MATHEMATICAL OPTIMIZATION SOCIETY NEWSLETTER

OPTIMA 107

Issue 107 January 2025

Chair's Column

Dear colleagues,

The year 2024 was a very good year for the MOS. One of the high points was our 25th International Symposium on Mathematical Programming (ISMP) in Montreal, Canada, in July. This was our first post-pandemic symposium and with 1562 registered participants, including 498 students, it was a great success. On behalf of the MOS, I thank Charles Audet, the whole GERAD team, and the members of the ISMP organization and scientific committees for their sustained efforts in planning and running the Symposium. I also thank all the MOS members who attended and contributed to this success. I heard from many of you that the event was enjoyable and fruitful.

I am delighted to report that with well over 1500 members, the MOS is amply beyond the threshold of 1000 members required to apply for the status of "full large member" of ICIAM, the International Council for Industrial and Applied Mathematics. Our application for this elevated status was approved in November and we are now recognized as one of the larger societies within ICIAM. Thank you to Sam Burer for leading this effort. This recognition raises our profile worldwide and will, I hope, lead to greater engagement with ICIAM and with other professional societies.

Some of you may not know that the MOS hosts and supports technical sections. The Stochastic Programming Society (SPS) brings together researchers interested in decision-making under uncertainty, and the Mixed Integer Programming Society (MIPS) encompasses those focusing on optimization problems involving discrete decisions. I am delighted that the MOS welcomed in 2024 a third technical section, the newly formed Bilevel Optimization Society. You can follow its progress at https://bileveloptimization.org. Thank you to everyone listed on the website for helping to make this happen!

We also held our triennial elections in 2024. As many of you already know, our newly elected Council members are: Katya Scheinberg (Chair-Elect), Fatma Kılınç-Karzan (previously Member-at-Large and now Treasurer-Elect), and Merve Bodur, Fabian Bastin, Jorge Vera, and Xiaojun Chen (Members-at-Large). Congratulations to each one of them, and thank you to our other colleagues who agreed to run. As per the MOS bylaws, the members-at-large have been in post since the ISMP while Katya and Fatma take up their roles in August 2025, with Samuel Burer remaining as Treasurer, David Morton as Chair of Executive Committee, Katya as Vice-Chair, and I as Chair until then. Thank you also to our outgoing Council members, John Birge, Marina Epelman, Andreas Wächter, Angelika Wiegele, and Wolfram Wiesemann for their years of service to the MOS.

The Council has been working on improvements for the MOS and the services it provides to you, its members. We will keep you informed about new developments, and we are always open to your comments, ideas and suggestions.

In closing, I wish you an optimal year in 2025!

Miguel F. Anjos, University of Edinburgh
miguel.f.anjos@ed.ac.uk

Note from the Editors

This edition of OPTIMA focuses on the theme of Machine Learning and Optimization. Decision-making and data are deeply interconnected, and there are many exciting opportunities for tightly integrating the two through machine learning and optimization. This issue features two articles showcasing advancements at the intersection of these two areas.

In the first article, Andrea Lodi explores how Machine Learning is revolutionizing Mixed-Integer Programming (MIP) by introducing innovative methods to improve algorithmic efficiency and scalability. The second article, by Axel Parmentier, examines the integration of combinatorial optimization with Machine Learning, highlighting its practical applications in solving real-world, data-driven challenges.

We would also like to remind our readers about the upcoming ICCOPT 2025 conference, one of the leading global events in continuous optimization.

We hope you find these contributions inspiring and, as always, we welcome your feedback and suggestions for future topics.

Sebastian Pokutta, Editor
Swati Gupta, Co-Editor
Omid Nohadani, Co-Editor

ML-Augmented MIP Solving

Andrea Lodi

This short paper summarizes and provides useful pointers to the recent wave of work on the use of machine learning to augment the capabilities of the mixed-integer programming (MIP) technology, with special emphasis to the current and potential improvements in the MIP solvers.

1 Introduction

The impressive success that *machine learning* (ML) has achieved in the last 10 to 15 years in areas like image recognition, machine translation, games and language processing led a number of other scientific disciplines to question the use of modern statistical learning within the real of their techniques.

Combinatorial optimization (CO) has been no exception to this trend and the interplay between CO and ML has developed steadily since the middle of the previous decade. Of course, notable attempts within this interplay predates this latest wave (see, e.g., the survey by Smith (1999)), but the surge of work in the last decade represents the core of this short paper. Even more specifically, we concentrate on the attempts of using ML for improving the *mixed-integer programming* (MIP) technology, a fundamental asset to solve CO problems through a variety of commercial (FICO, 2023; Gurobi Optimization, LLC, 2023; IBM, 2023) and open-source (Gamrath et al., 2020) MIP solvers.[1] This area is often referred to as ML-augmented MIP and has been the subject of the detailed survey by Scavuzzo et al. (2024) of which this short paper is a reasoned summary.

The remainder of the paper is organized as follows. In the next section, we discuss the basic principles of the MIP algorithmic technology and the way this is implemented in MIP solvers. Then, Section 3 reviews the way in which each of the MIP building blocks introduced in Section 2 has benefited from ML. Finally, in Section 4, we draw some brief conclusions and we outline some perspectives.

2 Solving MIP Problems

We are considering a MIP in the form[2]

$$\begin{aligned} z^* = \min\ & c^T x \\ \text{subject to } & Ax \geq b, \\ & x_j \in \mathbb{Z}_{\geq 0} \quad \forall j \in \mathcal{A}, \\ & x_j \in \{0,1\} \quad \forall j \in \mathcal{B}, \\ & x_j \geq 0 \quad \forall j \in \mathcal{C}, \end{aligned} \tag{1}$$

where $A \in \mathbb{Q}^{m \times n}$, $c \in \mathbb{Q}^n$, $b \in \mathbb{Q}^m$, and $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ define a partition of the variable index set $\{1, \dots, n\}$.

The basic observation is that formulation (1) is NP-hard while dropping the integrality requirement on each variable x_j such that $j \in \mathcal{A} \cup \mathcal{B}$ leads to a problem, the *linear programming* (LP) relaxation, that is polynomially solvable. Moreover, solving the LP relaxation gives an optimistic approximation of the MIP optimal value, i.e., $z_{LP} \leq z^*$, where x_{LP} is the optimal solution value of the LP relaxation. Now, it is intuitive to see that the difference between z_{LP} and z^* can be seen as a proxy of the quality of the LP relaxation: The closer the two values, the better the approximation provided. So, the algorithmic technology for solving MIPs develops around two questions. First, what can we do to strengthen such a relaxation? Second, what can we do if despite such strengthening some gap remains?

The algorithmic technology in the current MIP solvers is composed by four major building blocks.

- *Preprocessing/Configuration*. This is the collection of techniques that are applied before the MIP core resolution starts. The idea is to clean the formulation by removing some redundancy[3] so as to reduce the size of the problem and, more importantly, to exploit implications and characteristics of the data to make the constraints and the variable bounds in (1) tighter. In addition, the analysis performed in the preprocessing might be used to configure algorithmic decisions in the attempt of solving the MIP faster.[4]
- *Cutting Planes Generation*. This is the main tool for strengthening the LP: linear inequalities that are valid for the convex hull of feasible solutions but might be violated by the solution of the LP relaxation, say $\bar{x}$, are added to formulation (1) in rounds, i.e., the LP relaxation is solved each time new cutting planes remove (separate) $\bar{x}$ from the feasible region of the MIP, thus potentially improving z_{LP} at each round. This scheme, called the *cutting plane algorithm*, originates on the work of Gomory (1958, 1960) and has revolutionized MIP computation in the late 90s.[5]
- *Branching*. The two previous building blocks take care of the LP strengthening. However, in most of the practical cases, the effectiveness of cutting planes (or cuts) generation reduces with the number of rounds applied and the numerical stability of the resulting LP can be affected, so one stops the strengthening process before the entire gap between z_{LP} and z^* is closed. Then, the exactness of the approach is guaranteed by the implicit enumeration scheme that splits the original MIP into subMIPs: given $\bar{x}$ such that $(\bar{x}_j - \lfloor \bar{x}_j \rfloor) > 0$ for some $j \in \mathcal{A} \cup \mathcal{B}$, two subMIPs are created by augmenting the original MIP with either the constraint $x_j \leq \lfloor \bar{x}_j \rfloor$ or the constraint $x_j \geq \lceil \bar{x}_j \rceil$. Of course, $\bar{x}$ is infeasible for both the newly created subMIPs, while the optimal solution of the original MIP will belong to one of the two. This scheme is recursively applied for each subMIP and it is easy to visualize it as a tree search where each subMIP is a branching node created by the addition of one constraint, see Figure 1. This leads to the exact algorithm originally devised by Land and Doig (1960) and known as *branch and bound* (B&B).[6] The complete enumeration of all possible subMIPs is avoided by appropriate bounding considerations that allow to discard nodes that cannot lead to an optimal solution.
- *Primal Heuristics*. The framework composed by the three building blocks discussed above takes care of the dual component of MIP, i.e., deals the blue trajectory in Figure 1 that describes the z_{LP} evolution. However, the optimality proof is obtained when a primal, i.e., feasible solution of the problem, say $\tilde{x}$, with the same value of the best dual bound is computed. One can wait for $\tilde{x}$ to be computed naturally by monotonically restricting the solution space of the subMIPs via branching but this is generally slow. Thus, to allow a fast converge, the red trajectory in Figure 1 is obtained by running a number of specialized algorithms within the MIP resolution. Those algorithms are called primal heuristics and we distinguish them depending on their computational requirements, in which part of the scheme are invoked, etc.[7]

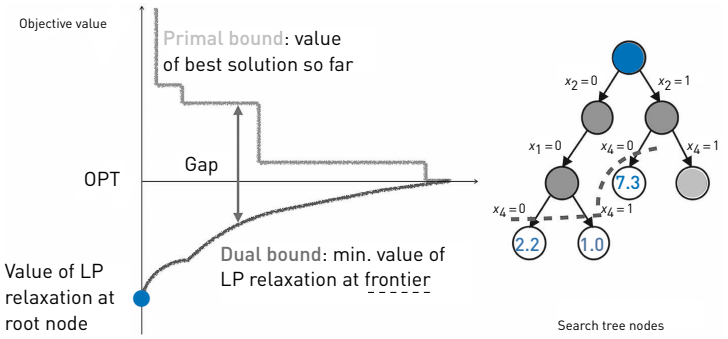

Figure 1. The branch-and-bound scheme for MIP (Picture courtesy of Elias Khalil)

In summary, MIP solvers solve MIP (1) by integrating in a sophisticated and reliable way the cutting plane and the branch-and-bound algorithms. This integration requires a number of algorithmic decisions that could benefit significantly by statistical learning as reviewed in the next section.

We conclude the section by noting that the current generation of MIP solvers extends its attention to more general classes of optimization problems than those described by (1), where the objective function or constraints can be nonlinear. The technology as presented here vastly applies to those classes too with very significant distinctions associated with potential lack of convexity.[8] Some of the attempts of improving the MIP technology that we survey in this paper are in fact useful (and applied) by MIP solvers to those more general classes, which is the reason why we adopted the notation MIP instead of MILP, where the "L" stands for *linear*.

3 ML-augmented MIP

Bengio et al. (2021) identified two major opportunities where modern statistical learning could help MIP technology by leveraging data. First, some of the algorithmic decisions of a MIP solver are simply too costly to make from the computational standpoint. In other words, one would know how to make a specific decision effectively but that would require too much time, a time that could be excessive with respect to how much that decision is relevant within the overall process. Second, for some other algorithmic decisions, instead, we do not have a good mathematical understanding and we are currently making them through human-learned rules.

