WASP SE Course Assignment - Andy Oertel

Introduction

My research is on certifying algorithms for combinatorial optimization. At this point I should explain briefly what this means. In combinatorial optimization we study how to solve optimization problems with discrete variables efficiently (e.g., constraint programming, mixed integer programming, ...). A major problem in combinatorial optimization is that modern solvers for these problems are very complex and contain a lot of systems that interact in unforeseen ways. Hence, designing good test cases is basically impossible, as it is difficult to design test that trigger these interactions and even then the solver could come up with the right solution even though this was due to a bug. Formal verification is also impossible for these solvers due to their complexity.

The idea of certifying algorithms is that the algorithm should not just provide a result, but also a formal proof/certificate showing the correctness of the result. Hence, if we can successfully check the certificate for the result of a certain problem, then we have a formal guarantee that the result is correct and that the result can be derived by the reasoning in the certificate. So if the reasoning in the certificate is close the reasoning done by the solver, we can certify the correctness of the reasoning by the solver for the problem. The big advantage compared to formal verification is that this approach actually scales to the complexity of modern solvers (demonstrated in multiple papers).

The certification enables various new approaches for testing such solvers. E.g., bugs can be detected even if the result is correct, but the reasoning to obtain this result is wrong, which could lead to an incorrect result for a different test case.

A limitation that I should mention is that this approach is only viable if the problem can be formally specified, which is easy for combinatorial optimization.

Robert's Lecture and My Research

Don't Repeat Yourself

One software engineering principle that was briefly mentioned in the lecture was *Don't Repeat Yourself*, which means that replication should be minimized. I would like to focus the discussion on certifying algorithms. Many groups have developed very domain specific certificate formats and checkers for their specific combinatorial optimization problem. However, these formats and checkers have a lot of things in common or are basically equivalent. So the same solutions have been implemented over and over again just with slight changes. For the first time however, our tool takes a more abstract view and provides a unified certification format and checker. Which also keeps the development effort for the formal verification of the checker minimal, as the core

checker stays the same for all kinds of problem domains.

Fuzzing

Using certifying algorithms fuzzing can become an extremely strong testing technique, as test cases can be generated completely at random without knowing the result that the software should return a priori. Using this random test case as input for a certifying algorithm gives us a result and a certificate. Hence, we can now check the certificate with a trusted (formally verified) tool to make sure that the software worked correctly on this instance. So even though we did not know the expected result when generating the test case we know that the result returned by the software is correct after we checked the certificate. Furthermore, due to the certificate we can not only check the correctness of the result, but also the correctness of the reasoning used by the solver.

Guest Lecture and My Research

Many People from Different Fields with Different Requirement

I mainly develop the checker for the certificates as part of my research project. Slowly more people get to know about our tool and our approach and get interesting in using it. Hence, there is a growing user base that comes up with new requirements and feature requests for the tool all the time. Unfortunately, these requirements do not overlap that often, and it is not that clear what the actual new requirements should look like, as the user just says that they want to do something without knowing what is needed to do that.

To bring some Behavioural Software Engineering into the discussion, it can be very hard to prioritize which feature is the most needed. While some users are exaggerating the need for their requirements, others are downplaying the importance of their issues, even if their requirement would bring more benefit to the tool. Also, there have been cases where my advisor owed a favour to someone and then their issue got prioritized even though this was not one of the most important issues we had at the time.

AI to Generate Test Instances in Structured Way

Relating my research to the lecture by Parthasarathy Dhasarathy and ML4SE. As mentioned before, certifying algorithms can improve fuzzing. As it does not matter in which way the test case was generated, a machine learning system could be used to first explore randomly to generate good test cases. Then over time the system could learn to generate meaningful instances, so that bugs are triggered more likely or generate more instances that cause the same to aid debugging efforts.

The certificate could also be given to the system, which could be used by the ML system to extract information about bugs or the running of the solver. This insight could then be used to improve the ability to find new bugs or detect pattern when solvers are reasoning in the same way.

CAIN Papers

Paper 1: "Exploring Hyperparameter Usage and Tuning in Machine Learning Research"[1]

This paper studies how hyperparameters and hyperparameter tuning is reported on in ML research papers. Tuning hyperparameters can turn a bad ML method into a good one and reporting on hyperparameters is important to reproduce research results. The authors look at ML papers and their code published between 2015 and 2022 and analyse them using static analysis tool to understand the usage and reporting of hyperparameters used in their research. The main finding is that in many cases no or only a few hyperparameters are tuned, while many other hyperparameters are not tuned. Furthermore, many papers do not mention any hyperparameter tuning, or they only do simple tuning techniques or even manual tuning. This leaves the impression that despite the relevance of hyperparameters for the performance of ML methods, ML papers and researchers do not consider them. Finally, the paper discusses explanation why hyperparameter tuning is underexplored and underreported in ML research.

