Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI-21)
Survey Track

End-to-End Constrained Optimization Learning: A Survey

James Kotary[1] , **Ferdinando Fioretto**[1] , **Pascal Van Hentenryck**[2] and **Bryan Wilder**[3]
[1]Syracuse University
[2]Georgia Institute of Technology
[3]Harvard University
{jkotary, ffiorett}@syr.edu, pvh@isye.gatech.edu, bwilder@g.harvard.edu

Abstract

This paper surveys the recent attempts at leveraging machine learning to solve constrained optimization problems. It focuses on surveying the work on integrating combinatorial solvers and optimization methods with machine learning architectures. These approaches hold the promise to develop new hybrid machine learning and optimization methods to predict fast, approximate, solutions to combinatorial problems and to enable structural logical inference. This paper presents a conceptual review of the recent advancements in this emerging area.

1 Introduction

Constrained optimization (CO) has made a profound impact in industrial and societal applications in numerous fields, including transportation, supply chains, energy, scheduling, and the allocation of critical resources. The availability of algorithms to solve CO problems is highly dependent on their form, and they range from problems that are efficiently and reliably solvable, to problems that provably have no efficient method for their resolution. Of particular interest in many fields are *combinatorial optimization* problems, which are characterized by discrete state spaces, and whose solutions are often combinatorial in nature, involving the selection of subsets, permutations, paths through a network, or other discrete structures to compose a set of optimal decisions. They are known for their difficulty, and are often NP-Hard.

Despite their complexity, many CO problems are solved routinely and the AI and Operational Research communities have devised a wide spectrum of techniques and algorithms to effectively leverage the problem structure and solve many hard CO problems instances within a reasonable time and with high accuracy. While this success has made possible the deployment of CO solutions in many real-world contexts, the complexity of these problems often prevents them to be adopted in contexts of repeated (e.g., involving expensive simulations, multi-year planning studies) or real-time nature, or when they depend in nontrivial ways on empirical data. However, in many practical cases, one is interested in solving problem instances sharing similar patterns. Therefore, machine learning (ML) methods appear to be a natural candidate to aid CO decisions and have recently gained traction in the nascent area at the intersection between CO and ML.

Current research areas in the intersection of CO and ML can be categorized into two main directions: *ML-augmented CO* and *End-to-End CO learning*. The former focuses on using ML to aid the decisions performed within an optimization algorithm used to solve CO problems. The latter involves the combination of ML and CO techniques to form integrated models which predict solutions to optimization problems. The survey subdivides this context to into two main application domains: **(1)** the fast, approximate prediction of solutions to CO problems and **(2)** the integration of data-driven inference with CO solvers for structured logical inference.

While there exists work surveying ML-augmented CO methods [Bengio *et al.*, 2020; Lodi and Zarpellon, 2017; Cappart *et al.*, 2021], the more modern end-to-end CO learning methods lack of a cohesive and critical analysis. The goal of this survey is to address this gap and provide a focused overview on the work to-date on end-to-end CO learning, provide a critical analysis, and pose a set of open questions.

2 Preliminaries

A constrained optimization (CO) problem poses the task of minimizing an *objective function* $f : \mathbb{R}^n \to \mathbb{R}_+$ of one or more variables $\boldsymbol{z} \in \mathbb{R}^n$, subject to the condition that a set of *constraints* $\mathcal{C}$ are satisfied between the variables:

$$\mathcal{O} = \underset{\boldsymbol{z}}{\operatorname{argmin}} f(\boldsymbol{z}) \;\text{ subject to }\; \boldsymbol{z} \in \mathcal{C}. \tag{1}$$

An assignment of values $\boldsymbol{z}$ which satisfies $\mathcal{C}$ is called a *feasible solution*; if, additionally $f(\boldsymbol{z}) \leqslant f(\boldsymbol{w})$ for all feasible $\boldsymbol{w}$, it is called an *optimal solution*.

A well-understood class of optimization problems are *convex* problems, those in which the constrained set $\mathcal{C}$ is a convex set, and the objective f is a convex function. Convex problems have the favorable properties of being efficiently solvable with strong theoretical guarantees on the existence and uniqueness of solutions [Boyd *et al.*, 2004].

A particularly common constraint set arising in practical problems takes the form $\mathcal{C} = \{\boldsymbol{z} \;:\; \boldsymbol{A}\boldsymbol{z} \leqslant \boldsymbol{b}\}$, where $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{b} \in \mathbb{R}^m$. In this case, $\mathcal{C}$ is a convex set. If the objective f is an affine function, the problem is referred to as *linear program* (LP). When a linearly constrained problem includes a quadratic objective rather than a linear one, the result is called a *quadratic program* (QP). If, in addition, some

subset of a problem's variables are required to take integer values, it is called *mixed integer program* (MIP). While LP and QP with convex objectives belong to the class of convex problems, the introduction of integral constraints ($\boldsymbol{x} \in \mathbb{N}^n$) results in a much more difficult problem. The feasible set in MIP consists of distinct points in $\boldsymbol{x} \in \mathbb{R}^n$, not only nonconvex but also disjoint, and the resulting problem is, in general, NP-Hard. Finally, nonlinear programs (NLPs) are optimization problems where some of the constraints or the objective function are nonlinear. Many NLPs are nonconvex and can not be efficiently solved [Nocedal and Wright, 2006].

Of particular interest are the *mixed integer linear programs* (MILPs), linear programs in which a subset of variables required to take integral values. This survey is primarily concerned with optimization problems involving linear constraints, linear or quadratic objective, and either continuous or integral variables, or a combination thereof.