The question is then how ML can help in both of these cases, and two types of learning algorithms have been used so far. On the one side, one can cast the case in which there is knowledge of how to effectively make a decision as a learning by *demonstration*. There is an expert, in this case an algorithm,[9] and we can observe its decisions and try to imitate them. This configures a *supervised learning* task where, once the ML model has been trained offline, at inference time the MIP decision will be made in a fraction of the time required by the expert, hopefully with good approximation. On the other side, if we do not have a deep understanding on how to characterize the quality of a decision, then we could make the data speak and learning by *exploration*. In this case, the ML model can be trained to make decisions and observe their impact by collecting for each decision a reward, so as to develop a decision policy that maximizes the reward cumulatively. This approach is in the real of *reinforcement learning*.

Due to space limitations, we will not be able here to give a proper introduction to ML techniques for which we refer to Bengio et al. (2021) and Scavuzzo et al. (2024) and the references thereafter. However, we will summarize the *learning process*, i.e., the process of *training* an ML model for its task. Technically, training is in itself an (inexact) optimization process characterized by

(i) *Optimization algorithms*, i.e., algorithms (generally) from continuous nonconvex optimization that are used to minimize the loss with respect to the expert or maximize the collected reward. The algorithm optimizes the parameters (or variables in optimization jargon) of the ML model.

(ii) *Hyper-parameters*, i.e., the parameters that do not belong to the ML model itself but influence the learning scheme, e.g., number of iterations, learning rate, etc.

(iii) *Data collection*, i.e., the phase in which the data used by ML is collected.

(iv) *Online vs offline learning*, i.e., if the data collection happens during the execution of the learning scheme or precedes it. In the latter case, the learning happens offline after the data collection and the ML model is executed once the learning has completed.

(v) *Train, validation and test datasets*, i.e., the split of the data used by ML to learn the model, tune the hyper-parameters and evaluate the model quality, respectively.

(vi) *Overfitting*, i.e., the phenomenon that is associated with exploiting the training data "too much". This appears when the quality of the prediction on the training set is significantly superior to that on the test set.

Before finally discussing how the learning process outlined above can be applied to the MIP building blocks presented in the previous section, we need to ask how to represent the MIP so as ML models can be used. In other words, what data should be used and how? Of course, this question is tailored to the learning task that one wants to solve and to four desirable properties for such representation, namely (1) *Permutation invariance*, i.e., permuting the order of the variables and/or constraints should leave the representation unchanged; (2) *Scale invariance*, i.e., it is preferred to keep values within controlled ranges, which helps the learning process; (3) *Size invariance*, i.e., the size of the representation should not depend on the size of the instance; and (4) *Low computational cost*, i.e., low cost of extracting, storing and processing data.

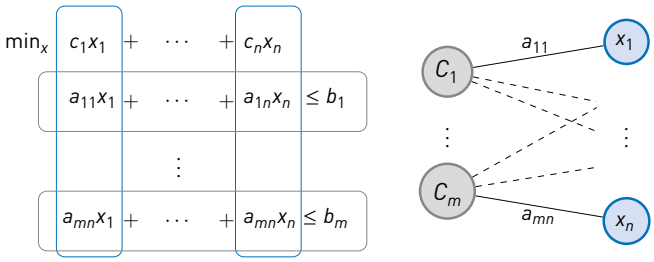

Figure 2. The bipartite graph representation of a MIP (Picture courtesy of Scavuzzo et al. (2024))

Three main representation trends have been established in the literature and characterized by Scavuzzo et al. (2024). The tradeoff that needs to be found is associated with the desirable properties just discussed.

- Khalil et al. (2016) use descriptors gathered in a vector of fixed size. Those descriptors aggregate information whose length would otherwise depend on the problem size, e.g., constraint coefficient statistics, min and max ratios of a_{ij} and b_i (for system $Ax \geq b$ in (1)).
- Gasse et al. (2019) use the bipartite graph representation of the constraint matrix A depicted in Figure 2. The advantage is that such representation has a straightforward and effective mapping into the so-called *graph neural networks* (GNNs). Each node of the graph is embedded in a vector of fixed size with all relevant characteristics of the node (either variable or constraint) and the GNN recursively embeds the vectors associated with neighborhood nodes so as capture the structure of the matrix. Such a recursion is based on aggregation and combination often obtained by a feed-forward NN and one iteration of the process is called *message passing*.[10]
- Zarpellon et al. (2021) take a somehow different approach focusing on the importance of historical information collected in the B&B tree (during execution). Thus, the focus is on tree characteristics, e.g., average branching depth, pseudocosts statistics, conflicts, etc.

We are finally ready to review some of the work that has been done so far to improve the MIP solving building blocks as presented in Section 2 though not exactly in that order.

3.1 Preprocessing/Configuration

This is so far the area registering the highest ML impact on the solvers. The reason is that many of the preprocessing/configuration decisions can be cast as classification and supervised learning benefits from good experts and a lot of data. The first work in the literature is probably that of Kruber et al. (2017) where the authors looked at the decision of applying or not Dantzig–Wolfe decomposition to a given MIP. Instead, the first work integrated within a MIP solver has been that of Bonami et al. (2018, 2022) concerning linearization of mixed-integer *quadratic* programming problems (MIQPs), where the objective function of formulation (1) is quadratic. For MIPQs, in most of the cases, the MIP technology allows to linearize the quadratic objective function (at the price of adding extra variables and constraints (McCormick, 1976)) and reformulate the problem as a MILP. We do not have a mathematical understanding of when linearizing is a goop idea but the effect of doing or not doing that can be dramatic in terms of computing time to solve (or not solving at all) the resulting instance. Bonami et al. (2018, 2022) suggested to simply make the data speak, i.e., they collected some significant amount of data by solving each instance with and without linearization (the best version becomes the artificial expert to imitate) and they trained a *support vector machine* model to predict if the linearizing is a good idea within the commercial MIP solver CPLEX IBM (2023). This methodology has been implemented and is currently the default method of choice for CPLEX since version 12.10. Since that remarkable success, ML approaches for preprocessing and configuration were applied to several other decisions like which simplex pricing rule to use (Hendel et al., 2019), which scaling method to apply (Berthold and Hendel, 2021), if using or not local cuts (Berthold et al., 2022), etc.

3.2 Branching

This is by far the area in which more work has been devoted, i.e., to improve branching by ML approaches. The reason is that branching is a fundamental component that, on the one side, we still do not understand very well, i.e., for which we do not have a clear mathematical characterization, and, on the other side, the most effective heuristic rules we have are computationally very expensive. Several types of decisions connected to branching have to be made, but the one is currently considered the most crucial is that of selecting at each node the variable to branch on, known as *variable selection*. Indeed, many variables x_j, $j \in \mathcal{A} \cup \mathcal{B}$ are likely to take a fractional value in the LP relaxation of any node and to guarantee completeness to the branch-and-bound scheme it is sufficient to select one. However, in order to close the gap between primal and dual bounds fast (see Figure 1) selecting an effective variable is crucial. For this purpose, the most effective heuristic, called *strong branching*, simulates the branching operation on each one of the candidate (fractional) variables and selects that whose effect in terms of dual bound improvement is stronger. It is easy to see that if implemented in this way, strong branching is far too expensive, so MIP solvers are using several work limits (reduced set of candidate variables, LPs not solved to optimality, etc.) and they use strong branching only at the beginning of tree search exploration while later settling for using pseudocosts, essentially the average of the effect of branching on each variable within the tree.

Then, it was natural since the middle of the 2010 decade to look into approximating strong branching through ML. This has been the goal of a few influential papers (Alvarez et al., 2017, 2016; Khalil et al., 2016) that have shown that learning was possible in that context by exploiting various supervised learning approaches.

Though, the first paper able to produce results competitive to the state-of-the-art solvers was that of Gasse et al. (2019). There, the use of GNNs associated with the bipartite graph representation of the constraint matrix (see Figure 2) turned out to be especially effective to capture the effect of branching on a variable with respect to constraints and in connection with the neighboring variables, i.e., those involved in the same constraints. The work of Gasse et al. (2019) has been influential not only for branching (with significant follow up work), but also to highlight the GNNs effectiveness for MIP, with many more examples of their use for other MIP learning tasks.

Several other tasks associated with the tree search design and exploration have been considered in the literature, for example, *node selection* (Labassi et al., 2022; Yilmaz and Yorke-Smith, 2021), though the most intriguing of the current (and future) research directions is that of going beyond imitating strong branching, for example through reinforcement learning (Etheve et al., 2020; Scavuzzo et al., 2022) or incorporating the information collected withing tree search (Zarpellon et al., 2021).

3.3 Primal Heuristics

The area of primal heuristics has also been a very active playground for ML-augmented MIP. Conceptually, the methodologies designed for this task can be split into three main categories: (a) guiding a heuristic search with a starting predicted solution, (b) solution improvement via a learned neighborhood selection criterion, and (c) learning a schedule to pre-existing heuristic routines. For (a) and (b), the important concept is the one of *large neighborhood search* (LNS): Optimize an auxiliary MIP of smaller size constructed by reducing the feasible region of the original MIP through fixing the value of some of the variables and optimizing the rest.[11] For (a), the idea is to produce a (partial) assignment of the binary variables in a binary or mixed-binary MIP (i.e., $\mathcal{B} \neq \emptyset$) that can then be used to guide the search, see, e.g., Ding et al. (2020). Often, this is obtained by starting from a set of collected solutions. For (b), the goal is to identify substructures of the problem that can be used to decompose it into smaller, more manageable sub-MIPs, see, e.g., Song et al. (2020). Finally, for (c), the idea is to decide within the arsenal of primal heuristics that any solver has (tens of primal heuristics) the order in which they have to be executed and for how long, see, e.g., Chmiela et al. (2021).

3.4 Cutting Planes Generation

As discussed in the previous section, cuts are valid inequalities that separate the optimal solution of an LP relaxation from the convex hull of feasible MIP solutions. MIP solvers have a variety of algorithms that produce valid cuts, so at each round many cuts could be potentially added to the LP relaxation with the drawback already anticipated of making the LP too large and numerically unstable. However, as for branching, there is no mathematical understanding of which cuts are stronger than others, especially because their effect is likely not individual but as a group. MIP solvers have internal cut selection routines that are designed to limit this issue but, worse than for branching, there is nothing like strong branching, i.e., there is no expert to imitate even if heuristic. This is the reason why the significant effort to use ML in the context of cuts selection is admittedly still at the demonstration level and has been concentrated on single-cut selection, i.e., deciding which single cut among the group generated in a round should be added. In this way, the problem can be framed as a Markov decision process and two approaches have been considered. Namely, Paulus et al. (2022) used imitation learning and essentially their expert is the extension of the strong branching idea to cuts: the effect of each cut is simulated by solving an LP, while Tang et al. (2020) used reinforcement learning, which allows to potentially go beyond the greedy look-ahead of one step at the price of more complex training convergence due to the size of the action space and the sparsity of the reward function.[12]

4 Limitations, Open Problems and Opportunities

The work reviewed in this paper has given in a relatively short amount of time significant evidence that the use of statistical learning can have very positive effects on the MIP technology. Some of the methods that have been designed are already used by the commercial and noncommercial solvers, which is remarkable. Some others, for example the work on branching, have shown strong potential but there is still a gap to be closed before an implementation into the solvers became possible. There are two main reasons that are slowing down this adoption phase.

