Efficient explainability of schedulability analysis

Sanjoy Baruah * Pontus Ekberg (Speaker) †

Even after a successful analysis establishing a system's schedulability (e.g., that all deadlines are always met in a safety-critical real-time system), one may still have to prove this fact to a third party, such as a Certification Authority (CA). The third party may mistrust the veracity of the analysis and may be unable or unwilling to carry out the costly analysis on their own. We would like to be able to convince them instead via a (relatively) simple and trustworthy proof. Ideally, such proofs are formal and machine-checkable, such as the formally verifiable response-time certificates produced by the POET tool [6], which use the PROSA framework [3] on top of Coq/Rocq.

One of the fundamental challenges to overcome in such a setup is that not all computational problems are amenable to efficiently verifiable proofs. Indeed, if we were to define "efficient" as polynomial time, and wanted proofs to be static certificates that can be handed over to a third party for separate verification without further interaction, then the computational problems amenable to this would be exactly those in complexity class NP. Unfortunately, most interesting schedulability problems for real-time systems are not in NP. It is, of course, possible to create verifiable proofs also for problems outside NP, but such proofs would not be both polynomial in size and checkable in polynomial time. In the SAT community, it is common for SAT solvers to be able to produce verifiable proofs of *unsatisfiability*. But since UNSAT is (probably) not in NP, such proofs cannot be small in general, and indeed it has been reported that solvers have created UNSAT proofs greater than 100 GB in size even for benchmark problems used in SAT solver competitions[1]. In an interesting use of such UNSAT certificates to settle a long-standing number-theoretic problem [5], the produced proof was reported to be 2 PB in size and to take over 15 CPU-years to verify. Though seemingly much less used than in SAT solving, similar certificates also exist for mixed-integer linear programming: VIPR [4] is a certificate format that supports verifiable proofs of MIP infeasibility and optimality.

In the work outlined here, we try to sidestep inherent scalability challenges in verifying solutions (and also aim to address problems beyond coNP where UNSAT and MIP infeasibility is found) by outlining principles to carve out efficiently verifiable subsets of a problem (outside NP) that we are interested in solving. In essence

*baruah@wustl.edu. Washington University in St. Louis, USA
†pontus.ekberg@it.uu.se. Uppsala University, Sweden
[1]See https://satcompetition.github.io/2023/certificates.html

this is done by identifying special cases that are in NP or are otherwise amenable to some other type of efficient verification. Carving out an NP subproblem from a larger non-NP problem is not much different from carving out a polynomial time subproblem from a larger non-polynomial time problem, a very common practice in most fields of algorithms. But, at least from our experience in real-time scheduling, identifying NP subproblems is a much more unusual take on the same basic principle and may require thinking about solutions slightly differently. Indeed, it may well happen that producing certificates to an NP subproblem is significantly more computationally demanding than just solving the original problem, but the benefit we are looking for is the easily verifiable certificate and not reducing the initial solution time. In [2] we present a small proof of principle by identifying some simple NP subproblems of the coNP-complete EDF-schedulability problem.

Further, we lift the definition of efficient approximation schemes from solving to verifying problems. As an example, in [2] we extended a well-known FPTAS for EDF-schedulability [1] to take also a certificate as input, which (disregarding the effort to find the certificate) can make it more efficient, exponentially so for some instances. We dubbed the resulting type of approximation scheme FPTVAS, for fully polynomial-time verification approximation scheme and defined it as follows.[2]

Definition 1 (FPTVAS) *A fully polynomial-time* ***verification*** *approximation scheme (FPTVAS) for a schedulability analysis problem is an algorithm that, given as input an instance* Γ*, a parameter* $\delta > 0$*,* ***and a certificate****, returns "unschedulable" if* Γ *is unschedulable on a speed-1 processor, and returns "schedulable" if* Γ *is schedulable on a speed-*$(1/(1+\delta))$ *processor. Its running time, and the certificate size, is bounded by a polynomial in the two parameters* $|\Gamma|$ *and* $\frac{1}{\delta}$*.*

FPTVASs can not only have improved efficacy compared to FPTASs, but there also exist problems that have an FPTVAS that provably have no FPTAS (assuming $\mathsf{P} \neq \mathsf{NP}$). An example of this is the partitioned multiprocessor EDF-schedulability problem for which the FPTVAS mentioned above is easily adapted, but for which no FPTAS exists as it trivially generalizes bin packing.

Finally, recognizing that pseudo-polynomial time (and not polynomial time) is the typical gold standard of efficiency in real-time scheduling, we observe that it makes sense to aim also for pseudo-polynomial time verification. We are not aware of any previous work directly focusing on this concept, but a *nondeterministic pseudo-polynomial time* (pseudoNP) class can readily be defined. While a very straightforward concept, it seems meaningful as pseudoNP contains real-world problems that are neither in NP nor can be solved in (deterministic) pseudo-polynomial time. An example of this is again the (bounded-utilization) partitioned multiprocessor EDF-schedulability problem.

[2] We recognize that Definition 1 may seem a bit clunky to the approximation algorithms community. It mirrors the way FPTASs are usually thought of in real-time scheduling theory: as δ-gap promise problems that are always safe (no false positives) and in which processor speed is the augmented resource. This definition can certainly be generalized to fit other settings.

The astute reader will have noticed that we have failed to mention interactive proofs, perhaps the most elegant computer science concept targeting verification by an agent with bounded resources. It may well turn out that this is the best approach also in our imagined use cases, but as it stands, it is precisely the interaction that we would like to avoid by working on simpler first principles.

References

[1] K. Albers and F. Slomka. An event stream driven approximation for the analysis of real-time systems. In *Proceedings of the 16th Euromicro Conference on Real-Time Systems (ECRTS)*, pages 187–195, June 2004.

[2] Sanjoy Baruah and Pontus Ekberg. Towards efficient explainability of schedulability properties in real-time systems. In *Proceedings of the 35th Euromicro Conference on Real-Time Systems (ECRTS)*, pages 2:1–2:20, 2023.

[3] Felipe Cerqueira, Felix Stutz, and Björn B. Brandenburg. PROSA: A case for readable mechanized schedulability analysis. In *Proceedings of the 28th Euromicro Conference on Real-Time Systems (ECRTS)*, pages 273–284, 2016.

[4] Kevin K. H. Cheung, Ambros Gleixner, and Daniel E. Steffy. Verifying integer programming results. In *Integer Programming and Combinatorial Optimization*, pages 148–160, Cham, 2017. Springer International Publishing.

[5] Marijn J. H. Heule. Schur number five. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, AAAI'18. AAAI Press, 2018.

[6] Marco Maida, Sergey Bozhko, and Björn B. Brandenburg. Foundational Response-Time Analysis as Explainable Evidence of Timeliness. In *Proceedings of the 34th Euromicro Conference on Real-Time Systems (ECRTS)*, pages 19:1–19:25, 2022.