Towards Sequence-to-Sequence Reinforcement Learning for Constraint Solving with Constraint-Based Local Search

Helge Spieker
Simula Research Laboratory
P.O. Box 134
1325 Lysaker, Norway
helge@simula.no

Abstract

This paper proposes a framework for solving constraint problems with reinforcement learning (RL) and sequence-to-sequence recurrent neural networks. We approach constraint solving as a declarative machine learning problem, where for a variable-length input sequence a variable-length output sequence has to be predicted. Using randomly generated instances and the number of constraint violations as a reward function, a problem-specific RL agent is trained to solve the problem. The predicted solution candidate of the RL agent is verified and repaired by CBLS to ensure solutions, that satisfy the constraint model. We introduce the framework and its components and discuss early results and future applications.

Introduction

We approach constraint solving with reinforcement learning (RL) and sequence-to-sequence neural networks. The goal of this approach is to build problem-specific machine learning models, that are able to support solving problems as they occur in constraint programming (CP). Dedicated constraint solvers are highly optimized, still, searching for good or even optimal solutions often is time-consuming due to large search spaces that have to be traversed. Here, an argument for machine learning are the shorter inference times compared to the solving times of constraint solvers.

Data-driven constraint solving is difficult to approach by supervised learning for two main reasons. First, it is computationally expensive to generate a sufficiently large training corpus for constraint problems, as it requires to solve each problem instance to generate the optimal labels. Second, supervised training potentially leads the model to closely imitate the solutions found by the solver used to build the training set (Bello et al. 2017). Especially if an instance has multiple optima, training only on one solution limits generalization to an individual problem-solving strategy.

These limitations are overcome by RL, where only a scalar reward is necessary as feedback for training. Computing this reward, which formed by the number of constraint violations and the objective value of the solution candidate, is easier to compute than an optimal or high-quality solution. Learning from rewards additionally allows the agent to find individual solution strategies and to decide which parts of the search space to explore, instead of adapting to the strategy behind the solution labels, as in supervised learning.

In constraint modeling, a domain expert explicitly models a problem in terms of constraints, variables, their domains, and, in case of optimization, an objective function. Such a constraint model can be described as a function, that takes a set of instance parameters as an input, and, with the help of a constraint solver, solves the instance for which then the solution is returned as a set of output variables, i.e. the assigned variables. Following this definition of a constraint problem, the process of constraint solving is framed as a structured sequence-to-sequence (S2S) problem with bandit feedback, that is, a reward is only received after the full solution to the problem instance is predicted.

Method

It might not be practical to deploy only a RL-trained model, as it is undesirable to receive unfeasible solutions for practical use cases, and feasibility of the solution can not be guaranteed. However, we propose to combine the RL agent with a constraint-based local search (CBLS) solver (Hentenryck and Michel 2009). In that way, every predicted solution can be checked for feasibility and, if necessary, be repaired via local search techniques while obeying the constraints of the problem. In case of an unsatisfiable instance, this is detected by the CBLS component, too. Also, it would be possible to give the solver additional time to find an improved solution starting from the predicted solution.

This hybrid framework (as shown in Figure 1) combines the advantages of RL-based constraint solving while guaranteeing feasible solutions through the CBLS solver, which can allow to handle larger problem instances and reduce total solving time than each technique alone. In this paper, we describe the design of the RL agent within the framework.

Model The COP to be solved by the agent is described as a S2S problem with variable lengths inputs and outputs. Each individual problem instance is described by instance parameters, which form the input to the agent. The solution is again a sequence, containing the assigned values for each variable. Our method is based on advantage actor-critic RL (A2C) (Mnih et al. 2016) for S2S. The architecture and algorithm have been previously introduced for machine translation with human feedback (Nguyen, Daumé III, and Boyd-

Copyright © 2019, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

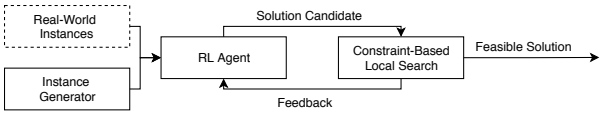

Figure 1: Hybrid constraint solving loop: A solution candidate is predicted by the RL agent and CBLS both ensures a feasible solution and provides a reward as feedback.

Graber 2017), which is a different domain with similar technical characteristics. We follow closely the previously proposed approach and extend it by using domain knowledge from constraint solving to improve the prediction process.