First, training is often performed by using data associated with specific classes of MIP instances, which leads to lack of *models' generalization*, i.e., the ability of a model to effectively work outside of the distribution it has been trained on. As discussed, the GNN architecture used for branching captures local information about the formulation, so it is inherently biased towards exploiting the specific characteristics of a MIP class. This is its strength and it allows to provide an effective approximation of strong branching without performing it, provided that the GNN model is trained separately for each MIP class it should be applied to. In other words, the GNN model in Gasse et al. (2019) trained on, say, set covering instances accurately predicts (strong) branching on instances of the same class (and nicely generalizes with respect to the size of the instances) but performs poorly on, say, facility location problems. This behavior limits the adoption of a GNN predictor for branching inside a MIP solver because, in general, MIP solvers are designed to run the same algorithm – in the so-called "default mode" – for any MIP instance they receive in input. One could argue that such a default-mode requirement is too strong in practice, for example because a company using MIP technology tends to solve over and over the same class of problems, not changing class every day, but currently this is the way MIP development works. Thus, it is an open problem to achieve generalization to the entire class of MIP instances, which would likely require to design ML models using an effective combination between data coming from the MIP formulations (as done by GNNs) and from the branch-and-bound execution, potentially combining offline and online learning (Khalil et al., 2016; Zarpellon et al., 2021).

The second reason that is affecting the adoption of ML in MIP software is technological: many of the methods that have proved successful so far use NNs whose training and deployment is especially effective on GPUs. This is a problem because, instead, MIP technology runs on CPUs. Training NNs on GPUs would not be a major limitation because, at least in the case of supervised learning, training generally happens offline. However, using NNs at inference time, for example at any node of the branch and bound, requires the

exchange of information between these two computing architectures, which is somehow cumbersome and inefficient. So, the open problem is twofold: on the methodological side, designing ML models that parsimoniously use NNs and can efficiently run on CPUs (see, for example, Gupta et al. (2020)), and, on the technological side, designing more efficient CPU/GPU communication.

We conclude the paper by mentioning that, besides the evidence already acquired of the benefits of augmenting MIP technology by statistical learning methods, there are two opportunities that provide even more hope to pursue this research direction.

On the one side, the MIP technology has started to use more and more the so-called *restarts*, i.e., the algorithm's execution is stopped because lack of progress is detected and restarted with a different configuration of the algorithmic parameters in the attempt of making it more effective. Currently, this restart mechanism can be seen as a classifier: if the solver in default mode has not been able to solve the instance at hand with a sort of predefined effort, then the instance is labeled "difficult" and another solver configuration is used. To the best of our knowledge, however, there is no sophisticated use of the data collected during this online learning phase, which looks like a wasted opportunity. Furthermore, the restart scheme provides a clear mechanism to collect data for each single instance, so the opportunity is to design a collection that is targeted to specific algorithmic approaches to be applied after the difficulty of the instance has been assessed. In other words, the open question is what data to collect, so that a better (configuration of the) algorithm can be executed, and which one.

On the other side, there is remarkable appetite in industrial applications for making MIP solvers more effective in *reoptimizing* an instance that has changed only minimally, for example, with slight modifications to the demand in unit commitment (Álinson S. Xavier, 2020) or vehicle routing (Morabit et al., 2024) problems. In these cases, it is conceivable that some portion of the solution process, including the solution itself, can be reused, instead of executing any algorithm from scratch. Once again, this clearly calls for exploiting data and augmenting MIP through modern statistical learning.

Andrea Lodi, Jacobs Technion-Cornell Institute,
Cornell Tech and Technion – IIT,
2 West Loop Road, New York, NY 10044, USA
andrea.lodi@cornell.edu

Notes

1. Two more general surveys about the interplay are presented by Bengio et al. (2021) and Kotary et al. (2021).
2. This notation is relatively standard, see, e.g., Wolsey (2020), to which we refer for a general introduction to MIP.
3. Redundancy might come from formulating (1) through modeling languages and appears when data is revealed.
4. The reader is referred to Savelsbergh (1994) for a structured overview of preprocessing.
5. The reader is referred to Cornuéjols (2008) for cutting planes in MIP.
6. A structured overview of branching schemes is presented by Linderoth and Savelsbergh (1999).
7. The forthcoming book by Berthold et al. (2025) is fully devoted to primal heuristics for MIP.
8. Of course, nonconvexity is present in general due to integrality requirements but that type is special and somehow "easy" to treat by the described relaxation mechanism.
9. In other words, differently from the classical ML cases in games, image recognition, etc. the expert is not a human.
10. The reader is referred to Cappart et al. (2023) for a detailed overview of GNNs and their impact in CO.
11. The reader is referred to Berthold et al. (2025) for details on LNS primal heuristics.
12. The reader is refereed to Deza and Khalil (2023) for a detailed survey on ML for cut generation.

References

Álinson S. Xavier, Feng Qiu, S. A. (2020). Learning to solve large-scale security-constrained unit commitment problems. *INFORMS Journal on Computing*, 33(2):739–756.

Alvarez, A. M., Louveaux, Q., and Wehenkel, L. (2017). A machine learning-based approximation of strong branching. *INFORMS Journal on Computing*, 29(1):185–195.

Alvarez, A. M., Wehenkel, L., and Louveaux, Q. (2016). Online learning for strong branching approximation in branch-and-bound. Working paper.

Bengio, Y., Lodi, A., and Prouvost, A. (2021). Machine learning for combinatorial optimization: a methodological tour d'horizon. *European Journal of Operational Research*, 290(2):405–421.

Berthold, T., Francobaldi, M., and Hendel, G. (2022). Learning to use local cuts.

Berthold, T. and Hendel, G. (2021). Learning to scale mixed-integer programs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 3661–3668.

Berthold, T., Lodi, A., and Salvagnin, D. (forthcoming, 2025). *Primal Heuristics in Integer Programming*. Cambridge University Press.

Bonami, P., Lodi, A., and Zarpellon, G. (2018). Learning a classification of mixed-integer quadratic programming problems. In *Procedings of the International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research (CPAIOR)*, volume 15, pages 595–604. Springer.

Bonami, P., Lodi, A., and Zarpellon, G. (2022). A classifier to decide on the linearization of mixed-integer quadratic problems in CPLEX. *Operations Research*, 70(6):3303–3320.

Cappart, Q., Chételat, D., Khalil, E. B., Lodi, A., Morris, C., and Velic□kovic□, P. (2023). Combinatorial optimization and reasoning with graph neural networks. *Journal of Machine Learning Research*, 24(130):1–61.

Chmiela, A., Khalil, E., Gleixner, A., Lodi, A., and Pokutta, S. (2021). Learning to schedule heuristics in branch and bound. *Advances in Neural Information Processing Systems*, 34:24235–24246.

Cornuéjols, G. (2008). Valid inequalities for mixed integer linear programs. *Mathematical Programming*, 112(1):3–44.

Deza, A. and Khalil, E. B. (2023). Machine learning for cutting planes in integer programming: A survey. In *International Joint Conference on Artificial Intelligence*, volume 32, pages 6592–6600. ijcai.org.

Ding, J.-Y., Zhang, C., Shen, L., Li, S., Wang, B., Xu, Y., and Song, L. (2020). Accelerating primal solution findings for mixed integer programs based on solution prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 1452–1459.

Etheve, M., Alès, Z., Bissuel, C., Juan, O., and Kedad-Sidhoum, S. (2020). Reinforcement learning for variable selection in a branch and bound algorithm. In *Proceedings of the International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research (CPAIOR)*, volume 17, pages 176–185. Springer.

FICO (2023). FICO Xpress Optimizer.

Gamrath, G., Anderson, D., Bestuzheva, K., Chen, W.-K., Eifler, L., Gasse, M., Gemander, P., Gleixner, A., Gottwald, L., Halbig, K., et al. (2020). The SCIP optimization suite 7.0.

Gasse, M., Chételat, D., Ferroni, N., Charlin, L., and Lodi, A. (2019). Exact combinatorial optimization with graph convolutional neural networks. *Advances in Neural Information Processing Systems*, 32.

Gomory, R. E. (1958). Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Society*, 64:275–278.

Gomory, R. E. (1960). An algorithm for the mixed integer problem. Technical Report RM-2597, The Rand Corporation.

Gupta, P., Gasse, M., Khalil, E., Mudigonda, P., Lodi, A., and Bengio, Y. (2020). Hybrid models for learning to branch. *Advances in neural information processing systems*, 33:18087–18097.

Gurobi Optimization, LLC (2023). Gurobi Optimizer Reference Manual.

Hendel, G., Miltenberger, M., and Witzig, J. (2019). Adaptive algorithmic behavior for solving mixed integer programs using bandit algorithms. In *Operations Research Proceedings 2018*, pages 513–519. Springer.

IBM (2023). IBM ILOG CPLEX Optimizer.

Khalil, E., Le Bodic, P., Song, L., Nemhauser, G., and Dilkina, B. (2016). Learning to branch in mixed integer programming. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30(1), pages 724–731.

Kotary, J., Fioretto, F., Hentenryck, P. V., and Wilder, B. (2021). End-to-end constrained optimization learning: A survey. In *International Joint Conference on Artificial Intelligence*, volume 30, pages 4475–4482. ijcai.org.

Kruber, M., Lübbecke, M. E., and Parmentier, A. (2017). Learning when to use a decomposition. In *Proceedings of the International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research (CPAIOR)*, volume 14, pages 202–210. Springer.

Labassi, A. G., Chételat, D., and Lodi, A. (2022). Learning to compare nodes in branch and bound with graph neural networks. *Advances in Neural Information Processing Systems*.

Land, A. H. and Doig, A. G. (1960). An automatic method of solving discrete programming problems. *Econometrica*, 28:497–520.

Linderoth, J. T. and Savelsbergh, M. W. (1999). A computational study of search strategies for mixed integer programming. *INFORMS Journal on Computing*, 11:173–187.

McCormick, G. P. (1976). Computability of global solutions to factorable nonconvex programs: Part I—convex underestimating problems. *Mathematical Programming*, 10(1):147–175.

Morabit, M., Desaulniers, G., and Lodi, A. (2024). Learning to repeatedly solve routing problems. *Networks*, 83(3):503–526.

Paulus, M. B., Zarpellon, G., Krause, A., Charlin, L., and Maddison, C. (2022). Learning to cut by looking ahead: Cutting plane selection via imitation learning. In *International Conference on Machine Learning*, pages 17584–17600. PMLR.

Savelsbergh, M. P. (1994). Preprocessing and probing techniques for mixed integer programming problems. *ORSA Journal on Computing*, 6:445–454.

Scavuzzo, L., Aardal, K., Lodi, A., and Yorke-Smith, N. (2024). Machine learning augmented branch and bound for mixed integer linear programming. arXiv:2402.05501.

Scavuzzo, L., Chen, F. Y., Chételat, D., Gasse, M., Lodi, A., Yorke-Smith, N., and Aardal, K. (2022). Learning to branch with tree MDPs. *Advances in Neural Information Processing Systems*.

Smith, K. A. (1999). Neural networks for combinatorial optimization: A review of more than a decade of research. *INFORMS Journal on Computing*, 11(1):15–34.

Song, J., Lanka, R., Yue, Y., and Dilkina, B. (2020). A general large neighborhood search framework for solving integer linear programs. *Advances in Neural Information Processing Systems*, 33:20012–20023.

Tang, Y., Agrawal, S., and Faenza, Y. (2020). Reinforcement learning for integer programming: Learning to cut. In *International Conference on Machine Learning*, pages 9367–9376. PMLR.

Wolsey, L. A. (2020). *Integer Programming*. John Wiley & Sons.

Yilmaz, K. and Yorke-Smith, N. (2021). A study of learning search approximation in mixed integer branch and bound: Node selection in SCIP. *AI*, 2(2):150–178.

Zarpellon, G., Jo, J., Lodi, A., and Bengio, Y. (2021). Parameterizing branch-and-bound search trees to learn branching policies. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 3931–3939.

