Unlocking the power of modern ML for use in CSP

Chris Cameron[1], Jason Hartford[2], Taylor Lundy[1], Kevin Leyton-Brown[1]

[1]Department of Computer Science, University of British Columbia [2]Mila, Université de Montréal
{cchris13, tlundy, kevinlb}@cs.ubc.ca, jason.hartford@mila.quebec

1 Abstract

Modern deep-learning architectures have unlocked the capability of training end-to-end models for instance-specific predictions about CSP problems that take the raw representation of a problem instance as the model input. Our work leverages *exchangeable* deep networks to reason about CSP problems in an end-to-end fashion and ultimately to synthesize new algorithms. We will begin by describing work that achieved state-of-the-art performance at predicting satisfiability of computationally-challenging SAT problems (Cameron, Chen, et al., 2020). Next, we extended this architecture to develop Monte Carlo Forest Search (MCFS), a new algorithm for automatically synthesizing strong UNSAT solvers for given distributions of formulas (Cameron, J. Hartford, Lundy, Truong, et al., 2022). This work leverages ideas from the Monte Carlo Tree Search (MCTS) algorithm that led to breakthroughs in AlphaGo and is applicable for guiding tree search in any CSP domain. Finally, we will describe work in the predict-then-optimize setting (Cameron, J. Hartford, Lundy, and Leyton-Brown, 2022) where model predictions generate the objective function or constraints of an optimization problem, a setting which has attracted interest after the introduction of the *optimization-as-a-layer* architecture.

Keywords: Improvements to SAT Solvers; Machine learning theory and algorithms; RL; MCTS

2 Presentation

We will present: Predicting SAT(Cameron, Chen, et al., 2020), Predict-then-optimize (Cameron, J. Hartford, Lundy, and Leyton-Brown, 2022), MCFS (Cameron, J. Hartford, Lundy, Truong, et al., 2022)

Over the past two decades, machine learning has been shown to be very useful for making instance-specific predictions about properties of CSP problems (e.g., algorithm runtime prediction (Hutter et al., 2014), algorithm selection (Xu, Hutter, et al., 2008), and satisfiability prediction (Xu, Hoos, and Leyton-Brown, 2012)). A key drawback of this work is its reliance on hand-engineered features, requiring between linear and cubic time in the size of the input. It is difficult to assess whether less computationally expensive features would yield similar results, to determine whether important features are missing, and to translate a modeling approach to a new domain. Learning representations *end-to-end* from raw problem descriptions via neural networks is a promising approach for addressing these obstacles. Evans et al., 2018 and Selsam and Bjørner, 2019 were among the first to apply this approach; this work is very interesting from a machine learning perspective, but has tended to focus on problems that are trivial from a combinatorial optimization perspective (e.g., requiring less than 1 second to solve for modern SAT solvers).

We will present our work on the first investigation of end-to-end deep networks for predicting satisfiability on challenging distributions (hours to solve). We encode raw CNF SAT problems as permutation-invariant sparse matrices, where rows represent clauses, columns represent variables, and matrix entries represent whether or not a variable is negated. We used an *exchangeable* deep network architecture (J. S. Hartford et al., 2018) that can handle arbitrary-sized problems and capture a key invariance exhibited by SAT problems (and many other CSP domains): satisfiability status is unaffected by reordering constraints and variables within constraints. We evaluated our approach on uniform-random 3-SAT instances at the phase transition (50% satisfiable), primarily to facilitate comparison with past work. Despite the fact that our models did not have access to hand-engineered features and that they were only able to learn linear-time computable features, we achieved better performance than that of Xu, Hoos, and Leyton-Brown (2012). For example, on 600-variable problems (which typically take hours to solve), we increased accuracy by 3% up to 84%. Our work introduced the first example of state-of-the-art performance for the end-to-end modelling of the relationship between the solution to a combinatorial decision problem and its raw problem representation on a distribution that is challenging for modern solvers.

Next, we turned our attention to leveraging exchangeable deep networks to guide decision making in CSP solvers. It has long been tempting to use reinforcement learning to directly synthesize heuristic policies that are optimized for problem distributions of interest. The idea dates back to Lagoudakis and Littman (2001) who used TD-learning to train a policy to select between seven predefined branch heuristics based on simple hand-crafted features. More recently, others have leveraged modern architectures (Kurin et al., 2019; Tönshoff et al., 2022; Yolcu and Póczos, 2019), although none of these approaches has (yet) had much impact in practice.

We asked whether the revolutionary breakthroughs in RL exemplified by AlphaGo (Silver et al., 2017) constitute a game changer. The key idea behind AlphaGo was combining Monte Carlo Tree Search (MCTS) rollouts with a neural network-based policy to find increasingly strong paths through the game tree. MCTS is a promising approach for finding strong paths through any large combinatorial space. However, proving that no solution exists in tree-search CSP solvers requires enumerating multiple paths to build out a *proof tree* that demonstrates that all possible variable assignments lead to a conflict. The order in which variables are assigned—the branching policy—has a dramatic effect on the size of the tree and the corresponding time to solve the problem, so algorithm designers seek branching policies that lead to small trees.

We will present an extension to MCTS for finding small search trees in a forest of alternatives, which we call Monte Carlo Forest Search (MCFS). The MCFS algorithm uses *Knuth samples* (Knuth, 1975) to obtain path-based and unbiased Monte Carlo approximations of tree sizes, allowing us to leverage MCTS to search over branching policies that produce trees. Since our deep neural network policies are far more expensive to evaluate than standard heuristics, we bound policy evaluations by querying a known strong subsolver below a certain depth in the proof tree. This focuses the search on early decisions where the policy has the highest potential to reduce tree size relative to inference cost. We evaluated our approach by integrating MCFS into the DPLL algorithm via the MiniSAT (Eén and Sörensson, 2003) framework and learning a branching heuristic to efficiently find small proof trees of unsatisfiability. We matched or improved over the performance of a strong baseline on two prominent SAT Competition distributions, highlighted by improving running time on the `sgen` distribution by 9% over the `kcnfs07` solver.