Optimization Solving Methods

A well-developed theory exists for solving convex problems. Methods for solving LP problems include *simplex* methods [Dantzig, 1951], *interior point* methods [Boyd *et al.*, 2004] and *Augmented Lagrangian* methods [Hestenes, 1969; Powell, 1969]. Each of these methods has a variant that applies when the objective function is convex. Convex problems [Boyd *et al.*, 2004] are in the class of $\boldsymbol{P}$, and can be assumed to be reliably and efficiently solvable in most contexts. For a review on convex optimization and Lagrangian duality, the interested reader is referred to [Boyd *et al.*, 2004].

MILPs require a fundamentally different approach, as their integrality constraints put them in the class of NP-Hard problems. *Branch and bound* is a framework combining optimization and search, representable by a search tree in which a LP *relaxation* of the MILP is formed at each node by dropping integrality constraints, and solved using solution methods for linear programming.

Finally, *Constraint Programming* [Rossi *et al.*, 2006] is an additional effective paradigm adopted to solve industrial-sized MI(L)P and discrete combinatorial programs.

Summary of Deep Learning for CO

Supervised deep learning can be viewed as the task of approximating a complex non-linear mapping from targeted data. Deep Neural Networks (DNNs) are deep learning architectures composed of a sequence of layers, each typically taking as inputs the results of the previous layer [LeCun *et al.*, 2015]. Feed-forward neural networks are basic DNNs where the layers are fully connected and the function connecting the layer is given by $\boldsymbol{o} = \pi(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b})$, where $\boldsymbol{x} \in \mathbb{R}^n$ and is the input vector, $\boldsymbol{o} \in \mathbb{R}^m$ the output vector, $\boldsymbol{W} \in \mathbb{R}^{m \times n}$ a matrix of weights, and $\boldsymbol{b} \in \mathbb{R}^m$ a bias vector. The function $\pi(\cdot)$ is often non-linear (e.g., a rectified linear unit (ReLU)).

Supervised learning consider datasets $\boldsymbol{\chi} = \{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^N$ consisting of N data points with $\boldsymbol{x}_i \in \mathcal{X}$ being a feature vector and $\boldsymbol{y}_i \in \mathcal{Y}$ the associated targets. The goal is to learn a model $\mathcal{M}_\theta : \mathcal{X} \rightarrow \mathcal{Y}$, where θ is a vector of real-valued parameters, and whose quality is measured in terms of a nonnegative, and assumed differentiable, *loss function* $\mathcal{L} : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}_+$. The learning task minimizes the empirical risk function (ERM):

$$\min_\theta J(\mathcal{M}_\theta, \boldsymbol{\chi}) = \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(\mathcal{M}_\theta(\boldsymbol{x}_i), \boldsymbol{y}_i). \tag{2}$$

Most of the techniques surveyed in this work use (variants of) DNNs whose training conforms to the objective above. Other notable classes of deep learning methods used to solve CO problems are Graph Neural Networks (GNNs), which exploit architectures designed to perform inference on data described by graphs, and Reinforcement Learning (RL), which differs from supervised learning in not requiring labeled input/output pairs and concerns with learning a policy that maximizes some expected reward function. We refer the reader to [Wu *et al.*, 2020] and [Sutton and Barto, 2018] for an extensive overview of GNNs and RL, respectively.

3 Overview of ML and CO

Current research areas in the synthesis of constrained optimization and machine learning can be categorized into two main directions: *ML-augmented CO*, which focuses on using ML to aid the performance of CO problem solvers, and *End-to-End CO learning (E2E-COL)*, which focuses on integrating combinatorial solvers or optimization methods into deep learning architectures. A diagram illustrating the main research branches within this area is provided in Figure 1.

The area related with End-to-End CO learning is the focus of this survey and is concerned with the data-driven prediction of solutions to CO problems. This paper divides this area into: **(1)** approaches that develop ML architectures to predict fast, approximate solutions to predefined CO problems, further categorized into *learning with constraints* and *learning on graphs*, and **(2)** approaches that exploit CO solvers as neural network layers for the purpose of structured logical inference, referred to here as the *Predict-and-Optimize* paradigm.

Before reviewing the E2E-COL research area, the paper provides a brief overview of ML-augmented CO.

4 ML-augmented CO

ML-augmented CO involves the augmentation of already highly-optimized CO solvers with ML inference models. Techniques in this area draw on both supervised and RL frameworks to develop more efficient approaches to various aspects of CO solving for both continuous and discrete combinatorial problems.

In the context of combinatorial optimization, these are broadly categorized into methods that learn to guide the search decisions in branch and bound solvers, and methods that guide the application of primal heuristics within branch and bound. The former include low-cost emulation of expensive branching rules in mixed integer programming [Khalil *et al.*, 2016; Gasse *et al.*, 2019; Gupta *et al.*, 2020], prediction of optimal combinations of low-cost variable scoring rules to derive more powerful ones [Balcan *et al.*, 2018], and learning to cut [Tang *et al.*, 2020] in cutting plane methods within MILP solvers. The latter include prediction of the most effective nodes at which to apply primal heuristics [Khalil *et al.*, 2017b], and specification of primal heuristics such as the optimal choice of variable partitions in large