Combinatorial Optimization Augmented Machine Learning: A Brief Introduction with Applications in Operations Research

Axel Parmentier

Combinatorial optimization augmented machine learning (COAML) is a novel and rapidly growing field that integrates methods from machine learning and combinatorial optimization to tackle data-driven problems that involve both uncertainty and combinatorics. These problems arise frequently when firms seek to leverage large and noisy datasets to better optimize their operations. COAML typically involves embedding combinatorial optimization layers into neural networks and training them with decision-aware learning techniques. This article provides an overview of the field, covering its main applications, algorithms, and theoretical foundations. We notably illustrate the techniques with the contribution that won the 2022 EURO meet NeurIPS 2022 vehicle routing competition.

1 Introduction

Using machine learning algorithms to solve combinatorial optimization problems has been a growing trend in the last few years. Consider a generic combinatorial optimization problem

$$\min_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x})} f^0(\boldsymbol{y}, \boldsymbol{x}). \tag{1}$$

Here $\boldsymbol{x}$ denotes an instance in the set $\mathcal{X}$, and $\mathcal{Y}(\boldsymbol{x})$ denotes the set of feasible solutions $\boldsymbol{y}$ of $\boldsymbol{x}$. We focus here in problems with finite but combinatorial $\mathcal{Y}(\boldsymbol{x})$.

The classic approach in combinatorial optimization is to design algorithms that can solve the problem for any instance $\boldsymbol{x}$ in $\mathcal{X}$. Distributional information over $\mathcal{X}$ is typically ignored. This convention is reflected in the notations, and the instance $\boldsymbol{x}$ would typically be omitted in the notation of the objective function f^0.

Machine learning does not consider instances $\boldsymbol{x}$ independently anymore, but consider them as the realizations of a random variable X. We introduce a hypothesis class $\mathcal{H}$ of *policies* $h : \boldsymbol{x} \in \mathcal{X} \longmapsto \boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x})$. Learning means finding the best policy in $\mathcal{H}$, i.e., the policy that minimizes the *risk*

$$\min_{h \in \mathcal{H}} \mathcal{R}(h) = \mathbb{E}_X\big[f^0(h(X), X)\big], \tag{2}$$

where the expectation is taken with respect to the distribution $\mathbb{P}_X$ of X. In machine learning, this distribution is typically unknown, but we have access to a training set $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n$ of instances drawn from $\mathbb{P}_X$. Learning algorithm thus typically minimize the empirical risk

$$\min_{h \in \mathcal{H}} \hat{\mathcal{R}}_n(h) = \frac{1}{n} \sum_{i=1}^{n} f^0(h(\boldsymbol{x}_i), \boldsymbol{x}_i). \tag{3}$$

Learning can also be supervised. The training set may contain pairs $(\boldsymbol{x}_i, \bar{\boldsymbol{y}}_i)$ of instances $\boldsymbol{x}_i$ with a target solution $\bar{\boldsymbol{y}}_i$. One then introduces a loss $\mathcal{L}(\boldsymbol{y}, \bar{\boldsymbol{y}})$ that quantifies the difference between $\boldsymbol{y}$ and $\bar{\boldsymbol{y}}$. The *supervised learning* problem minimizes this loss over the training set

$$\min_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(h(\boldsymbol{x}_i), \bar{\boldsymbol{y}}_i). \tag{4}$$

One critical element for the making such approaches practical is the choice of $\mathcal{H}$. If we do not put any restriction on $\mathcal{H}$, then, given a

new $\boldsymbol{x}$, an optimal policy will return an optimal solution of (1). This requires to solve the resulting combinatorial optimization problem, and there is nothing to learn. We therefore generally restrict ourselves to a set $\mathcal{H} = \{h_{\boldsymbol{w}} : \boldsymbol{w} \in \mathcal{W}\}$ of policies parametrized $\boldsymbol{w}$ in $\mathcal{W}$, where $\mathcal{W}$ is a subset of $\mathbb{R}^{d_{\mathcal{W}}}$ for some $d_{\mathcal{W}}$ in $\mathbb{N}$.

Using a neural network $h_{\boldsymbol{w}}$ with parameters $\boldsymbol{w}$ seems natural. However, classic neural networks which alternate linear regression layers with non-linear but analytical activation functions cannot make prediction in combinatorial spaces out of the box, and a decoder is generally needed to rebuild a solution $\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x})$. A common strategy in the literature consists in embedding such a neural network in a constructive heuristic (Bengio et al., 2021). For instance, on a traveling salesman problem application, the neural network predicts the "best" vertex to add to the current partial tour until a full tour is obtained. Unfortunately, constructive heuristics tend to perform poorly on combinatorial optimization problems and the resulting algorithm is generally not competitive with state-of-the-art combinatorial optimization algorithms.

COAML delegates the reconstruction of $\boldsymbol{y}$ to a combinatorial optimization oracle $\hat{y}$ that solves the linear problem

$$\hat{y}(\boldsymbol{\theta}) = \underset{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x})}{\arg\max}\, \boldsymbol{\theta}^\top \boldsymbol{y}, \tag{5}$$

where we have omitted $\boldsymbol{x}$ in the arguments of $\hat{y}(\boldsymbol{\theta})$ for notational simplicity. The linear objective of (5) presupposes an embedding of $\mathcal{Y}(\boldsymbol{x})$ in $\mathbb{R}^{d(\boldsymbol{x})}$ for some $d(\boldsymbol{x})$ that may depend on $\boldsymbol{x}$. A statistical model $\varphi_{\boldsymbol{w}} : \boldsymbol{x} \in \mathcal{X} \to \varphi_{\boldsymbol{w}}(\boldsymbol{x}) \in \mathbb{R}^{d(\boldsymbol{x})}$ is used to predict the direction $\boldsymbol{\theta}$ for a given $\boldsymbol{x}$. The resulting parametrized family of policies is

$$\mathcal{H}_{\mathcal{W}} = \{h_{\boldsymbol{w}} : \boldsymbol{x} \mapsto \hat{y} \circ \varphi_{\boldsymbol{w}}(\boldsymbol{w}),\ \boldsymbol{w} \in \mathcal{W}\}. \tag{6}$$

Oracle $\hat{y}$ is assumed to be efficient and scalable. Even though (5) is stated as a linear optimization problem, $\hat{y}$ can be any kind of combinatorial optimization oracle: linear or mixed integer linear programming solvers, sorting algorithms, shortest path algorithms, etc. The tremendous efforts spent by the mathematical programming and combinatorial optimization communities to design efficient algorithms for such problems makes such policies widely applicable. The statistical model $\varphi_{\boldsymbol{w}}$ is typically a neural network.

The combinatorial optimization $\hat{y}$ must treated as a layer in the neural network $h_{\boldsymbol{w}}$ to obtain good results. In other words, the learning problems (3) and (4) focus on predicting the right $\boldsymbol{y}$ and not the right direction $\boldsymbol{\theta}$. These methods are called *smart-predict-then-optimize* approaches by Elmachtoub and Grigas (2021), in contrast with the traditional *estimate-then-optimize* approaches, which use a "classic" learning algorithm to train $\varphi_{\boldsymbol{w}}$ to predict the correct $\boldsymbol{\theta}$, ignoring the fact that this prediction will be fed to $\hat{y}$ to provide a solution of (1).

Section 2 details the application settings where COAML is relevant. Section 3 focuses on neural architectures. Supervised learning is detailed in Section 4, while Section 5 deals with risk minimization and theoretical guarantees. We conclude with open problems.

2 Application settings

Learning heuristic for hard combinatorial optimization problems. Industries now have access to huge amount of data. They exploit these data to improve the performance and the resilience of their industrial processes. When optimizing industrial processes, designing algorithms that scale is the key to performance. Indeed, the first source of profits from algorithms that optimize industrial processes come from decreasing marginal costs. Let us illustrate this on a routing application. Even with the best routing algorithm, if there is only three requests to deliver, the truck will do a tour of the city and the cost per request will be high. Having one thousand requests gives the flexibility to serve requests in the same neighborhood with the same truck, which gives low marginal cost per request. However, the algorithm must scale on large instances if we want to reap these benefits. This explains why the combinatorial optimization community has spent so much effort to design scalable algorithms for combinatorial optimization problems, perhaps at the cost of using simplified objectives. Problems with linear objectives like (5) are therefore ubiquitous in combinatorial optimization, and decades of research have lead to practically efficient algorithms for many applications. However, these methods often do not scale anymore when the objective function accounts for more complex phenomena or resilience, which lead to non-linearities or stochastic aspects. In that context, we need a scalable algorithm for a hard to optimize problem like (1), but only have access to scalable algorithms for a simpler problem of the form (5).

Complexity theory tells us that we cannot hope to have a policy $h_{\boldsymbol{w}}$ of the form (6) that performs well for all instances $\boldsymbol{x}$. However, instances encountered in practice have a specific structure, which is encoded in the distribution $\mathbb{P}_{\mathcal{X}}$. On many applications, we can obtain excellent performance in practice, which means a low risk (2).

Example 1. Stochastic vehicle scheduling. The vehicle scheduling problem (VSP) is a classic combinatorial optimization problem where a fleet of vehicles has to serve a set of timed requests. The goal is to build sequences of requests for each vehicle that cover all requests at minimum cost. A typical application is aircraft routing, where airplanes have to operate flights. When the objective is the sum of the arc costs, the VSP can be solved efficiently as a minimum cost flow problem on the graph where vertices are requests and arcs a in A are pairs of requests that can be chained in a vehicle route. Practitioners are interested in solving stochastic versions of the VSP. For instance, in a flight is late, the airplanes that serves it may propagate delay to the next flights is operates. Airline are interested in aircraft routing solutions that minimize the expected cost delay. These stochastic variants are much more challenging computationally, and there is generally no algorithm that scales well enough for industrial applications. Parmentier (2021b) has proposed to use COAML to approximate the stochastic version of VSP by a reparametrized version of the deterministic VSP. As illustrated on Figure 1, a statistical model $\varphi_{\boldsymbol{w}}$ is used to compute linear arc costs $\boldsymbol{\theta} = (\theta_a)_a$ that lead to good solutions of the stochastic problem. A deterministic VSP solver then returns a solution $\boldsymbol{y}$. Such an approach has the industrial advantage of being able to exploit an existing deterministic solver with all the specific industrial constraints that have been implemented in it.

Structured learning. Combinatorial optimization layers have emerged in the structured learning community in the early 2010s. They were state of the art on many vision applications before the advent of deep learning (Nowozin and Lampert, 2011). In that context,

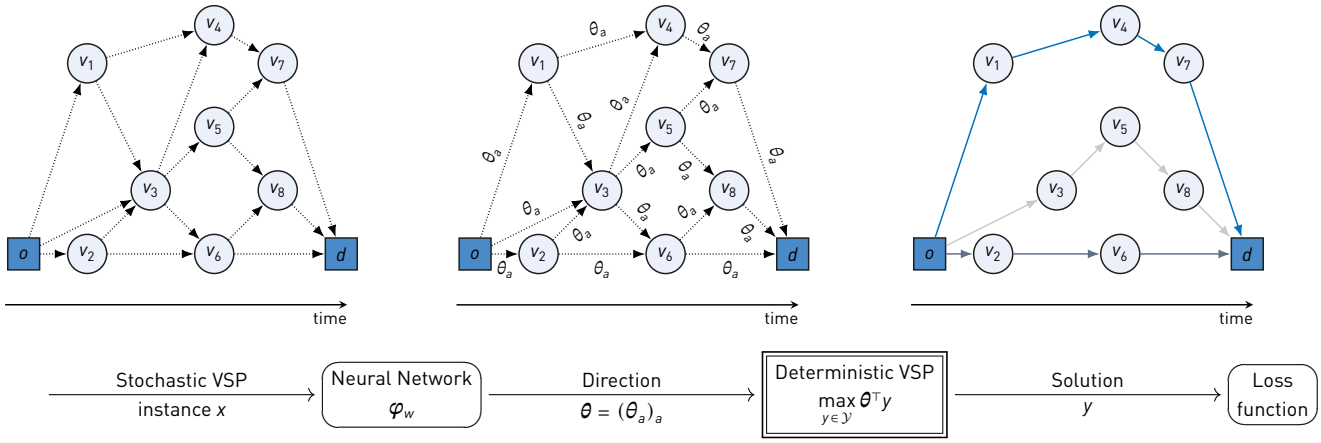

Figure 1. Neural network with combinatorial optimization layer for the stochastic vehicle scheduling problem. Vertices represent requests. Dotted arrows arcs a, which are pairs of requests that can be operated in a sequence. Colored path give vehicle routes in the solution returned. (Image adapted from Dalle et al. (2022))

$\boldsymbol{x}$ is typically an input output pair $(\tilde{\boldsymbol{x}}, \tilde{y})$ drawn from an unknown distribution on $(\tilde{X}, Y)$. Given a new $\tilde{\boldsymbol{x}}$, the goal is to predict the corresponding output $\boldsymbol{y}$. We are clearly in a supervised learning setting, and the goal of the learning problem is to minimize the expectation of a loss $\mathcal{L}$ as in (4). Many of the modern approaches have been pioneered in this community (Berthet et al., 2020; Blondel et al., 2020), which is still very active today on applications such as ranking problems for search engines. Blondel and Roulet (2024) give a recent introduction to the field.