While there are no hyperparameters in combinatorial optimization solvers, we have a lot of parameters that could be tuned. Even though many papers reported big improvements in the performance of solvers, many papers and researchers ignore this fact and just use some guessed default values or manual tuning of parameters. Only recently software frameworks for parameter tuning of combinatorial optimization solvers have been investigated and are still mainly unused. For combinatorial optimization solver it is often the case that extensive tuning on one class of instances can reduce their performance in general. Hence, tuning is only really effective if it can be assured that the problems we want to solve exhibit a similar enough structure. Then the parameters can be tuned to be effective for this problem class, and we have to know a priori which problem class we want to solve.

A larger AI-intensive software project in which my project and this paper would fit is a combinatorial optimization solver, as they are basically classical AI and state-of-the-art for many AI applications. However, many groups also integrate ML methods into their solvers. As mentioned before, my research can be applied to the development of these solvers to improve the correctness and understand the reasoning used by the solvers. The insight from the paper is that the development process should use parameter tuning for the classical parameters of the software and the hyperparameters of the ML part of the solver. Tuning the classical parameters and hyperparameters could greatly improve the performance of the solver and thus improve the problems solved by the tool. Furthermore, ML could be used to detect the problem class the problem we want to solve is from and then adjust the classical parameters to most efficiently solve the problem. To make full use of automated methods and predictive models, we could even use an ML model to predict classical parameters for a problem from a problem class that has not been solved before.

My research and parameter tuning could even be combined. The insights obtained from the

certificate on how the algorithm reasons about the problem could be used to more deeply understand how changing one parameter impacts the reasoning performed by the solver. This could then be used to more effectively search through the space of parameters or to fit the parameters exactly to the specific problem we want to solve.

Paper 2: "Novel Contract-based Runtime Explainability Framework for End-to-End Ensemble Machine Learning Serving"[2]

The paper studies so-called End-to-End Ensemble Machine Learning Serving (EEMLS), which are machine learning models in the cloud that can be accessed and used by customers as a service. The main motivation is that similar to traditional service level agreements (SLAs) for clouds, these contracts should be extended to cover ML specific properties. Additionally, to classical metrics like response time or throughput of a service, also prediction accuracy and confidence should be considered as part of an SLA contract (e.g., guarantee that prediction is at least 85% accurate). The question that arises from this idea is how to monitor the EEMLS service quality and how/when to make adjustments to improve the quality to the agreed standard. To enable the monitoring of ML models, the authors developed a software framework to instrument ML models. This framework collects data from each ML model in the ensemble through the monitoring and produces an explainability report, which contains the inference graph representing the aggregation of the result from the different ML models. Each ML model contains model specific properties and metadata.

In my paper I also instrument existing software to understand the inner working and quality (correctness) of the execution. At the moment, certificates are only checked after execution, but it would also be possible to do on-the-fly proof checking and analysis during the execution. This could make integration of classical AI methods into the framework possible and interesting. Hence, the explainability report could include solvers as inference steps and statistics about the reasoning they did. Then based on the report, the solvers could be tuned to improve performance on future instances, as it seems that the set of instances sent by the customer has changed over time. Hence, this would enable adaptability of the system to new use cases to guarantee a minimum service quality, which is probably measured in more classical means (e.g., response time) for these classical solvers.

A larger AI-intensive software project that could make use of my research and this paper could be for applications that can benefit of an ensemble of ML models and exact reasoning. E.g., consider a software for medical image analysis that uses ML models for recognizing and identifying features in the medical image and then a solver can reason about the features to come to a medical diagnosis. Based on the certificate, we can not only trust that the final reasoning is correct, but we can also analyse the certificate to find out which features are the most relevant to come to this diagnosis. Then we can go even further back and look at which ML models came up with these features and how confident they were. This also enables humans to look at the reasoning and the features and validate the judgement done by the software.

As discussed above, the report could also be used as feedback to improve the performance of future reasoning performed by the classical optimization solver. The goal of this could be to

improve response times, shorter explanations that are easier to understand by humans, or improve the accuracy of the diagnosis. For instance, the last item could be achieved by giving feedback to the ML models to improve their accuracy or use/prioritize different ML models that are more useful to reach a correct diagnosis.

References

1. Simon, Sebastian, et al. "Exploring Hyperparameter Usage and Tuning in Machine Learning Research." 2023 IEEE/ACM 2nd International Conference on AI Engineering–Software Engineering for AI (CAIN). IEEE, 2023. ↩

2. Nguyen, Minh-Tri, Hong-Linh Truong, and Tram Truong-Huu. "Novel Contract-based Runtime Explainability Framework for End-to-End Ensemble Machine Learning Serving." Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering-Software Engineering for AI (CAIN). 2024. ↩