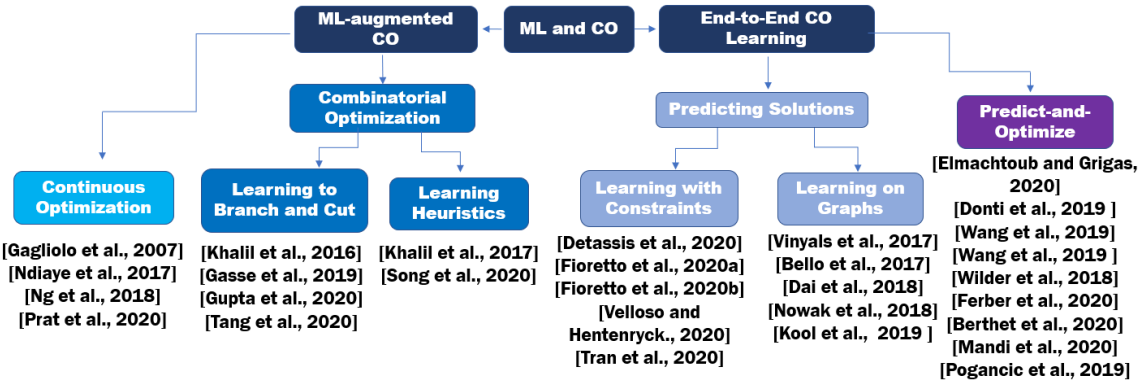

Figure 1: Machine Learning and Constrained Optimization.

neighborhood search [Song *et al.*, 2020]. The reader is referred to the excellent surveys [Lodi and Zarpellon, 2017; Bengio *et al.*, 2020] for a thorough account of techniques developed within ML-augmented combinatorial optimization.

Techniques in this area have also used ML to improve decisions in continuous CO problems and include learning restart strategies [Gagliolo and Schmidhuber, 2007], learn rules to ignore some optimization variables leveraging the expected sparsity of the solutions and consequently leading to faster solvers, [Ndiaye *et al.*, 2017], and learning active constraints [Ng *et al.*, 2018; Prat and Chatzivasileiadis, 2020] to reduce the size of the problem to be fed into a CO solver .

5 E2E-COL: Predicting CO Solutions

A diverse body of work within the end-to-end CO learning literature has focused on developing ML architectures to predict fast, approximate solutions to predefined CO problems *end-to-end* without the use of CO solvers at the time of inference, by observing a set of solved instances or execution traces. These approaches contrasts with those that use ML to augment search-based CO solvers and configure their subroutines to direct the solver to find solutions efficiently. This survey categorizes the literature on *predicting CO solutions* into **(1)** methods that incorporate constraints into end-to-end learning, for the prediction of feasible or near-feasible solutions to both continuous and discrete constrained optimization problems, and **(2)** methods that learn combinatorial solutions on graphs, with the goal of producing outputs as combinatorial structures from variable-sized inputs. These two categories, referred to as *learning with constraints* and *learning CO solutions*, respectively, are reviewed next.

5.1 Learning with Constraints

The methods below consider datasets $\boldsymbol{\chi} = \{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^N$ whose inputs $\boldsymbol{x}_i$ describe some problem instance specification, such as matrix $\boldsymbol{A}$ and vector $\boldsymbol{b}$ describing linear constraints in MILPs, and the labels $\boldsymbol{y}_i$ describe complete solutions to problem $\mathcal{O}$ with input $\boldsymbol{x}_i$. Notably, each sample may specify a different problem instance (with different objective function coefficients and constraints).

An early approach to the use of ML for predicting CO problem solutions was presented by Hopfield and Tank [1985], which used Hopfield Networks ([Hopfield, 1982]) with modified energy functions to emulate the objective of a traveling salesman problem (TSP), and applied Lagrange multipliers to enforce feasibility to the problem's constraints. It was shown in Wilson and Pawley [1988] however, that this approach suffers from several weaknesses, notably sensitivity to parameter initialization and hyperparameters. As noted by Bello et al. [2017], similar approaches have largely fallen out of favor with the introduction of practically superior methods.

Frameworks that exploit Lagrangian duality to guide the prediction of approximate solutions to satisfy the problem's constraints have found success in the context of continuous NLPs including energy optimization [Fioretto *et al.*, 2020b; Velloso and Van Hentenryck, 2020] as well as constrained prediction on tasks such as transprecision computing and fair classification [Fioretto *et al.*, 2020a; Tran *et al.*, 2021].

Other end-to-end learning approaches have demonstrated success on the prediction of solutions to constrained problems by injecting information about constraints from targeted feasible solutions. Recently, Detassis et al. [2020] presented an iterative process of using an external solver for discrete or continuous optimization to adjust targeted solutions to more closely match model predictions while maintaining feasibility, reducing the degree of constraint violation in the model predictions in subsequent iterations.

5.2 Learning Solutions on Graphs

By contrast to approaches learning solutions to unstructured CO problems, a variety of methods learn to solve CO cast on graphs. The development of deep learning architectures such as sequence models and attention mechanisms, as well as GNNs, has provided a natural toolset for these tasks and led to substantial improvements in the state of the art.

Vinyals et al. [2015] introduced the *pointer network*, in which a sequence-to-sequence model uses an encoder-decoder architecture paired with an attention mechanism to produce permutations over inputs of variable size. The resulting model was used to learn solutions to the TSP and the Delaunay triangulation problems from previously solved instances in a supervised manner, and demonstrated some ability to generalize over variable-sized problem instances.