Contextual stochastic optimization. In stochastic optimization, a decision maker has to take a decision $\boldsymbol{y}$ in $\mathcal{Y}$ whose cost $\tilde{f}(\boldsymbol{y}, \boldsymbol{\xi})$ depends upon the realization $\boldsymbol{\xi}$ of an unobserved random noise $\tilde{\xi}$. Sometimes, the decision maker has access to a context $\tilde{X}$ that is correlated with $\tilde{\xi}$. Contextual stochastic optimization (Sadana et al., 2024) exploits this information to improve decisions. The goal is to train a policy $\tilde{h} : \tilde{\boldsymbol{x}} \in \widetilde{\mathcal{X}} \longmapsto \boldsymbol{y} \in \mathcal{Y}$ that minimizes the expected cost $\tilde{f}$

$$\min_{\tilde{h} \in \tilde{\mathcal{H}}} \mathbb{E}_{(\tilde{X}, \tilde{\xi})} \Big[\tilde{f}\big(\tilde{h}(\tilde{X}), \tilde{\xi}\big) \Big]. \tag{7}$$

An optimal policy $\tilde{h}^*$ solves the stochastic optimization problem that results from conditioning $\tilde{\xi}$ on $\tilde{X} = \tilde{\boldsymbol{x}}$.

$$\tilde{h}^*(\tilde{\boldsymbol{x}}) \in \operatorname*{arg\,min}_{\boldsymbol{y} \in \mathcal{Y}} \mathbb{E}\Big[\tilde{f}(\boldsymbol{y}, \tilde{\xi}) \mid \tilde{X} = \tilde{\boldsymbol{x}} \Big]. \tag{8}$$

However, the stochastic optimization (8) problem might be challenging, which prevents its application if decisions must be taken quickly. Furthermore, the joint distribution over $(\tilde{X}, \tilde{\xi})$ is typically unknown, but the decision maker has access to a training set of instances $\tilde{\boldsymbol{x}}_1, \boldsymbol{\xi}_1, \dots, \tilde{\boldsymbol{x}}_n, \boldsymbol{\xi}_n$. Applying (8) would require to estimate the conditional expectation of the cost given the context. Such an estimate-then-optimize approach is referred to as a generative approach in machine learning. COAML is particularly well suited for contextual stochastic optimization because it focuses on policies as those in as in (6) that go space $\mathcal{X}$ to a combinatorial space $\mathcal{Y}$. We just have to define $\boldsymbol{x} = (\tilde{\boldsymbol{x}}, \boldsymbol{\xi})$ and $f^0(\boldsymbol{y}, vx) = \tilde{f}(\boldsymbol{y}, \boldsymbol{\xi})$ and train a policy $h_{\boldsymbol{w}}$ as in (6) to minimize the empirical risk (3).

Such a smart-predict-then-optimize approach that directly predicts $\boldsymbol{y}$ is referred to as a discriminative approach in machine learning, while the estimate-then-optimize approach that calibrates a model predicting $\boldsymbol{\theta}$ is called a generative approach. On combinatorial $\mathcal{Y}$, generative approaches tend to lead to a statistically intractable learning problem (predicting the true $\boldsymbol{\theta}$) followed by a computationally intractable optimization problem (8) (large scale stochastic optimization is difficult). Discriminative policies avoid these two pitfalls and tend to perform better when we take into account the learning aspects.

Remark that nothing prevents us from defining a set of feasible solutions that depends on the context $\tilde{\boldsymbol{x}}$. This enables to bring context to the hard combinatorial optimization problem approximation setting mentioned in the first paragraph. This is very relevant in practice: industrialists often have to solve a variant of a combinatorial optimization problem every day that is also affected by uncertainty, as detailed in the following example.

Example 1 (continued). If we come back to the stochastic vehicle scheduling problem of Example 1, the context $\tilde{\boldsymbol{x}}$ could be the weather forecast, which is correlated with the delay $\tilde{\xi}$ of the flights. The flights to cover, and thus the set of feasible solutions $\mathcal{Y}(\boldsymbol{x})$ may change from one week to another and therefore depend on $\boldsymbol{x}$.

Two-stage stochastic optimization. In the same vein, COAML can be used to model two-stage stochastic optimization problems

$$\min_{\boldsymbol{y} \in \mathcal{Y}} \tilde{f}(\boldsymbol{y}, \tilde{\xi}) \quad \text{where} \quad \tilde{f}(\boldsymbol{y}, \tilde{\xi}) = \operatorname*{arg\,min}_{\boldsymbol{z} \in \mathcal{Z}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi})} c(\boldsymbol{y}, \boldsymbol{z}, \boldsymbol{\xi}).$$

Indeed, such problems can be formulated as special case of stochastic optimization where $\tilde{f}$ is the value of a second stage problem. They can be made contextual. The layer (5) is then typically a linear problem $\max_{\boldsymbol{y} \in \mathcal{Y}} \boldsymbol{\theta}^\top \boldsymbol{y}$ on the first stage decision only (Dalle et al., 2022), or a linear problem $\max_{(\boldsymbol{y}, \boldsymbol{z}) \in \tilde{\mathcal{Z}}} \boldsymbol{\theta}^\top (\boldsymbol{y}^\top, \boldsymbol{z}^\top)^\top$ on the first and second stage decisions if $\mathcal{Z}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi}) = \tilde{\mathcal{Z}}$ does not depend on $\boldsymbol{\xi}$ (Parmentier, 2021a).

Data driven optimization. COAML provides a natural way to model *data driven optimization* problems. Indeed, the neural network $\varphi_{\boldsymbol{w}}$ can be chosen to handle any kind of data as context $\tilde{\boldsymbol{x}}$, as highlighted in the following example.

Example 2. Paths in images. Vlastelica et al. (2020) introduce a benchmark that has been frequently reused in the literature. It consists in computing shortest paths on maps given as images. The neural network φ_w is a convolutional neural network that predicts the arcs cost of a shortest path problem. The combinatorial optimization oracle $\hat{y}$ is Dijkstra algorithm.

Multistage stochastic optimization. Let us conclude this section by a flagship application of COAML in operations research: multistage stochastic optimization problems in a combinatorial setting. Consider a Markov decision process. A system evolves in discrete time steps $t = 0, 1, \dots, T$. At each time step t, the system is in a state $X_t = \boldsymbol{x}_t$ in $\mathcal{X}_t$, and the decision maker has to take a decision $Y_t = \boldsymbol{y}_t$ in $\mathcal{Y}_t$. The system then evolves toward a new state $\boldsymbol{x}_{t+1}$ according to a possibly unknown transition probability $p(X_{t+1}|X_t, Y_t)$. The decision maker incurs a cost $c_t(X_t, Y_t)$ that depends upon the current state and the decision taken. A solution is a policy $h : \boldsymbol{x}_t \in \mathcal{X}_t \longmapsto \boldsymbol{y}_t \in \mathcal{Y}_t$ that minimizes the expected cost.

$$\min_{h \in \mathcal{H}} \mathbb{E}\left[\sum_{t=0}^{T} c_t(X_t, Y_t) \mid h\right]. \tag{9}$$

Such problems can be solved exactly by dynamic programming when $\mathcal{X}_t$ is sufficiently small to be enumerated and optimizing the value function over $\mathcal{Y}_t$ is tractable. The multistage stochastic optimization literature provides tools to handle large $\mathcal{Y}$ when the dimension of $\mathcal{X}$ is moderate, while reinforcement learning provides algorithms when the dimension of $\mathcal{X}$ is large but $\mathcal{Y}$ is small. However, there is not much literature on the case where both $\mathcal{X}$ and $\mathcal{Y}$ are combinatorial. COAML provides a natural way to encode policies in such a setting, as illustrated by the following example.

Example 3. Dynamic vehicle routing. The 2022 EURO meets NeurIPS vehicle routing competition (Kool et al., 2022) focused on a dynamic vehicle routing problem. The problem considers a rapid delivery service, which uses capacitated vehicles to serve customers requests from a depot. Each request must be served within a given time window. Requests arrive dynamically, and vehicles are dispatched in wave to serve them. The state X_t of the system at time t is the set of request that has not been served at wave time t. The decision Y_t is the subset of requests that are served by the vehicles dispatched at time t as well as the route to serve them. The goal is to find a policy h that minimizes the expected routing cost.

Baty et al. (2024) remark that $\mathcal{Y}_t(\boldsymbol{x}_t)$ is the set of feasible solutions of a prize collecting capacitated vehicle routing problem (CVRP), a variant of the CVRP where it is not mandatory to serve requests, but each request v served brings a prize θ_v. The goal is to maximize profit, which is the sum of the prizes collected minus the routing cost. However, if there are natural routing costs in the dynamic VRP, there is no notion of prizes. The authors build a policy h_w illustrated on Figure 2. A neural network φ_w predicts the requests prizes $(\theta_v)_v$, and the resulting prize collecting CVRP is solved to get a solution $\boldsymbol{y}_t$. The resulting policy won the competition with a significant margin.

3 Architectures

When applying COAML to a specific problem, the first task is to design the architecture of the policy h_w. This implies choosing the combinatorial optimization oracle $\hat{y}$ and the statistical model φ_w. The choice of $\hat{y}$ is applications dependent and dictated by the structure of the set of feasible solutions $\mathcal{Y}(\boldsymbol{x})$ and the algorithms that scale on that problem. On the contrary, the choice of φ_w is more generic.

The main difficulty linked to the choice of φ_w is the fact that the dimension $d(\boldsymbol{x})$ of the instances $\boldsymbol{x}$ may vary. That is, the dimension of the input and the output of the neural network φ_w may vary. Most contributions in the literature handle this issue with of the following architectures.

Parallel generalized linear model. A simple but efficient strategy consists in applying the same generalized linear model to all dimensions of the instance $\boldsymbol{x}$. More precisely, we define a feature map $\boldsymbol{\phi}\colon (i, \boldsymbol{x}) \mapsto \boldsymbol{\phi}(i, \boldsymbol{x}) \in \mathbb{R}^{d_W}$ that maps an instance $\boldsymbol{x}$ and dimension i in $\{1, \dots, d(\boldsymbol{x})\}$ to a vector $\boldsymbol{\phi}(i, \boldsymbol{x})$ of fixed dimension. We then define the statistical model φ_w as

$$\varphi_w(\boldsymbol{x}) = (\theta_i)_{i \in \{1,\dots,d(\boldsymbol{x})\}} \quad \text{where} \quad \theta_i = \boldsymbol{w}^\top \boldsymbol{\phi}(i, \boldsymbol{x}). \tag{10}$$

Example 1 (continued). The dimensions of the flow polytope used in the stochastic VSP corresponds to the arcs a of the digraph. We therefore build a vector of features $\boldsymbol{\phi}(a, \boldsymbol{x})$ summarizing the characteristics of the prospective connection a in the instance $\boldsymbol{x}$. Arcs cost are predicted as $\theta_a = \boldsymbol{w}^\top \boldsymbol{\phi}(a, \boldsymbol{x})$ (Parmentier, 2021b).

