Learning to Price : Structured Learning to scale up Column Generation

Axel Parmentier

Ecole des Ponts Paristech, France, axel.parmentier@enpc.fr

Mots-clés : *Column Generation, Stochastic Vehicle Scheduling Problem, Machine learning for Operations Research, Structured learning, Path problems, Flow problems*

In the past few years, machine learning (ML) techniques have become increasingly popular to speed-up the resolution of operations research (OR) problems. Consider a generic optimization problem

$$\min_{x \in \mathcal{X}(\Gamma)} f(x; \Gamma),$$

where Γ is an instance of the problem, $\mathcal{X}(\Gamma)$ is the set of feasible solutions of Γ, and $x \mapsto f(x; \Gamma)$ the objective function of Γ. In contract with the current practice, we mention the instance Γ explicitely because machine learning schemes typically consider a set of instances to extract informations that are relevant for any instances of the problem and not just the instance concerned. Figure 1.a illustrates the general scheme of algorithms that exploit ML techniques to solve OR problems : First, an ML predictor ϕ_θ extracts relevant information $\phi_\theta(\Gamma)$ on instance Γ, and second an optimization algorithm $\mathcal{A}$ uses this information to solve the problem. The ML predictor ϕ_θ belongs to a family $(\phi_\theta)_{\theta \in \Theta}$, and the objective of the learning algorithm is to find the parameter θ in Θ that leads to the best performance of the algorithm $\mathcal{A}$ on the set of instances of interest.

The different methodologies identified by Bengio et al. [1] in their survey differ by the kind of ML predictor ϕ_θ and algorithm $\mathcal{A}$ they use. For instance, end-to-end learning approaches use a deep neural network as ϕ_θ and a simple greedy heuristic as $\mathcal{A}$, while other approaches use a simple features based ML predictor ϕ_θ to find a good parametrization $\phi_\theta(\Gamma)$ of an advanced combinatorial optimization solver $\mathcal{A}$. In this work, we consider the following situation, that we have frequently encountered in the practice of OR. We want to solve large instances of a hard problem

$$\min_{x \in \mathcal{X}^{\mathrm{h}}(\Gamma^{\mathrm{h}})} f^{\mathrm{h}}(x; \Gamma^{\mathrm{h}}), \tag{h}$$

and we have an algorithm $\mathcal{A}^{\mathrm{h}}$ that can solve only moderate size instances of our problem. But our hard problem is a variant of an "easy" problem

$$\min_{x \in \mathcal{X}^{\mathrm{e}}(\Gamma^{\mathrm{e}})} f^{\mathrm{e}}(x; \Gamma^{\mathrm{e}}), \tag{e}$$

that is, a problem for which we know an efficient solution algorithm $\mathcal{A}^{\mathrm{e}}$ that can handle large instances. Let $\aleph^{\mathrm{h}}$ and $\aleph^{\mathrm{e}}$ be the set of instances of the hard and the easy problem, respectively.

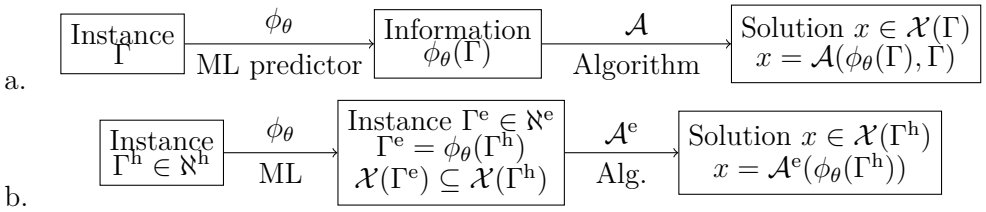

FIG. 1 – a. General scheme of ML algorithm for OR problems. b. the paradigm we propose

