Learning structured approximations of operations research problems.

Axel Parmentier

CERMICS, École des Ponts, Champs sur Marne, France
axel.parmentier@enpc.fr

Mots-clés : *Machine learning for combinatorial optimization, structured approximation, approximation algorithm, stochastic vehicle scheduling problem, single machine scheduling problem.*

Suppose that we are interested in solving a hard operations research problem

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

that is variant of a classic operations research problem

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

for which a practically efficient algorithm $\mathcal{A}^{\mathrm{e}}$ is available in the literature. The letters h and e stand for hard and easy. Γ^{h} and Γ^{e} respectively denote the instances considered. We indicate the instance Γ in the definition of the objective function and the set of admissible solution because several instances of the same problem will be considered simultaneously in the learning algorithm. The approach proposed is illustrated on Figure 1. A machine learning predictor $\varphi_{\boldsymbol{\theta}}$ is used to encode the hard problem instance Γ^{h} into an instance of Γ^{e}. Algorithm $\mathcal{A}^{\mathrm{e}}$ then finds an optimal solution x^{e}. Finally, a decoding algorithm ψ is used to rebuild a solution x^{h} for Γ^{h} from x^{e}. The encoder $\varphi_{\boldsymbol{\theta}}$ is a statistical predictor and its computation is very fast. The easy problem (e) is chosen because even large instances are tractable with a (potentially advanced) algorithm $\mathcal{A}^{\mathrm{e}}$ from the literature. For instance, $\mathcal{A}^{\mathrm{e}}$ can involve the resolution of a tractable MILP formulation with a solver. Finally, the decoder ψ is not mandatory and is typically a descent heuristic. Hence, the learned algorithm is sufficiently fast to be applied online (on a single instance). And it can be applied many times (~ 1000 times) to several instances during the CPU intensive but offline learning phase.

Previous contributions [2, 3] show the practical advantage of this approximation paradigm : appropriate $\varphi_{\boldsymbol{\theta}}$ and (e) retain most of the structure of (h) and make the hard problem instance tractable with $\mathcal{A}^{\mathrm{e}}$. And their numerical experiments show that the learned algorithm is practically efficient and that it meets the scaling challenge [1, Section 6.2]. But they underline two challenges that must be met to make the approach work. First, we must design $\varphi_{\boldsymbol{\theta}}$ in such a way that the solution returned has a small objective $f^{\mathrm{h}}(x^{\mathrm{h}},\Gamma^{\mathrm{h}})$. And second, we must formulate the learning problem and propose a learning algorithm. The initial contributions

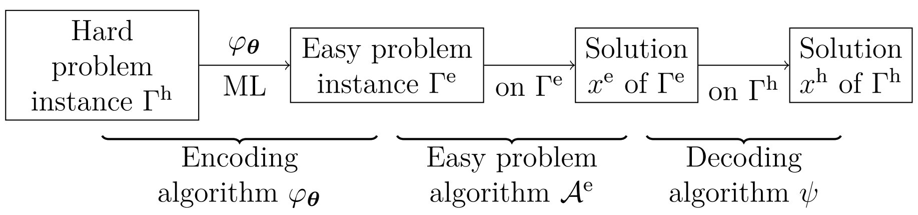
Hard problem instance Γ^{h}
$\varphi_{\boldsymbol{\theta}}$
ML
Easy problem instance Γ^{e}
on Γ^{e}
Solution x^{e} of Γ^{e}
on Γ^{h}
Solution x^{h} of Γ^{h}
Encoding algorithm $\varphi_{\boldsymbol{\theta}}$
Easy problem algorithm $\mathcal{A}^{\mathrm{e}}$
Decoding algorithm ψ

FIG. 1 – ML to approximate hard problems by well-solved ones

illustrate on specific applications how to build and learn approximations, but do not provide a generic method to address these challenges. In this work, we formalize the notion of structured approximation of one problem by another, propose generic methods to design and learn such approximations, provide theoretical guarantees on their performances, and show their practical performance through numerical experiments.

We make the following contributions :

1. We formalize the notion of *structure* of a problem in order to introduce the notion of *structured approximation* of an operations research problem (h) by another one (e). Practically, such an approximation of (h) is given by an easy problem (e) sharing the same structure, as well as an encoder $\varphi_{\boldsymbol{\theta}}$ and a decoder ψ that satisfy several properties.
2. We introduce a formulation of the learning problem and algorithms to solve it. We expect them to work on any structured approximation such that the dimension of $\boldsymbol{\theta}$ is moderate (≤ 100) and the solution pipeline can be applied several hundred to several thousand times during the learning phase. Contrarily to previous formulations of the learning problem which required a training set containing instances of (h) as well as an optimal (or at least a good) solution of each instance, this formulation requires only instances of (h) but no solution. This is a practical advantage because no algorithm for the hard problem is required.
3. We demonstrate the practical performance of our learning algorithm on the applications of the literature [2, 3] : Our learned algorithm (which has no access to optimal solutions of instances in the training set) matches the performance of those of the literature (which have access to optimal solutions), and are therefore state-of-the-art heuristics for the problems considered.

To the best of our knowledge, the literature on machine learning algorithms for combinatorial optimization problems has focused on the practical efficiency of these heuristics and no theoretical guarantees on the learning algorithms have been proposed [1, Section 6.2]. More precisely, convergence results for the estimator of $\boldsymbol{\theta}$ they use may exist in the statistical learning literature, but the consequences of these results in terms guarantees on the optimality gap of the solution returned by the learned algorithm have not been studied. Since we propose structured approximations, it is natural to wonder if and at which speed our learning algorithm converges towards the best approximation.

4. Leveraging tools from statistical learning theory, we prove the convergence of the learning algorithm toward the approximation with the best expected loss, and an upper bound on the convergence speed that does not depend on the diversity instances structures.
5. We also prove that, if $\varphi_{\boldsymbol{\theta}}$ is sufficiently regular and there exists a parameter $\tilde{\boldsymbol{\theta}}$ such that the easy problem objective approximate well the hard problem objective, then with high probability on the training sample, the learned algorithm is an approximation algorithm for the hard problem (h) whose approximation ratio improves with the size of the training set.

Références

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

[2] Axel Parmentier. Learning to Approximate Industrial Problems by Operations Research Classic Problems. *Operations Research*, April 2021.

[3] Axel Parmentier and Vincent T'Kindt. Learning to solve the single machine scheduling problem with release times and sum of completion times. *arXiv :2101.01082 [cs, math]*, January 2021.