Parallel neural network. The key idea in the previous technique is to apply in parallel the same statistical model $\boldsymbol{\phi} \mapsto \boldsymbol{w}^\top \boldsymbol{\phi}$ to all the dimensions of the instance $\boldsymbol{x}$. But we can replace the generalized linear model by a neural network $\boldsymbol{\psi}_w$.

$$\varphi_w(\boldsymbol{x}) = (\theta_i)_{i \in \{1,\dots,d(\boldsymbol{x})\}} \quad \text{where} \quad \theta_i = \boldsymbol{\psi}_w \circ \boldsymbol{\phi}(i, \boldsymbol{x}). \tag{11}$$

Graph neural network. A limit of the two previous approaches is that relations between the dimensions of the instance $\boldsymbol{x}$ are not taken into account. More precisely, they can be taken into account only through the feature map $\boldsymbol{\phi}$. When the relations between these dimensions can be encoded by a graph G whose vertices are the i in $\{1, \dots, d(\boldsymbol{x})\}$, one natural way to do this is to use a graph neural network (GNN) ψ_w.

$$\varphi_w(\boldsymbol{x}) = \psi_w(\tilde{\boldsymbol{\phi}}) \quad \text{where} \quad \tilde{\boldsymbol{\phi}} = \big(\boldsymbol{\phi}(i, \boldsymbol{x})\big)_{i \in \{1,\dots,d(\boldsymbol{x})\}}. \tag{12}$$

Example 3 (continued). Baty et al. (2024) won the EURO-NeurIPS challenge using a sparse GNN on the dynamic vehicle routing problem. Requests v are the node of the graph G and edges are pairs of requests that are close geographically and temporally.

4 Supervised learning

We now turn to the supervised learning problem (4). We suppose having a training set of instances $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n$ with target solutions $\bar{\boldsymbol{y}}_1, \dots, \bar{\boldsymbol{y}}_n$, and consider the following learning problem.

$$\min_{w} \sum_{i=1}^{n} \mathcal{L}\left(\hat{y} \circ \varphi_w(\boldsymbol{x}_i), \bar{\boldsymbol{y}}_i\right)$$

Our main task is to design a loss $\mathcal{L}(\boldsymbol{y}, \bar{\boldsymbol{y}})$ on the solution space. Since φ_w is a neural network, we want a loss that can be optimized using stochastic gradient descent.

In this section, we fix an $\boldsymbol{x}$ and omit the dependence on $\boldsymbol{x}$ for notational simplicity. We denote $\mathcal{Y}(\boldsymbol{x})$ by $\mathcal{Y}$, and by d the dimension

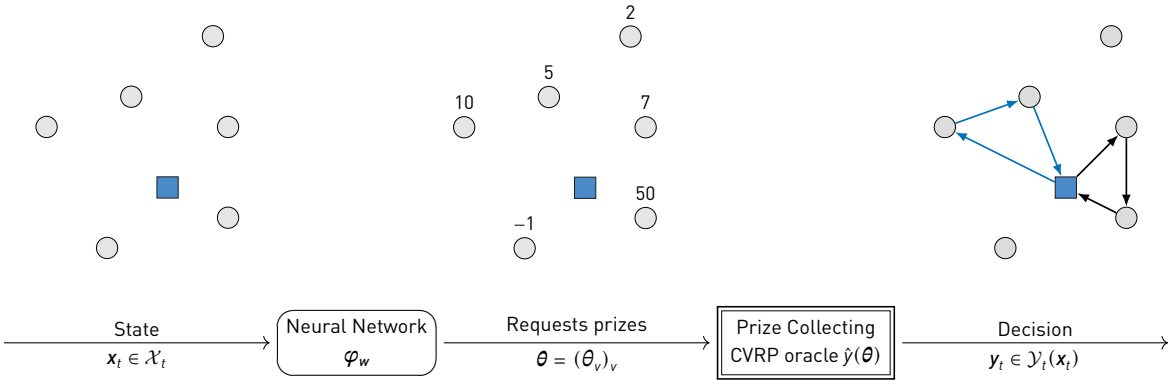

Figure 2. COAML policy for the EURO-NeurIPS dynamic vehicle routing problem. Gray nodes represent request, the blue square the depot, arrows the dispatched routes. (Illustration is a courtesy of Léo Baty.)

of θ. Let $\mathcal{C}$ be the polytope $\text{conv}(\mathcal{Y})$. Remark that since $\hat{y}$ does not depend on w, fixing θ amounts to fixing y. Most approaches therefore define a loss ℓ between the direction θ and the target solution $\bar{y}$, leading to the learning problem

$$\min_w \sum_{i=1}^{n} \ell(\varphi_w(x_i), \bar{y}_i), \tag{13}$$

where $\ell(\theta, \bar{y}) = \mathcal{L}(\hat{y}(\theta), \bar{y})$. The main difficulty when defining such a loss is illustrated on Figure 3. Consider a vertex y of $\mathcal{C}$. The output of $\hat{y}$ is y for any θ in the interior of the normal cone of y. As a consequence, $\hat{y}$ is piecewise constant on each cone of the *normal fan* of $\mathcal{C}$, which is the partition of $\mathbb{R}^d$ into normal cones of the vertices of $\mathcal{C}$. In the structured learning literature, authors often replace this piecewise constant loss by the structured Hinge loss (Nowozin and Lampert, 2011, Chapter 6), a surrogate upper bound that is convex in θ. It has recently enjoyed a new popularity as the SPO+ loss (Elmachtoub and Grigas, 2021). We present here a recent alternative that enjoys nice geometric properties and is convenient on practical applications.

Regularized prediction, Fenchel Young loss, and Bregman divergence. Since maximizing a linear objective on the vertices $\mathcal{Y}$ of a polytope $\mathcal{C} = \text{conv}(\mathcal{Y})$ is equivalent to maximizing it on $\mathcal{C}$, we can replace (5) by

$$\hat{y}(\theta) = \arg\max_{y \in \mathcal{C}} \theta^\top y. \tag{14}$$

The non-differentiability of $\hat{y}$ comes from the fact that the optimal solution may "jump" from one vertex to another when θ crosses the boundary between two normal cones of the normal fan. This behavior disappears when we regularize (14) with a smooth and strictly convex function Ω, which pushes the optimal solution to the interior of the polytope.

$$\arg\max_{y \in \mathcal{C}} \theta^\top y - \Omega(y) \tag{15}$$

When defining a loss ℓ between θ and $\bar{y}$, we would like $\ell(\theta, \bar{y})$ to be small when $\bar{y}$ is "close to" be an optimal solution of (15) for θ. It is then natural to define the *Fenchel Young loss* as the non-optimality of $\bar{y}$ as a solution of (15) for θ.

$$\begin{aligned} \ell_\Omega^{\text{FY}}(\theta, \bar{y}) &= \max_{y \in \mathcal{C}} \big(\theta^\top y - \Omega(y)\big) - \big(\theta^\top \bar{y} - \Omega(\bar{y})\big) \\ &= \Omega^*(\theta) + \Omega(\bar{y}) - \theta^\top \bar{y}, \end{aligned} \tag{16}$$

where $\Omega^*(y) = \max_{y \in \mathcal{C}} \big(\theta^\top y - \Omega(y)\big)$ is the Fenchel conjugate of Ω. Remark that the optimum in (15) is obtained in $\nabla\Omega^*(\theta)$, and that regularizing (14) amounts to replacing $\hat{y}$ by $\nabla\Omega^*$. Figure 4 illustrates the Fenchel Young loss for a simple polytope.

The loss ℓ_Ω^{FY} has all the desirable properties of a loss in ML. It is smooth, convex, and differentiable in θ with gradient.

$$\nabla_\theta \ell_\Omega^{\text{FY}}(\theta, \bar{y}) = \nabla\Omega^*(\theta) - \bar{y}. \tag{17}$$

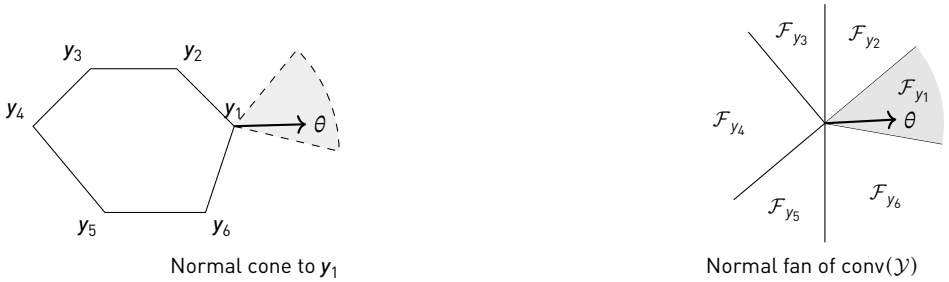

Figure 3. Normal cone to $\bar{y}$ and normal fan of $\mathcal{C}$. Oracle $\hat{y}$ is piecewise constant on each cone of the normal fan.

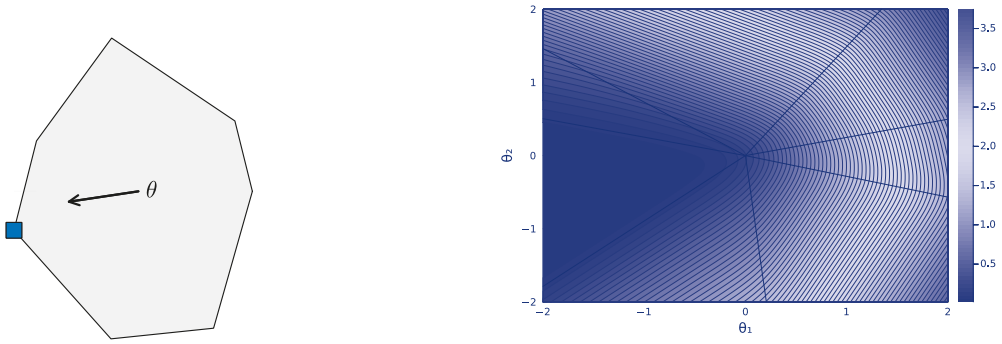

Figure 4. A polytope and the corresponding Fenchel Young loss $\ell_\Omega^{\mathrm{FY}}(\theta, \bar{y})$ value for $\bar{y}$ being the blue square vertex using a regularization by perturbation Ω.