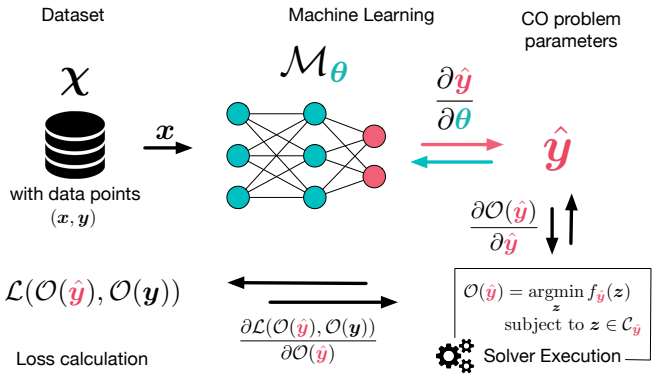

Figure 2: Predict-and-optimize framework; gradients of a solver output (solution) must be computed with respect to its input (problem parameters) in order to maximize empirical model performance.

This pointer network architecture was also adopted by Bello et al. [2017] but developed an improved model for learning the TSP by training it with RL, using tour length as the reward signal. The move from supervised to RL was motivated partly by the difficulties associated with obtaining target solutions that are optimal, and the existence of nonunique optimal solutions. Kool et al. [2019] applied an attention-based RL model to the TSP as well as variants of the vehicle routing problem, but with a graph attention network [Velickovic *et al.*, 2018] inspired by the Transformer architecture [Vaswani *et al.*, 2017]. This neural network design introduces invariance to permutations of the input nodes, improving learning efficiency.

Dai et al. [2017a] adopted a different RL approach based on a greedy heuristic framework which determines approximate solutions by the sequential selection of graph nodes. The selection policy is learned by reinforcement learning, using a graph neural network (GNN) to predict actions given a graph embedding representation of the current solution's state. Despite the trend toward RL-oriented frameworks, Nowak et al. [2018] discusses a purely supervised learning method for the general quadratic assignment problem based on the use of GNN's trained on representations of individual problem instances and their targeted solutions. Inferences from the model take the form of permutations, which are converted into feasible solutions by a beam search.

These and more approaches are covered in detail in Cappart et al. [2021], which provides a thorough survey on CO and reasoning with GNNs.

6 E2E-COL: Predict-and-Optimize

A burgeoning topic in the intersection of ML and CO is the fusion of prediction (ML) and decision (CO) models, in which decision models are represented by partially defined optimization problems, whose specification is completed by parameters that are predicted from data. The resulting composite models employ constrained optimization as a neural network layer and are trained *end-to-end*, based on the optimality of their decisions. This setting is altogether different in motivation to those previously discussed, in which the goal was to solve predefined CO instances with increased efficiency. The goal here is the synthesis of predictive and prescriptive techniques to create ML systems that learn to make decisions based on empirical data.

The following constrained optimization problem is posed, in which the objective function $f_{\boldsymbol{y}}$ and feasible region $\mathcal{C}_{\boldsymbol{y}}$ depend on a parameter vector $\boldsymbol{y}$:

$$\mathcal{O}(\boldsymbol{y}) = \operatorname*{argmin}_{\boldsymbol{z}} f_{\boldsymbol{y}}(\boldsymbol{z}) \;\text{ subject to }\; \boldsymbol{z} \in \mathcal{C}_{\boldsymbol{y}}. \tag{3}$$

The goal here is to use supervised learning to predict $\hat{\boldsymbol{y}}$ the unspecified parameters from empirical data. The learning task is performed so that the optimal solution $\mathcal{O}(\hat{\boldsymbol{y}})$ best matches a targeted optimal solution $\mathcal{O}(\boldsymbol{y})$, relative to some appropriately chosen loss function. The empirical data in this setting is defined abstractly as belonging to a dataset χ, and can represent any empirical observations correlating with targeted solutions to (3) for some $\boldsymbol{y}$. See Figure 2 for an illustration.

This framework aims to improve on simpler two-stage approaches, in which a conventional loss function (e.g. MSE) is used to target labeled parameter vectors $\boldsymbol{y}$ that are provided in advance, before solving the associated CO problem to obtain a decision. Such approaches are suboptimal in the sense that predictions of $\boldsymbol{y}$ do not take into account the accuracy of the resulting solution $\mathcal{O}(\boldsymbol{y})$ during training.

We note that there are two ways to view the predictions that result from these integrated models. If $\hat{\boldsymbol{y}}$ is viewed as the prediction, then the calculation of $\mathcal{O}(\hat{\boldsymbol{y}})$ is absorbed into the loss function $\mathcal{L}(\hat{\boldsymbol{y}}, \boldsymbol{y})$, which targets the provided parameter vectors. Otherwise, the loss function $\mathcal{L}(\mathcal{O}(\hat{\boldsymbol{y}}), \mathcal{O}(\boldsymbol{y}))$ is considered separately from the decision model and aims to match computed optimal solutions to targeted ones. One advantage sought in either case is the opportunity to minimize during training the ultimate error in the computed optimal objective values $f_{\hat{\boldsymbol{y}}}(\mathcal{O}(\hat{\boldsymbol{y}}))$, relative to those of the target data. This notion of training loss is known as *regret:*

$$\textit{regret}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = f_{\hat{\boldsymbol{y}}}(\mathcal{O}(\hat{\boldsymbol{y}})) - f_{\boldsymbol{y}}(\mathcal{O}(\boldsymbol{y})).$$

Otherwise the optimal solution $\mathcal{O}(\boldsymbol{y})$ is targeted and one can use regret$(\mathcal{O}(\hat{\boldsymbol{y}}), \mathcal{O}(\boldsymbol{y}))$*, regardless of whether the corresponding $\boldsymbol{y}$ is available.* Depending on the techniques used, it may be possible to minimize the regret without access to ground-truth solutions, as in Wilder et al. [2019], since the targeted solutions $\mathcal{O}(\boldsymbol{y})$ contribute only a constant term to the overall loss. It is worth mentioning that because the optimization problem in (3) is viewed as a function, the existence of nonunique optimal solutions is typically not considered. The implication then is that $\mathcal{O}(\boldsymbol{y})$ is directly determined by $\boldsymbol{y}$.