Finally, we will present our work on the predict-then-optimize setting, where real-world optimization problems are generated from predictions from historical data (e.g., an optimizer that aims to recommend fast routes relies upon travel-time predictions). Typically, learning the prediction model used to generate the optimization problem and solving that problem are performed in two separate stages. Recent work has showed how such prediction models can be learned end-to-end by differentiating through the optimization task, launching a new research area bridging ML and CSP (Amos and Kolter, 2017). End-to-end methods often yield empirical improvements, which are typically attributed to making better error tradeoffs than the standard loss function used in a two-stage solution. We refine this explanation and more precisely characterize when end-to-end can improve performance. When prediction targets are stochastic, a two-stage solution must make an a priori choice about which statistics of the target distribution to model—we consider expectations over prediction targets—while an end-to-end solution can make this choice adaptively. Even when there is no limit on model capacity or training set size, we show how the two-stage approach can fail catastrophically in the common case where the prediction stage models expectations over prediction targets.

We then consider a novel setting in which predict-then-optimize is a common paradigm and end-to-end learning is particularly relevant: where multiple prediction targets are combined to obtain each of the objective function's coefficients. We show that this setting can give rise to potentially unbounded performance gaps between the end-to-end and the two-stage approaches. These results highlight two main features of an optimization problem that together can lead to suboptimal two-stage solutions, and that practitioners should look out for in practice. The first is that multiple targets are learned independently, without accounting for correlations. This is likely to happen when the targets are conceptually different and tend to be learned separately, but may be correlated via latent factors that affect them both. The second is that targets are combined nonlinearly to compute coefficients of the objective function for downstream optimization. This is quite common in practice, notably where the cost of some decision is the product of two inputs (e.g., total cost = cost/unit · #units). Although the reader might feel that two-stage learning is obviously problematic in such domains, we illustrate that two-stage learning has been widely used in real-world optimization problems likely to satisfy these conditions.

References

Amos, B. and J. Z. Kolter (2017). “OptNet: Differentiable optimization as a layer in neural networks”. *Proceedings of the 34th International Conference on Machine Learning*. Vol. 70. Proceedings of Machine Learning Research. PMLR, pp. 136–145.

Cameron, C., R. Chen, et al. (2020). “Predicting propositional satisfiability via end-to-end learning”. *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 04, pp. 3324–3331.

Cameron, C., J. Hartford, T. Lundy, and K. Leyton-Brown (2022). “The perils of learning before optimizing”. *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 36. 4, pp. 3708–3715.

Cameron, C., J. Hartford, T. Lundy, T. Truong, et al. (2022). “Monte Carlo Forest Search: UNSAT Solver Synthesis via Reinforcement learning”. *arXiv preprint (to appear November 21st)*.

Eén, N. and N. Sörensson (2003). “An extensible SAT-solver”. *Proceedings of the 6th International Conference on Theory and Applications of Satisfiability Testing*. SAT ’03, pp. 502–518.

Evans, R. et al. (2018). “Can Neural Networks Understand Logical Entailment?” *arXiv preprint*. arXiv: 1802.08535.

Hartford, J. S. et al. (2018). “Deep Models of Interactions Across Sets”. *Proceedings of the 35th International Conference on Machine Learning*. Vol. 80. ICML ’18, pp. 1914–1923.

Hutter, F. et al. (2014). “Algorithm runtime prediction: methods & evaluation”. *Artificial Intelligence* 206, pp. 79–111.

Knuth, D. E. (1975). “Estimating the efficiency of backtrack programs”. *Mathematics of Computation* 29.129, pp. 122–136.

Kurin, V. et al. (2019). “Can Q-learning with graph networks learn a generalizable branching heuristic for a SAT solver?” *Proceedings of the 34th International Conference on Neural Information Processing Systems*. NeurIPS ’20, pp. 9608–9621.

Lagoudakis, M. G. and M. L. Littman (2001). “Learning to select branching rules in the DPLL procedure for satisfiability”. *Electronic Notes in Discrete Mathematics* 9, pp. 344–359.

Selsam, D. and N. Bjørner (2019). “Guiding high-performance SAT solvers with unsat-core predictions”. *Proceedings of the 22nd International Conference on Theory and Applications of Satisfiability Testing*. SAT ’19, pp. 336–353.

Silver, D. et al. (2017). “Mastering the game of Go without human knowledge”. *Nature* 550, pp. 354–359.

Tönshoff, J. et al. (2022). “One Model, Any CSP: Graph Neural Networks as Fast Global Search Heuristics for Constraint Satisfaction”. *arXiv preprint* arXiv:2208.10227, pp. 1–23.

Xu, L., H. H. Hoos, and K. Leyton-Brown (2012). “Predicting Satisfiability at the Phase Transition”. *Proceedings of the 26th AAAI Conference on Artificial Intelligence*. AAAI ’12. AAAI Press, pp. 584–590.

Xu, L., F. Hutter, et al. (2008). “SATzilla: Portfolio-based Algorithm Selection for SAT”. *Journal of Artificial Intelligence Research* 32, pp. 565–606.

Yolcu, E. and B. Póczos (2019). “Learning local search heuristics for boolean satisfiability”. *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. NeurIPS ’19, pp. 7992–8003.