This loss actually nicely captures the geometry induced by (15). Given a smooth and strictly convex Ω, the *Bregman divergence* associated to Ω is defined as

$$\mathcal{D}_\Omega(\bar{y}|y) = \Omega(\bar{y}) - \big(\Omega(y) + \nabla\Omega(y)^\top(\bar{y} - y)\big). \tag{18}$$

It is the difference at $\bar{y}$ between the value of Ω and the value of the tangent of Ω at y, and thus measures the difference between $\bar{y}$ and y. It is a generalization of the squared Euclidean distance, which is obtained when $\Omega(y) = \frac{1}{2}\|y\|^2$. Under the additional assumption that Ω is Legendre-type, the gradient of Ω induces a bijection from $\mathcal{C}$ to $\mathbb{R}^d$, whose inverse is $\nabla\Omega^*$. If in addition $\mathcal{C}$ is plain dimensional, Blondel et al. (2020) have shown that, given y and $\bar{y}$ in $\mathcal{C}$, their duals $\theta = \nabla\Omega^*(y)$ and $\bar{\theta} = \nabla\Omega^*(\bar{y})$, we have

$$\ell_\Omega^{\mathrm{FY}}(\theta, \bar{y}) = \mathcal{D}_\Omega(\bar{y} \mid y) = \mathcal{D}_{\Omega^*}(\theta \mid \bar{\theta}). \tag{19}$$

In other words, the *Fenchel Young loss* is a primal-dual Bregman divergence associated to the linear oracle and

$$\ell_\Omega^{\mathrm{FY}}(\varphi_w(x), \bar{y}) = D_\Omega\big(\overbrace{\nabla\Omega^* \circ \varphi_w(x)}^{\substack{\text{Regularized}\\ \text{neural network}\\ \text{prediction}}} \big| \bar{y}\big).$$

It remains to define regularization functions Ω that lead to good prediction performance and tractable gradients. One natural choice is $\Omega(y) = \frac{1}{2}\|y\|^2$. The regularized prediction (15) then amounts to minimizing the squared Euclidean distance between the prediction and the target solution. It is a convex quadratic optimization problem that is tractable for some polytope $\mathcal{C}$. However, on some $\mathcal{C}$, only the non-regularized linear optimization oracle $\hat{y}$ is available. We now introduce an Ω that requires only calls to $\hat{y}$.

Perturbation. Let us define the perturbed linear prediction

$$F(\theta) = \mathbb{E}\Big[\arg\max_{y \in \mathcal{C}}(\theta + \varepsilon Z)^\top y\Big], \tag{20}$$

where $\varepsilon > 0$ is and Z is a standard Gaussian vector on $\mathbb{R}^d$. Berthet et al. (2020) suggest using the Fenchel dual of F as regularization Ω. They show that F is smooth and strictly convex, and that Ω is Legendre-type, hence F is its own bidual and $\Omega^* = F$. Danskin's theorem then gives

$$\nabla\Omega^*(\theta) = \nabla F(\theta) = \mathbb{E}\Big[\arg\max_{y \in \mathcal{Y}}(\theta + Z)^\top y\Big].$$

Unbiased estimates of $\nabla\Omega^*$ and hence of $\nabla_\theta \ell_\Omega^{\mathrm{FY}}(\theta, \bar{y})$ can be obtained by sampling Z and solving the resulting inner maximization problem using $\hat{y}$. This gives a convenient way to solve the learning problem (13) via stochastic gradient descent using only call to $\hat{y}$ to compute the stochastic gradients.

Imitation learning for multistage problem. When considering multistage stochastic optimization problems (9), supervised learning amounts to imitation learning. The policy h_w is trained to imitate a policy h^*. On many operations research applications such as the dynamic vehicle routing problem of Example 3, the transition probability is a deterministic function of the current state, the decision taken, and some noise ξ_t that is not observed at decision time but observed after transition

$$X_{t+1} = f(X_t, Y_t, \xi_t). \tag{21}$$

Having observed a full episode $\boldsymbol{\xi}_1, \dots, \boldsymbol{\xi}_T$, computing a posteriori the optimal trajectory $\boldsymbol{x}_1, \boldsymbol{y}_1, \dots, \boldsymbol{x}_T, \boldsymbol{y}_T$ for this episode is a deterministic problem that can be solved when the episode is over. Let δ^* be the corresponding policy. Such a policy is called *anticipative* and cannot be applied in practice because computing the $\bar{\boldsymbol{y}}_t$ requires to know the future $\boldsymbol{\xi}_{t+1}, \dots, \boldsymbol{\xi}_T$. However, it can be recomputed a posteriori to generate a training set $\boldsymbol{x}_1, \boldsymbol{y}_1, \dots, \boldsymbol{x}_n, \boldsymbol{y}_n$. The supervised learning problem (13) then leads to a non-anticipative and thus applicable policy h_w that imitates h^* (Baty et al., 2024). The approach can be further improved using off-policy learning techniques (Greif et al., 2024).

5 Risk minimization and generalization guarantees

Designing algorithms for the empirical risk minimization problem (3) is challenging because its objective is piecewise constant. Using black-box global optimization solvers enabled to show that risk minimization can lead to excellent h_w (Parmentier, 2021a). However, such solvers are not deep learning compatible and can be applied only to generalized linear models with $\boldsymbol{w}$ of moderate dimension. Structured Hinge losses are applicable to deep learning and have been applied in that context (Elmachtoub and Grigas, 2021). Unfortunately, they lead to tractability issues when f^0 is not linear in $\boldsymbol{y}$. Designing scalable and deep learning compatible risk minimization algorithms is therefore an important open problem for COAML.

Beyond requiring no target $\bar{\boldsymbol{y}}$ in the training set, one advantage of risk minimization is that it provides guarantees on the solution returned. Let us denote by $R^* = \min_{h \in \mathcal{H}} \mathcal{R}(h)$ the risk of the best policy where $\mathcal{H}$ contains all possible policies (not necessarily based

on COAML), and by $R^\dagger = \min_{\boldsymbol{w}} \mathcal{R}(h_{\boldsymbol{w}})$ the risk of the best policy in the class $\mathcal{H}_{\mathcal{W}}$. The gap

$$R^* - R^\dagger \geq 0$$

is the *model bias* and can only be improved by changing the class $\mathcal{H}_{\mathcal{W}}$. Demelas et al. (2024) provide a first attempt that exploits deep learning and Lagrangian duality to bound this model bias.

The goal of the learning algorithm is to find a $\boldsymbol{w}$ such that $\mathcal{R}(h_{\boldsymbol{w}})$ is as close as possible to $R^\dagger$. Aubin-Frankowski et al. (2024) prove an upper bound on the excess risk $\mathcal{R}(h_{\hat{\boldsymbol{w}}_n}) - R^\dagger$ of the $\hat{\boldsymbol{w}}_n$ returned by their learning algorithm that stands with high probability on the drawing of the training set $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n$. They also show that $\mathcal{R}(h_{\hat{\boldsymbol{w}}_n}) - R^\dagger$ converges to 0 with the size of the training set. Parmentier (2021a) shows that on some problem, the model bias can be bounded, and thus $h_{\hat{\boldsymbol{w}}_n}$ is an approximation algorithm where the approximation ratio is controlled in expectation over $\mathbb{P}_X$.

6 Open problems

COAML is a young field that has already shown its potential on a wide range of applications. However, many open problems remain. On the architecture side, we focused on learning the objective direction $\hat{\boldsymbol{\theta}}$. Learning constraints seems relevant for many applications, and not much has been done on this matter. On the learning side, designing scalable and deep learning compatible risk minimization algorithms is a key challenge. One challenge that is both theoretical and practical is the design of learning algorithms with good generalization guarantees and that bound the model bias and the excess risk.

Axel Parmentier, CERMICS, ENPC
Institut Polytechnique de Paris, 6 et 8 av Blaise Pascal, Cité Descartes, Champs sur Marne, 77455 Marne-la-Vallée, France
axel.parmentier@enpc.fr

References

Aubin-Frankowski, P.-C., De Castro, Y., Parmentier, A., and Rudi, A. (2024). Generalization bounds of surrogate policies for combinatorial optimization problems. arXiv:2407.17200.

Baty, L., Jungel, K., Klein, P. S., Parmentier, A., and Schiffer, M. (2024). Combinatorial Optimization-Enriched Machine Learning to Solve the Dynamic Vehicle Routing Problem with Time Windows. *Transportation Science*.

Bengio, Y., Lodi, A., and Prouvost, A. (2021). Machine learning for combinatorial optimization: A methodological tour d'horizon. *European Journal of Operational Research*, 290(2):405–421.

Berthet, Q., Blondel, M., Teboul, O., Cuturi, M., Vert, J.-P., and Bach, F. (2020). Learning with differentiable perturbed optimizers. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, pages 9508–9519.

Blondel, M., Martins, A. F. T., and Niculae, V. (2020). Learning with fenchel-young losses. *J. Mach. Learn. Res.*, 21:35:1–35:69.

Blondel, M. and Roulet, V. (2024). The Elements of Differentiable Programming. arXiv:2403.14606.

Dalle, G., Baty, L., Bouvier, L., and Parmentier, A. (2022). Learning with combinatorial optimization layers: a probabilistic approach. arXiv2207.13513.

Demelas, F., Roux, J. L., Lacroix, M., and Parmentier, A. (2024). Predicting lagrangian multipliers for mixed integer linear programs. In *Forty-first International Conference on Machine Learning*.

Elmachtoub, A. N. and Grigas, P. (2021). Smart "Predict, then Optimize". *Management Science*.

Greif, T., Bouvier, L., Flath, C. M., Parmentier, A., Rohmer, S. U., and Vidal, T. (2024). Combinatorial optimization and machine learning for dynamic inventory routing. arXiv:2402.04463.

Kool, W., Bliek, L., Numeroso, D., Reijnen, R., Afshar, R. R., Zhang, Y., Catshoek, T., Tierney, K., Uchoa, E., Vidal, T., and Gromicho, J. (2022). The EURO meets NeurIPS 2022 vehicle routing competition.

Nowozin, S. and Lampert, C. H. (2011). Structured Learning and Prediction in Computer Vision. *Foundations and Trends® in Computer Graphics and Vision*, 6(3–4):185–365.

Parmentier, A. (2021a). Learning structured approximations of operations research problems. arXiv:2107.04323.

Parmentier, A. (2021b). Learning to Approximate Industrial Problems by Operations Research Classic Problems. *Operations Research*.

Sadana, U., Chenreddy, A., Delage, E., Forel, A., Frejinger, E., and Vidal, T. (2024). A survey of contextual optimization methods for decision-making under uncertainty. *European Journal of Operational Research*.

Vlastelica, M., Paulus, A., Musil, V., Martius, G., and Rolinek, M. (2020). Differentiation of Blackbox Combinatorial Solvers.

MOS Awards 2024

Miguel Anjos

Thank you to all who attended the ISMP Opening Ceremony and Awards Ceremony. For those who missed it, here is the list of awardees.

Mathematical Programming Computation Outstanding Paper of the Year 2022

Jordan Jalving, Sungho Shin, and Victor M. Zavala, "A graph-based modeling abstraction for optimization: Concepts and implementation in Plasmo.jl."

Mathematical Programming Computation Outstanding Paper of the Year 2023

Daniel Rehfeldt, Thorsten Koch, and Yuji Shinano, "Faster exact solution of sparse MaxCut and QUBO problems."