Representation Although the values in constraint problems are integers, they are embedded into dense vectors for input and output. Choosing an embedding is due to the reason, that the integers are often used as categorical data and not necessarily in a nominal way. In this case, using an embedding allows more representational power, and in nominal relations it is still represented by the learned weights.

Due to the structure of the constraint problem, each variable has a fixed position and also the total length of the output sequence is known beforehand, which allows to simplify prediction. During preprocessing, the COP is simplified, such that the domains are tightened. While sampling the model output, we exploit this information and allow only values from the tightened domain instead of the original, larger domain. Thereby, we reduce the instance action space.

Reward The reward is formed by the number of violated constraints in the solution candidate and its objective value. If the solution candidate is a feasible solution and satisfies all constraints, the reward is equal to the objective value of the solution (in case of maximization), otherwise the number of constraint violations is returned as a negative reward.

This reward function puts emphasis first on the feasibility of the solution candidate by giving low rewards for unfeasible candidates without considering the objective. After a feasible solution strategy has been found, the emphasis switches towards maximizing the problem objective. The reward for an instance can be calculated either by a problem-specific function or, by using the abstract constraint model, as part of the CBLS solver used within the framework (see Figure 1). Having a generic, domain-agnostic reward function allows applying the method to different problems.

Experiments

We show first experiments of the presented method on one constraint optimization problem, namely Maximum Increasing with Limits. For the full paper, we include experiments on harder constraint optimization problems, such as jobshop scheduling or car sequencing (Perron and Shaw 2004).

Maximum Increasing with Limits (MIWL) is a constraint maximization problem with the goal to find the maximum sum of a list of positive integers x, excluding 0, with length n. The maximum size of each integer x_i ($i \in \{1..n\}$) is limited by an instance parameter $limits_i$ ($x_i < limits_i$) and the total sequence x has to be strictly increasing ($x_i < x_{i+1}$). The instance is then an integer n and a list of $limits$, the solution is the list of integers x.

We train the agent on 150,000 random instances with up to 250 elements. During the initial 10 supervised epochs, the agent predicts 32% feasible solution candidates. After one epoch of RL, this value increases to 88% and after some further epochs to 92%. Both the number of feasible solutions and the total reward converges after 20 epochs.

We also evaluate the minimum reward during validation to gain insights on the worst performance of the RL agent. At the end of the pre-training phase, the minimal reward is -88, i.e. the worst solution candidate has 88 violations. After convergence, the worst solution candidate has 10 violations, i.e. a reward of -10, which shows that the predicted solution is close to be feasible and is likely to be repaired in few additional local search iterations afterwards.

The initial results on MIWL show the ability to predict solution candidates and underline the motivation to use RL for training. Besides the reduced effort in training data generation, without having to solve the training instances to an optimal solution, giving a scalar reward instead of a feasible solution shows better generalization.

Conclusion & Outlook

We present a hybrid framework for constraint solving, consisting of a neural network and a CBLS solver. Using a S2S architecture allows solving problem instances with variable-length inputs and outputs with different lengths, instead of training a model for one specific instance size. The agent is trained with RL under bandit feedback. The generic reward function is implemented in terms of constraint violations, which is a metric that is generally accessible and problem-independent. Our method is thereby applicable to a wide range of constraint problems. Initial experiments show success in learning constraint solving from scalar rewards for an exemplary problem.

Combining RL and CBLS allows to exploit historical information and learn from the constraint model. Providing an RL agent shifts computational load from solving time to training time. The two main components of the framework, the RL agent and the CBLS solver, are modular and can be individually tuned and adapted to problem-specific needs. Next steps are to consider more complex constraint optimization problems and to evaluate different sequence-to-sequence network architectures.

References

Bello, I.; Pham, H.; Le, Q. V.; Norouzi, M.; and Bengio, S. 2017. Neural Combinatorial Optimization. In *ICLR*.

Hentenryck, P. V., and Michel, L. 2009. *Constraint-Based Local Search*. The MIT Press.

Mnih, V.; Badia, A. P. A. P.; Mirza, M.; Graves, A.; Lillicrap, T. P.; Harley, T.; Silver, D.; and Kavukcuoglu, K. 2016. Asynchronous Methods for Deep Reinforcement Learning. In *ICML*, 1928–1937.

Nguyen, K.; Daumé III, H.; and Boyd-Graber, J. 2017. Reinforcement Learning for Bandit Neural Machine Translation with Simulated Human Feedback. In *EMNLP*, 1464–1474.

Perron, L., and Shaw, P. 2004. Combining Forces to Solve the Car Sequencing Problem. In *CPAIOR*, 225–239.