Training these end-to-end models involves the introduction of external CO solvers into the training loop of a ML model, often a DNN. Note that combinatorial problems with discrete state spaces do not offer useful gradients; viewed as a function, the *argmin* of a discrete problem is piecewise constant. *The challenge of forming useful approximations to $\frac{\partial \mathcal{L}}{\partial y}$ is central in this context and must be addressed in order to perform backpropagation.* It may be approximated directly, but a more prevalent strategy is to model $\frac{\partial \mathcal{O}(y)}{\partial y}$ and $\frac{\partial \mathcal{L}}{\partial \mathcal{O}}$ separately, in which case the challenge is to compute the former term by *differentiation through argmin*. Figure 2 shows the role of this gradient calculation in context.

6.1 Quadratic Programming

One catalyst for the development of this topic was the introduction of *differentiable optimization layers*, beginning with Amos et al. [2017] which introduced a GPU-ready QP solver that offers exact gradients for backpropagation by differentiating the KKT optimality conditions of a quadratic program at the time of solving, and utilizing information from the forward pass to solve a linear system for incoming gradients, once the outgoing gradients are known. Subsequently, Donti et al. [2017] proposed a *predict-and-optimize* model in which QPs with stochastic constraints were integrated in-the-loop to provide accurate solutions to inventory and power generator scheduling problems specified by empirical data.

6.2 Linear Programming

Concurrent with Donti et al. [2017], an alternative methodology for end-to-end learning with decision models, called *smart predict-and-optimize* (SPO), was introduced by Elmachtoub and Grigas [2021], which focused on prediction with optimization of constrained problems with linear objectives, in which the cost vector is predicted from data and the feasible region $\mathcal{C}$ is invariant to the parameter $\boldsymbol{y}$:

$$\mathcal{O}(\boldsymbol{y}) = \underset{\boldsymbol{z}}{\operatorname{argmin}}\, \boldsymbol{y}^T \boldsymbol{z} \;\text{ subject to }\; \boldsymbol{z} \in \mathcal{C}. \tag{4}$$

The target data in this work are the true cost vectors $\boldsymbol{y}$, and an inexact subgradient calculation is used for the backpropagation of regret loss $\mathcal{L}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \hat{y}^T(\mathcal{O}(\hat{\boldsymbol{y}}) - \mathcal{O}(\boldsymbol{y}))$ on the decision task, by first defining a convex surrogate upper bound on regret called the *SPO+loss*, for which it is shown that $\mathcal{O}(\boldsymbol{y}) - \mathcal{O}(2\hat{\boldsymbol{y}} - \boldsymbol{y})$ is a useful subgradient. Since this work is limited to the development of surrogate loss functions on regret from the optimization task, it does not apply to learning tasks in which the full solution to an optimization problem is targeted. The paper includes a discussion justifying the method's use on problems with discrete constraints in $\mathcal{C}$, as in combinatorial optimization, but experimental results are not provided on that topic. It is, however, demonstrated that the approach succeeds in a case where $\mathcal{C}$ is convex but nonlinear.

Wilder et al. [2019] introduced an alternative framework to predict-and-optimize linear programming problems, based on exact differentiation of a smoothed surrogate model. While LPs are special cases of QPs, the gradient calculation of Amos et al. [2017] does not directly apply due to singularity of the KKT conditions when the objective function is purely linear. This is addressed by introducing a small quadratic regularization term to the LP objective $f_{\boldsymbol{y}}(\boldsymbol{z}) = \boldsymbol{y}^T \boldsymbol{z}$ so that the problem in (3) becomes

$$\mathcal{O}(\boldsymbol{y}) = \underset{\boldsymbol{z}}{\operatorname{argmin}}\, \boldsymbol{y}^T \boldsymbol{z} + \epsilon \|\boldsymbol{z}\|^2 \;\text{ subject to }\; \boldsymbol{A}\boldsymbol{z} \leqslant \boldsymbol{b}. \tag{5}$$

The resulting problems approximate the desired LP, but have unique solutions that vary smoothly as a function of their parameters, allowing for accurate backpropagation of the result. The integrated model is trained to minimize the expected optimal objective value across all training samples, which is equivalent to minimizing the regret loss but without the need for a target dataset. This work demonstrated success on problems such as the knapsack (using LP relaxation) and bipartite matching problems where a cost vector is predicted from empirical data (e.g., historical cost data for knapsack items), and is shown to outperform two-stage models which lack integration of the LP problem. We note that although the differentiable QP solving framework of Amos et al. [2017] is capable of handling differentiation with respect to any objective or constraint coefficient, this work only report results on tasks in which the *cost* vector is parameterized within the learning architecture, and constraints are held constant across each sample. *This limitation is common to all of the works described below, as well.*

Next, Mandi et al. [2020] introduced an altogether different approach to obtaining approximate gradients for the *argmin* of a linear program. An interior point method is used to compute the solution of a homogeneous self-dual embedding with early stopping, and the method's log-barrier term is recovered and used to solve for gradients in the backward pass. Equivalently, this can be viewed as the introduction of a log-barrier regularization term, by analogy to the QP-based approach Wilder et al. [2019]:

$$\mathcal{O}(\boldsymbol{y}) = \underset{\boldsymbol{z}}{\operatorname{argmin}}\, \boldsymbol{y}^T \boldsymbol{z} + \lambda \left(\sum_i \ln(z_i) \right) \;\text{ subject to }\; \boldsymbol{A}\boldsymbol{z} \leqslant \boldsymbol{b}.$$

Further, the method's performance on end-to-end learning tasks is evaluated against the QP approach of Wilder et al. [2019] on LP-based predict-and-optimize tasks, citing stronger accuracy results on energy scheduling and knapsack problems with costs predicted from data.