Paul Y. Tseng Memorial Lectureship in Continuous Optimization 2024

Awarded to Kim-Chuan Toh:

> *Kim-Chuan Toh has made many outstanding contributions in the area of continuous optimization including development of very successful software, original contributions to design and implementation of algorithms as well as original contributions to theory. In particular, Kim-Chuan has made indispensable contributions to solution methods for semidefinite programming and by extension to solving convex optimization problems more broadly.*
>
> *In addition to his high-quality and high-impact contributions in continuous optimization, Kim-Chuan has made exemplary contributions in leadership and in service to the optimization community and to promotion of research and applications of continuous optimization in the Asia-Pacific region.*

A. W. Tucker Prize 2024

Awarded to Yang Liu for the Thesis *Sparsification, Online Optimization, and an Almost-Linear Time Algorithm for Maximum Flow*:

> *This thesis is a tour de force that tackles multiple fundamental optimization questions from unique and modern angles. First, Liu gives an almost-linear-time algorithm that computes exact maximum- and minimum-cost flows in directed graphs. Second, he offers methods with nearly optimal complexity properties for solving l_p-norm regression problems, leveraging new sparsification techniques. Finally, Liu considers online discrepancy minimization: given a sequence of T n-dimensional vectors with norm ≤ 1, the linear-time algorithm succeeds with high probability to choose signs such that the signed sum of vectors is below $O(\log nT)$.*

A. W. Tucker Prize 2024 Finalists

Jason Altschuler is a finalist for The A. W. Tucker Prize 2024 for the Thesis *Transport and Beyond: Efficient Optimization Over Probability Distributions*:

> *This thesis studies two optimization problems for which the decision variables are probability distributions. Both parts relate to optimal transport, a geometrically meaningful measure of distance between any pair of probability distributions that has received substantial attention in areas such as machine learning. The first part focuses on optimal transport with so-called entropic regularization. Altschuler's thesis shows that the seemingly distinct problems of optimal transport, min-mean-cycle, matrix balancing, and matrix scaling can all be viewed under the same lens as optimization over a joint probability distribution with similarly constrained marginals. Using this lens, the thesis provides approximation algorithms for all four problems that have nearly optimal run times, namely, close to the n^2 run time required merely to read an n-by-n matrix input. The second part of the thesis goes beyond the setting of optimization over joint probability distributions with two marginals to multi-marginal optimal transport (MOT) problems with $k \geq 2$ marginals. Such problems, which arise in data science, PDE simulation, and quantum chemistry, are difficult in fundamental ways. The thesis addresses multiple questions about the potential polynomial-time solvability of MOT problems. It shows that certain MOT problems are indeed intractable, while also offering a unified algorithmic framework – built upon classical ideas from oracle-based optimization algorithms – for solving certain structured MOT problems.*

Nathan Klein is a finalist for The A. W. Tucker Prize 2024 for the Thesis *Finding Structure in Entropy: Improved Approximation Algorithms for TSP and other Graph Problems*:

> *This thesis revisits a fundamental question in the design of approximation algorithms for solving the traveling salesman problem (TSP). In particular, it revisits the decades-old question of whether it is possible to improve upon the (3/2)-approximation algorithm of Christofides & Serdyukov for metric TSP. Incredibly, after considerable effort by many to answer this question, this*

thesis proves that for at least some small epsilon there exists a deterministic ((3/2)-epsilon)-approximation algorithm for metric TSP, and that the integrality gap of the natural LP relaxation for TSP is at most (3/2)-epsilon. The Christofides & Serdyukov algorithm yields no better than a (3/2)-approximation because it can be saddled with a spanning tree (for initialization) with poor properties. Klein's thesis (1) leverages a "max entropy" approach for choosing a random initial spanning tree based on a carefully chosen distribution and proves that this yields undesirable properties with low probability; and (2) proves new properties about the structure of near-minimum cuts of graphs, which in turn gives a setting in which the prior probabilistic tools can be applied.

Lagrange Prize in Continuous Optimization 2024

Awarded to Jérôme Bolte and Edouard Pauwels for "Conservative set valued fields, automatic differentiation, stochastic gradient methods and deep learning", *Mathematical Programming* **188** (2021):

Bolte and Pauwels model the algorithmic differentiation operation via new variational objects called conservative set valued fields. They also conduct the convergence analysis to critical points for non-smooth stochastic first-order methods. Their paper combines techniques from several fields including non-smooth analysis, first-order stochastic approximation, machine learning, and algorithmic differentiation, with a central role played by o-minimal geometry. Bolte and Pauwels thus provide a remarkable theoretical and algorithmic contribution to the field of continuous optimization.

Beale-Orchard-Hays Prize for Excellence in Computational Mathematical Programming 2024

Awarded to Bartolomeo Stellato, Goran Banjac, Paul Goulart, Alberto Bemporad, and Stephen Boyd for "OSQP: An operator splitting solver for quadratic programs", *Mathematical Programming Computation* **12** (2020):

The committee commends the following aspects of this work: the robust implementation of the software, with attention to issues such as preprocessing problem instance and detecting infeasibility; the long list of users and successful practical applications of the software; the impact the paper has already made on the field, as evidenced by its large number of citations; the careful and extensive numerical testing in the paper.

and to David Applegate, Mateo Díaz, Oliver Hinder, Haihao Lu, Miles Lubin, Brendan O'Donoghue, and Warren Schudy for "Practical large-scale linear programming using primal-dual hybrid gradient", *Advances in Neural Information Processing Systems* **34** (2021), and "Faster first-order primal-dual methods for linear programming using restarts and sharpness", *Mathematical Programming* **201** (2023):

The committee commends the following aspects of this work: its long-term potential to make first-order methods a practical option to solve large-scale linear programming problems; its adaptability to GPUs and other parallel computing architectures; the careful algorithmic engineering work to make the methods practical; and the sophisticated and innovative analysis used to justify and describe the performance of the algorithms.

Delbert Ray Fulkerson Prize 2024

Awarded to Ben Cousins and Santosh Vempala for "Gaussian cooling and $O*(n^3)$ algorithms for volume and Gaussian volume", *SIAM Journal on Computing* **47** (2018):

Computing the volume of a convex body is an ancient challenge. Cousins & Vempala develop a fast algorithm that approximates the volume of a well-rounded convex body while querying membership of a cubic number of points, and also present the fastest algorithms for integrating and sampling from a Gaussian measure restricted to a convex body. The impact of the ideas goes far beyond the two core problems addressed.

and to Nathan Keller and Noam Lifshitz for "The junta method for hypergraphs and the Erdős–Chvátal Simplex conjecture", *Advances in Mathematics* **392** (2021):

The authors give a new approach for solving Turán-type problems, which concern the maximum number of edges a hypergraph can have without containing certain expansions of a given forbidden substructure. The junta method approximates these hypergraphs by juntas, and the authors successfully apply it to the Erdős–Chvátal simplex conjecture, the Erdős–Sós forbidding one intersection problem, and the Frankl–Füredi special simplex problem.

and to Zilin Jiang, Jonathan Tidor, Yuan Yao, Shengtong Zhang, and Yufei Zhao for "Equiangular lines with a fixed angle", *Annals of Mathematics* **194** (2021):

The authors solve a combinatorial geometry problem that has received considerable attention since the 1960s: determine the maximum number of lines in d-dimensional space such that the angle between every pair is exactly θ. For fixed θ and large d, the authors provide a sharp bound. The proofs combine combinatorial ideas with tools from spectral graph theory in a clever, original and elegant way.

George B. Dantzig Prize 2024

Awarded to Stephen J. Wright

The George B. Dantzig Prize is awarded to Stephen J. Wright for his fundamental contributions to nonlinear optimization. He pioneered infeasible interior point methods which culminated in his 1997 SIAM monograph on the subject. Moreover, Stephen Wright contributed highly cited, outstanding, and very influential work in a broad range of fields in mathematical optimization, including algorithms for control, nonsmooth optimization with applications to compressed sensing, machine learning, and data science. His comprehensive contributions range from theory, algorithm design and analysis to applications and the development of high-impact software.

Congratulations!

Miguel Anjos, University of Edinburgh
Chair, Mathematical Optimization Society
miguel.f.anjos@ed.ac.uk

ICCOPT 2025: 8th International Conference on Continuous Optimization

ICCOPT 2025 will be held on the campus of the University of Southern California in Los Angeles, July 21–24, 2025, to be preceded by a 2-day summer school.

Some preliminary details can be found on the website sites.google.com/view/iccopt2025/home. We will continue to upload information of the conference on the site. For now, please note that the conference registration has opened on November 1, 2024.

We look forward to welcoming you to this exciting conference of our field.

Plenary Speakers

- Alexandre d'Aspremont, CNRS and Ecole Normale Supérieure Paris
- Claudia Sagastizábal, IMECC Unicamp Brazil
- Kim-Chuan Toh, National University of Singapore
- Stephen J. Wright, University of Wisconsin

Semi-Plenary Speakers

- Jérôme Bolte, Université Toulouse 1 Capitole
- John C. Duchi, Stanford University
- Maryam Fazel, University of Washington
- Tim Hoheisel, McGill University
- Mingyi Hong, University of Minnesota
- Ruth Misener, Imperial College London
- Anthony Man-Cho So, Chinese University of Hong Kong
- Angelika Wiegele, Alpen-Adria-Universität Klagenfurt

Clusters/Chairs

1. *Nonlinear optimization*
   Liaison: Serge Gratton
   Cluster Chairs: Frank E. Curtis, Oliver Hinder, Stefan Ulbrich
2. *Nonsmooth optimization*
   Liaison: Shimrit Shtern
   Cluster Chairs: Tim Hoheisel, Radu loan Bot, Pedro Perez-Aros
3. *Conic and semi-definite optimization*
   Liaison: Ying Cui
   Cluster Chairs: Kim-Chuan Toh, Angelika Wiegele
4. *Optimization under uncertainty, data driven optimization*
   Liaison: Johannes Royset
   Cluster Chairs: Johannes Royset, Tito Homem-de-Mello, Daniel Kuhn, Junyi Liu
5. *Optimization for data science*
   Liaison: Akiko Takeda
   Cluster Chairs: Akiko Takeda, Alura Palagi, Yuxin Chen
6. *Fixed points and variational inequalities*
   Liaison: Xiaojun Chen
   Cluster Chairs: Xiaojun Chen, Jelena Diakonikolas, Jinlai Shen
7. *Interplay between continous and discrete optimization*
   Liaison: Roberto Cominetti
   Cluster Chairs: Alper Atamturk, Ruth Misener, Daniel Bienstock
8. *Computational software*
   Liaison: Etienne de Klerk
   Cluster Chairs: Cosmin Gheorghita Petra, Defeng Sun
9. *Derivative free optimization*
   Liaison: Veronica Piccialli
   Cluster Chairs: Giampaolo Liuzzi, Ana Luisa Custodio
10. *Global optimization*
    Liaison: Veronica Piccialli
    Cluster Chairs: Marco Locatelli, Sonia Cafieri
11. *Multi-agent optimization and games*
    Liaison: Roberto Cominetti
    Cluster Chairs: Julio Deride, Mingyi Hong, Uday V. Shanbhag
12. *Optimization on manifolds*
    Liaison: Niao He
    Cluster Chairs: Nicolas Boumal, Pierre-Antoine Absil, Shigian Ma
13. *Optimization for emerging technologies (LLM quantum computing)*
    Liaison: Martin Jaggi
    Cluster Chairs: Martin Jaggi, Tamás Terlaky, Ruoyu Sun
14. *PDE-constrained optimization*
    Liaison: Michael Hintermüller
    Cluster Chairs: Michael Hintermüller, Drew Kouri
15. *Optimization applications (communication, energy, health, ML, ...)*
    Liaison: Martin Jaggi
    Cluster Chairs: Lin Xiao, Claudia Sagastizábal

Summer School

We are particularly interested in encouraging talented students and postdoctoral fellows to attend the summer school. Since we have capped the capacity of the school at one hundred count, the earlier we receive your show of interest, the higher the chance we can accept your application to the school.

Lecturers

- Ying Cui, University of California, Berkeley
- Jon Lee, University of Michigan
- Meisam Razaviyayn, University of Southern California
- Johannes O. Royset, University of Southern California

Further information: sites.google.com/view/iccopt2025/home

Jong-Shi Pang and Meisam Razaviyayn
Co-Chairs, ICCOPT 2025

IMPRINT ■ *Editor:* Sebastian Pokutta, Zuse Institute Berlin (ZIB), Takustraße 7, 14195 Berlin, Germany. pokutta@zib.de ■ *Co-Editors:* Swati Gupta, Massachusetts Institute of Technology, Sloan School of Management, 100 Main Street Cambridge, MA 02142, USA, swatig@mit.edu Omid Nohadani, Artificial Intelligence and Data Science, Benefits Science Technologies, Boston, MA 02110, USA, onohadani@gmail.com ■ *Founding Editor:* Donald W. Hearn ■ Published by the Mathematical Optimization Society, www.mathopt.org ■ Design and typesetting by Christoph Eyrich, Berlin, Germany ■ Printed by Oktoberdruck GmbH, Berlin, Germany