leveraging research in supervised ranking, the aim is to produce strategies that gather the best of all properties: 1) using a small number of search nodes, approaching the good performance of SB, 2) maintaining a low computation footprint as in PC, and 3) selecting variables adaptively based on the properties of the given instance.

In our novel framework, and in the context of a single branch-and-bound search, we first observe the decisions made by SB, and collect: features that characterize variables at each node of the tree, and labels that discriminate among branching variables. In a second phase, we learn an easy-to-evaluate surrogate function that mimics SB, by solving a *learning-to-rank* problem, with the collected data being used for training. In a third phase, the learned ranking function is used for branching.

Compared to recent machine learning methods for node and variable selection in MIP [He *et al.*, 2014; Alvarez *et al.*, 2014], our approach: 1) can be applied to instances *on-the-fly*, without an upfront offline training phase on a large set of instances, and 2) consists of solving a ranking problem, as opposed to regression or classification, which are less appropriate for variable selection.

Results. We do not give details of the experimental setup here, and refer to our paper instead [Khalil *et al.*, 2016]. Our approach, SB+ML, significantly outperforms competing methods, solving more instances than both PC and SB+PC (a hybrid of SB and PC), while also requiring around 36% and 16% fewer nodes on average, respectively. In terms of running time, our method incurs some additional overhead per-node due to feature computations, yet it can be 15 to 20% faster than the competitors overall on instances of medium and hard difficulties, thanks to large savings in the number of nodes processed.

2.2 An Online Learning Approach

I have also studied variable selection through the lens of online learning, specifically *multi-armed bandits*. I have worked on bandit approaches that automatically balance *exploitation* (selecting the variable with the best average performance in previous nodes) with some *exploration* (selecting variables that are rarely selected), or alternatively with *risk-aversion* (selecting variables which have been selected more often), offering more nuanced selection rules. Extensive experimental results show that one of our methods can outperform the classical pseudocost branching (PC) strategy. To our knowledge, this is the first attempt at integrating online bandit learning with branching, a promising direction for tackling selection tasks within exact search algorithms.

Our main observation is that the variable selection problem can be cast as a *multi-armed bandit* (MAB) problem [Robbins, 1985], a prominent paradigm in online learning. In the multi-armed bandit problem, at each round an algorithm selects one out of a set of arms (actions), and observes the reward associated with the selected arm only; the goal is to select arms so as to minimize the regret w.r.t. the best arm. In branch-and-bound, the variable selection problem is naturally formulated in MAB terms as follows: each node of the search tree is a round, each variable is an arm, and some performance measure, function of the children resulting from the branching, is the reward of the selected variable. This work makes the case for more integrated research in search algorithms and online learning, as there is much room for developing better theory and algorithms for learning during search.

Results. The experimental setup is similar to that in [Khalil *et al.*, 2016]. The risk-averse MAB rule we propose, RA−VAR, is shown to outperform PC, solving more instances, and using around 6% fewer nodes, on average. For instances solved by both RA−VAR and PC, RA−VAR uses fewer nodes for 55% of those instances. The improvement we obtain is statistically significant in a Wilcoxon signed-rank test.

3 Research Plan

Over the long term, I plan on expanding my research along two main axes. First, in terms of ML approaches, I will develop fully online learning algorithms for decision-making in branch-and-bound, in the spirit of the work presented in section 2.2, and fully offline learning algorithms for families of instances with similar structure. Second, I will apply the data-driven approach to other components of branch-and-bound, such as primal heuristics and node selection. I believe that working along these two axes will bring about a thesis that fully integrates search, optimization and learning, with strong practical impact. Next, I discuss short-term research steps.

I am currently working on improving the supervised learning framework that I proposed in [Khalil *et al.*, 2016] by adapting the learning to the structure of the input instance, through dynamically adjusting the number of training nodes (in the data collection phase), depending on the usefulness of the current data and associated ranking model, or learning multiple models in adaptation to the search progress.

Another topic I am currently working on is *primal heuristics*, algorithms that try to find feasible solutions during the search. Modern solvers implement many of these heuristics (44 in SCIP), and deciding *which* heuristics to run and *when* to run them (i.e. at which nodes) is an important task. Solvers currently have pre-set parameter values that address these two questions, without taking into account the structure of the instance at hand, nor the evolution of the search. I am addressing these questions using bandit ideas, and expect to have concrete algorithms and results by the end of the year.

References

[Achterberg and Wunderling, 2013] Tobias Achterberg and Roland Wunderling. Mixed integer programming: Analyzing 12 years of progress. In Michael Jünger and Gerhard Reinelt, editors, *Facets of Combinatorial Optimization*. 2013.

[Achterberg, 2009] Tobias Achterberg. SCIP: solving constraint integer programs. *Mathematical Programming Computation*, 1(1):1–41, 2009.

[Alvarez *et al.*, 2014] Alejandro Marcos Alvarez, Quentin Louveaux, and Louis Wehenkel. A supervised machine learning approach to variable branching in branch-and-bound. 2014. Technical Report, Université de Liège.

[He *et al.*, 2014] He He, Hal Daumé III, and Jason Eisner. Learning to search in branch-and-bound algorithms. In *NIPS*, 2014.

[Khalil *et al.*, 2016] Elias B. Khalil, Pierre Le Bodic, Le Song, George Nemhauser, and Bistra Dilkina. Learning to branch in mixed integer programming. In *AAAI*, 2016.

[Lodi, 2013] Andrea Lodi. The heuristic (dark) side of MIP solvers. In *Hybrid Metaheuristics*, pages 273–284. Springer, 2013.

[Robbins, 1985] Herbert Robbins. Some aspects of the sequential design of experiments. In *Herbert Robbins Selected Papers*, pages 169–177. Springer, 1985.