Our first contribution is a novel "ML for OR" paradigm that uses the moderately efficient algorithm $\mathcal{A}^{\mathrm{h}}$ to learn a practically efficient heuristic for hard problem of interest (h). It is illustrated on Figure 1.b and can be described as follows. First, we use an ML predictor $\phi_\theta : \aleph^{\mathrm{h}} \to \aleph^{\mathrm{e}}$ to turn our instance Γ^{h} of the hard problem into an instance Γ^{e} of the easy problem (e) such that the set of feasible solution $\mathcal{X}^{\mathrm{e}}(\Gamma^{\mathrm{e}})$ is contained in the set of feasible solution $\mathcal{X}^{\mathrm{h}}(\Gamma^{\mathrm{h}})$. Then, we use the efficient algorithm $\mathcal{A}^{\mathrm{e}}$ to find a solution x in $\mathcal{X}^{\mathrm{e}}(\Gamma^{\mathrm{e}})$, and return it as the solution of our instance Γ^{h} of the hard problem. The main question that must be answered to make this approach work is the following : *How to choose θ in such a way that $\mathcal{A}^{\mathrm{e}}$ applied on $\phi_\theta(\Gamma^{\mathrm{h}})$ provides a good solution of Γ_h ?*

Our second contribution is methodology to answer that question. Given a sample of moderate size instances of (h), we use algorithm $\mathcal{A}^{\mathrm{h}}$ to compute the solution of these instances. We then show that the problem of learning θ can then be seen as a structured learning problem [3]. Structured learning is a branch of machine learning that approximates functions $t = h(s)$ where t is in a combinatorially large space $\mathcal{T}(s)$ as follows : Given a new s, it solves an auxiliary inference problem

$$\min_{t \in \mathcal{T}(s)} g_\theta(s, t),$$

and returns an optimal t as the predicted value of $h(s)$. In our case, s is an instance Γ^{h} of the hard problem (h), $h(s)$ is an optimal solution of Γ^{h}, and our auxiliary inference problem is the easy problem (e) on instance $\phi_\theta(\Gamma^{\mathrm{h}})$. Having framed the choice of θ as a structured learning problem, we can use a probabilistic learning approach [3, Chapter 5] to learn θ.

Structured learning requires ad-hoc learning algorithms for each kind of auxiliary inference problem – of easy problem (e) in our application. We demonstrate the relevance of our approach using two of the most common OR problems as easy problems (e) : the usual shortest path problems and and the flow problem, both on acyclic digraphs. To the best of our knowledge, neither of these have ever been considered as auxiliary inference problems in structured learning. Our third contribution is a probabilistic learning approach to handle them. Our paradigm and algorithms therefore provide a direct method to approximate any path problem by a usual shortest path problem, and any path partition problem by a flow problem. We also provide matheuristics that exploit our approximations by usual shortest path problems as an approximate pricers in a column generation scheme, and can thus deal with any problem that can be solved by a column generation whose pricing subproblem is a path problem.

We demonstrate the efficiency of our approach with the stochastic vehicle scheduling problem (VSP) as hard problem (h). Given a set of tasks with fixed beginning and ending time, the VSP aims at finding the sequences of tasks operated by some vehicles so as to operate all these tasks at minimum cost. In its stochastic version, which has notably applications to airline operations [2], tasks may be delayed, vehicles propagate delay, and propagated delay cost is minimized. The deterministic VSP is easy to solve with a flow approach, while the stochastic VSP is (not so well) solved using column generation. The flow based and column generation heuristics we get using our paradigm are more than three order of magnitude faster than the algorithm $\mathcal{A}^{\mathrm{h}}$ we used to train them, and can handle instances with several thousand tasks when $\mathcal{A}^{\mathrm{h}}$ could only solve instances with at most 50 tasks.

Références

[1] Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization : a methodological tour d'horizon. *arXiv preprint arXiv :1811.06128*, 2018.

[2] Shan Lan, John-Paul Clarke, and Cynthia Barnhart. Planning for robust airline operations : Optimizing aircraft routings and flight departure times to minimize passenger disruptions. *Transportation Science*, 40(1) :15–28, 2006.

[3] Sebastian Nowozin, Christoph H Lampert, et al. Structured learning and prediction in computer vision. *Foundations and Trends® in Computer Graphics and Vision*, 6(3–4) :185–365, 2011.