Berthet et al. [2020] detailed an approach based on stochastic perturbation to differentiate the output of linear programs with respect to their cost vectors. The output space of the LP problem is smoothed by adding low-amplitude random noise to the cost vector and computing the expectation of the resulting solution in each forward pass. This can be done in Monte Carlo fashion and in parallel across the noise samples. The gradient calculation views the solver as a black box in this approach, and does not require the explicit solving of LP for operations that can be formulated as LP, but are simple to perform (e.g., sorting and ranking). Results include a replication of the shortest path experiments presented in Pogančić et al. [2020], in which a model integrated with convolutional neural networks is used to approximate the shortest path through stages in a computer game, solely from images.

6.3 Combinatorial Optimization

Ferber et al. [2020] extended the approach of Wilder et al. [2019] to integrate MILP within the end-to-end training loop, with the aim of utilizing more expressive NP-Hard combinatorial problems with parameters predicted from data. This is done by reducing the MILP with integer constraints to a LP by a method of cutting planes. In the ideal case, the LP that results from the addition of cutting planes has the same optimal solution as its mixed-integer parent. Exact gradients can then be computed for its regularized QP approximation as in Wilder et al. [2019]. Although the LP approximation to MILP improves with solving time, practical concerns arise when the MILP problem cannot be solved to completion. Each instance of the NP-Hard problem must be solved in each forward pass of the training loop, which poses clear runtime

issues. A disadvantage of the approach is that cutting-plane methods are generally considered to be inferior in efficiency to staple methods like branch and bound. Improved results were obtained on portfolio optimization and diverse bipartite matching problems, when compared to LP-relaxation models following the approach of Wilder et al. [2019].

Mandi et al. [2020] investigates the application of the same SPO approach to NP-Hard combinatorial problems. Primary among the challenges faced in this context is, as in Ferber et al. [2020], the computational cost of solving hard problems within every iteration of the training. The authors found that continuous relaxations of common MILP problems (e.g. knapsack) often offer subgradients of comparable quality to the full mixed-integer problem with respect to the SPO loss, so that training end-to-end systems with hard CO problems can be simplified in such cases by replacing the full problem with an efficiently solvable relaxation, an approach termed *SPO-relax*. The authors put continuous relaxations into the broader category of "weaker oracles" for the CO solver, which also includes approximation algorithms (e.g. greedy approximation for knapsack). The main results showed that SPO-relax achieves accuracy competitive with the full SPO approach but with shorter training times on a handful of discrete problems. The SPO-relax approach was compared also against the formulation of Wilder et al. [2019] on equivalent relaxations, but no clear winner was determined.

Pogančić et al. [2020] introduced a new idea for approximating gradients over discrete optimization problems for end-to-end training, which relies on viewing a discrete optimization problem as a function of its defining parameters (in this context coming from previous layers), whose range is piecewise constant. The only requirement is that the objective be linear. The gradient calculation combines the outputs of two calls to an optimization solver; one in the forward pass on initial parameters $\boldsymbol{y}$, and one in the backward pass on perturbed parameters $\bar{\boldsymbol{y}}$. The results are used to construct a piecewise *linear* function which approximates the original solver's output space, but has readily available gradients. Because the gradient calculation is agnostic to the implementation of the solver, it is termed "black-box differentiation". As such, input parameters to the solver do not correspond explicitly to coefficients in the underlying optimization problem. Results on end-to-end learning for the shortest path problem, TSP and min-cost perfect matching are shown. In each case, the discrete problem's specification is implicitly defined in terms of images, which are used to predict parameters of the appropriate discrete problem through deep networks. The optimal solutions coming from blackbox solvers are expressed as binary indicator matrices in each case and matched to targeted optimal solutions using a Hamming distance loss function.

Finally, Wang et al. [2019] presented a differentiable solver for the MAXSAT problem, another problem form capable of representing NP-Hard combinatorial problems. Approximate gradients are formed by first approximating the MAXSAT problem as a related semidefinite program (SPD), then differentiating its solution analytically during a specialized coordinate descent method [Wang *et al.*, 2018] which solves the SDP. The successful integration of MAXSAT into deep learning is demonstrated with a model trained to solve sudoku puzzles represented only by handwritten images.

7 Challenges and Research Directions

The current state of the art in integrating combinatorial optimization with end-to-end machine learning shows promise on challenging tasks for which there was previously no viable approach. Further, it has been demonstrated that a variety of non-overlapping approaches can be effective. Despite these encouraging results, a number of challenges remain that must be addressed to allow an integration that lives up to its full potential. **(1)** Despite the variety of approaches, the success of the predict-and-optimize paradigm has been demonstrated on a relatively limited set of optimization problems and a majority of reported experimental results focus on linear programming formulations. Challenges posed by the parametrization of constraints stand in the way of broader applications, but have not been yet been addressed. **(2)** Issues associated with the runtime of combinatorial solvers in-the-loop still make some potential applications impractical. **(3)** Additionally, despite being possible in theory, the role of the CO model in-the-loop has not been generalized successfully beyond being applied as the final layer of a deep model. The use of additional layers beyond the solver, or even compositions of CO solving layers, could potentially lead to new applications if the practical challenges were to be overcome. **(4)** In predicting solutions to CO problems, the current methods cannot reliably guarantee the problem constraints to be satisfied. This critical shortcoming may be addressed by integrating ML approaches with methods from the robust optimization literature or by developing ad-hoc layers to project the predictions onto the feasible space. **(5)** While it has been observed in limited contexts [Demirović *et al.*, 2019] that predict-and-optimize frameworks based on optimization layers are competitive only when the underlying constrained problem is convex, this area still lacks theoretical results providing guarantees on their viability or performance. Finally, **(6)** the need for uniform benchmark experiments and systematic comparisons between each predict-and-optimization framework is apparent. Demirović et al. [2019] provided a study comparing the approaches of Wilder et al. [2019] and Elmachtoub and Grigas [2021], along with problem-specific approaches, on knapsack problems but did not conclude strongly as to which method should be preferred. Further, Mandi et al. [2020] reports that for knapsack problems, SPO performs comparably on the knapsack problem whether the LP relaxation or the full problem is used, but does not show that this effect generalizes to other NP-Hard problems. This signals a need for studies that test performance on a variety of hard CO problems.

Although the approaches surveyed are still in an early stage of their development, and have been adopted mainly for academic purposes, we strongly believe that the integration between combinatorial optimization and machine learning is a promising direction for the development of new, transformative, tools in combinatorial optimization and learning.

Acknowledgments

This research is partially supported by NSF grant 2007164. Its views and conclusions are those of the authors only.

References

[Amos and Kolter, 2017] Brandon Amos and J Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning (ICML)*, pages 136–145. PMLR, 2017.

[Balcan *et al.*, 2018] Maria-Florina Balcan, Travis Dick, Tuomas Sandholm, and Ellen Vitercik. Learning to branch. In *International conference on machine learning (ICML)*, pages 344–353. PMLR, 2018.

[Bello *et al.*, 2017] Irwan Bello, Hieu Pham, Quoc V. Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. In *International Conference on Learning Representations, (ICLR) Workshops*. OpenReview.net, 2017.

[Bengio *et al.*, 2020] Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: a methodological tour d'horizon. *European Journal of Operational Research*, 2020.

[Berthet *et al.*, 2020] Quentin Berthet, Mathieu Blondel, Olivier Teboul, Marco Cuturi, Jean-Philippe Vert, and Francis R. Bach. Learning with differentiable pertubed optimizers. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[Boyd *et al.*, 2004] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[Cappart *et al.*, 2021] Quentin Cappart, Didier Chételat, Elias Khalil, Andrea Lodi, Christopher Morris, and Petar Veličković. Combinatorial optimization and reasoning with graph neural networks. *arXiv preprint arXiv:2102.09544*, 2021.

[Dantzig, 1951] George B Dantzig. Maximization of a linear function of variables subject to linear inequalities. *Activity analysis of production and allocation*, 13:339–347, 1951.

[Demirović *et al.*, 2019] Emir Demirović, Peter J Stuckey, James Bailey, Jeffrey Chan, Chris Leckie, Kotagiri Ramamohanarao, and Tias Guns. An investigation into prediction+optimisation for the knapsack problem. In *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research (CPAIOR)*, pages 241–257, 2019.

[Detassis *et al.*, 2020] Fabrizio Detassis, Michele Lombardi, and Michela Milano. Teaching the old dog new tricks: supervised learning with constraints. In Alessandro Saffiotti, Luciano Serafini, and Paul Lukowicz, editors, *Proceedings of the First International Workshop on New Foundations for Human-Centered AI (NeHuAI)*, volume 2659 of *CEUR Workshop Proceedings*, pages 44–51, 2020.

[Donti *et al.*, 2017] Priya L. Donti, J. Zico Kolter, and Brandon Amos. Task-based end-to-end model learning in stochastic optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5484–5494, 2017.

[Elmachtoub and Grigas, 2021] Adam N Elmachtoub and Paul Grigas. Smart "predict, then optimize". *Management Science*, 2021.

[Ferber *et al.*, 2020] Aaron Ferber, Bryan Wilder, Bistra Dilkina, and Milind Tambe. Mipaal: Mixed integer program as a layer. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

[Fioretto *et al.*, 2020a] Ferdinando Fioretto, Pascal Van Hentenryck, Terrence W. K. Mak, Cuong Tran, Federico Baldo, and Michele Lombardi. Lagrangian duality for constrained deep learning. In *European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD)*, volume 12461, pages 118–135, 2020.

[Fioretto *et al.*, 2020b] Ferdinando Fioretto, Terrence W.K. Mak, and Pascal Van Hentenryck. Predicting ac optimal power flows: Combining deep learning and lagrangian dual methods. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 630–637, 2020.

[Gagliolo and Schmidhuber, 2007] Matteo Gagliolo and Jürgen Schmidhuber. Learning restart strategies. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 792–797, 2007.

[Gasse *et al.*, 2019] Maxime Gasse, Didier Chételat, Nicola Ferroni, Laurent Charlin, and Andrea Lodi. Exact combinatorial optimization with graph convolutional neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 15554–15566, 2019.

[Gupta *et al.*, 2020] Prateek Gupta, Maxime Gasse, Elias B. Khalil, Pawan Kumar Mudigonda, Andrea Lodi, and Yoshua Bengio. Hybrid models for learning to branch. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[Hestenes, 1969] Magnus R Hestenes. Multiplier and gradient methods. *Journal of optimization theory and applications*, 4(5):303–320, 1969.

[Hopfield and Tank, 1985] John Hopfield and D Tank. Neural computation of decisions in optimization problems. *Biological cybernetics*, 52:141–52, 02 1985.

[Hopfield, 1982] John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

[Khalil *et al.*, 2016] Elias Khalil, Pierre Le Bodic, Le Song, George Nemhauser, and Bistra Dilkina. Learning to branch in mixed integer programming. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 30, 2016.

[Khalil *et al.*, 2017a] Elias B. Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 6348–6358, 2017.

[Khalil *et al.*, 2017b] Elias B. Khalil, Bistra Dilkina, George L. Nemhauser, Shabbir Ahmed, and Yufen Shao. Learning to run heuristics in tree search. In *Proceedings*

of the Twenty-Sixth International Joint Conference on Artificial Intelligence, (IJCAI), pages 659–666, 2017.

[Kool *et al.*, 2019] Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[LeCun *et al.*, 2015] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–444, 2015.

[Lodi and Zarpellon, 2017] Andrea Lodi and Giulia Zarpellon. On learning and branching: a survey. *Top*, 25(2):207–236, Jul 2017.

[Mandi and Guns, 2020] Jayanta Mandi and Tias Guns. Interior point solving for lp-based prediction+optimisation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[Mandi *et al.*, 2020] Jayanta Mandi, Peter J Stuckey, Tias Guns, et al. Smart predict-and-optimize for hard combinatorial optimization problems. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 34, pages 1603–1610, 2020.

[Ndiaye *et al.*, 2017] Eugene Ndiaye, Olivier Fercoq, Alexandre Gramfort, and Joseph Salmon. Gap safe screening rules for sparsity enforcing penalties. *The Journal of Machine Learning Research*, 18(1):4671–4703, 2017.

[Ng *et al.*, 2018] Yeesian Ng, Sidhant Misra, Line A Roald, and Scott Backhaus. Statistical learning for dc optimal power flow. In *2018 Power Systems Computation Conference (PSCC)*, pages 1–7. IEEE, 2018.

[Nocedal and Wright, 2006] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[Nowak *et al.*, 2018] Alex Nowak, Soledad Villar, Afonso S Bandeira, and Joan Bruna. Revised note on learning quadratic assignment with graph neural networks. In *2018 IEEE Data Science Workshop (DSW)*, pages 1–5. IEEE, 2018.

[Pogančić *et al.*, 2020] Marin Vlastelica Pogančić, Anselm Paulus, Vit Musil, Georg Martius, and Michal Rolinek. Differentiation of blackbox combinatorial solvers. In *International Conference on Learning Representations (ICLR)*, 2020.

[Powell, 1969] Michael JD Powell. A method for nonlinear constraints in minimization problems. *Optimization*, pages 283–298, 1969.

[Prat and Chatzivasileiadis, 2020] Eléa Prat and Spyros Chatzivasileiadis. Learning active constraints to efficiently solve bilevel problems. *arXiv preprint arXiv:2010.06344*, 2020.

[Rossi *et al.*, 2006] Francesca Rossi, Peter Van Beek, and Toby Walsh. *Handbook of constraint programming*. Elsevier, 2006.

[Song *et al.*, 2020] Jialin Song, Ravi Lanka, Yisong Yue, and Bistra Dilkina. A general large neighborhood search framework for solving integer linear programs. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 20012–20023, 2020.

[Sutton and Barto, 2018] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[Tang *et al.*, 2020] Yunhao Tang, Shipra Agrawal, and Yuri Faenza. Reinforcement learning for integer programming: Learning to cut. In *International Conference on Machine Learning (ICML)*, pages 9367–9376. PMLR, 2020.

[Tran *et al.*, 2021] Cuong Tran, Ferdinando Fioretto, and Pascal Van Hentenryck. Differentially private and fair deep learning: A lagrangian dual approach. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2021.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5998–6008, 2017.

[Velickovic *et al.*, 2018] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations, (ICLR)*, 2018.

[Velloso and Van Hentenryck, 2020] Alexandre Velloso and Pascal Van Hentenryck. Combining deep learning and optimization for security-constrained optimal power flow. *arXiv:2007:2007.07002*, 2020.

[Vinyals *et al.*, 2015] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *Advances in Neural Information Processing Systems (NeurIPS*, pages 2692–2700, 2015.

[Wang *et al.*, 2018] Po-Wei Wang, Wei-Cheng Chang, and J. Zico Kolter. The mixing method: low-rank coordinate descent for semidefinite programming with diagonal constraints. *arXiv:1706.00476*, 2018.

[Wang *et al.*, 2019] Po-Wei Wang, Priya Donti, Bryan Wilder, and Zico Kolter. Satnet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. In *International Conference on Machine Learning (ICML)*, pages 6545–6554. PMLR, 2019.

[Wilder *et al.*, 2019] Bryan Wilder, Bistra Dilkina, and Milind Tambe. Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 33, pages 1658–1665, 2019.

[Wilson and Pawley, 1988] GV Wilson and GS Pawley. On the stability of the travelling salesman problem algorithm of hopfield and tank. *Biological Cybernetics*, 58(1):63–70, 1988.

[Wu *et al.*, 2020] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 2020.