What is the Best Algorithm for Probabilistic Model Checking?

Tim Quatmann

RWTH Aachen University
tim.quatmann@cs.rwth-aachen.de

Markov chains and Markov decision processes are popular formalisms for modeling systems with probabilistic behavior. Their analysis commonly relies on computing reachability probabilities and expected accumulated rewards. Computing such measures *reliably* is a core functionality of probabilistic model checkers. Implementations usually are based on *value iteration (VI)*. In recent years, several variants of VI have emerged that—in contrast to the classical approach—produce *sound* approximations. In terms of runtime, the two most promising variants are *sound value iteration (SVI)* and *optimistic value iteration (OVI)*. Both approaches have been compared empirically by Hartmanns and Kaminski (2020), where an implementation in the model checker mcsta was exercised on a broad selection of benchmarks. The digest of the evaluation is that OVI is faster than SVI for the majority of instances. We replicate these research results using an implementation of both SVI and OVI in the model checker Storm.

Value Iteration The original VI algorithm dates back to the 1950s [1] and—besides probabilistic model checking—also finds applications in artificial intelligence and operations research. Roughly, VI approaches a fixpoint of the so-called Bellman operator $\Phi\colon \mathbb{R}^n \to \mathbb{R}^n$ by computing $\boldsymbol{x}_i \coloneqq \Phi(\boldsymbol{x}_{i-1})$ for increasing $i > 0$ and starting values $\boldsymbol{x}_0$. The algorithm terminates, when $\|\boldsymbol{x}_i - \boldsymbol{x}_{i-1}\|_\infty$ falls below a certain threshold. As argued in [2], this termination criterion, however, does not provide guarantees on the accuracy of the obtained solution. In fact, there exists adversarial models where VI—as implemented in model checkers such as mcsta, PRISM, and Storm—yields vastly incorrect results.

As a remedy, [2] proposes *interval iteration (II)* that synchronously runs two instances of VI with different starting values to approach the fixpoint of the Bellman operator from below *and* from above. SVI [4] also runs two VI instances synchronously: one instance computes $\boldsymbol{x}_i \coloneqq \Phi(\boldsymbol{x}_{i-1})$ for increasing $i > 0$ (as in VI) whereas the other instance provides bounds on the difference between $\boldsymbol{x}_i$ and the desired fixpoint. In contrast, OVI [3] uses classical VI to approach the fixpoint from below and a heuristic to "guess" candidates for upper bounds based on the current lower bound. Those candidates are verified using a separate instance of VI and a principle called Park induction.

Comparing SVI and OVI Empirical evaluations in [4,3] indicate that SVI and OVI outperform II in terms of runtime. Furthermore, [3] shows that—using their implementations in mcsta—OVI is faster than SVI on the majority of benchmarks from the Quantitative Verification Benchmark Set (QVBS). We replicate the experiments of [3] using implementations of OVI and SVI in the model checker Storm. There are a few differences compared to the implementations from [3]:

- Storm aggressively preprocesses the model using graph algorithms to , e.g., filter out states with reachability probability 1 or to collapse so-called end components. mcsta only performs these steps when required for correctness.
- Storm analyses the strongly connected components (SCC) of the model sequentially—thus invoking VI, OVI, or SVI multiple times—once per SCC. mcsta applies these algorithms directly for the entire model.
- Storm is written in C++ whereas mcsta is written in C#.

Replication Study We evaluated the Storm implementations on an Intel Xeon Platinum 8160 CPU using 32 GB RAM. We considered all benchmark instances from the QVBS on which Storm invokes SVI resp. OVI. This yields a total of 374 model checking queries—including the 79 instances from [3]. For 285 instances, both OVI and SVI terminate within 2 seconds. 18 instances could not be solved within 1800 seconds. Fig. 1 summarises the remaining results. Each point (x, y) in the plot refers to one benchmark instances and indicates that the OVI-based computations took x seconds whereas the SVI-based computations took y seconds. Those runtimes do *not* include model state space construction—which is independent of either SVI or OVI.

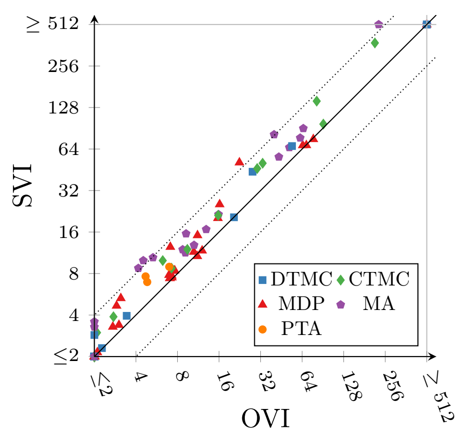

Fig. 1. Comparison of SVI and OVI

We observe that OVI clearly takes the lead over SVI for the vast majority of benchmark instances—confirming observations of [3]. Thus, OVI can be considered the preferred solution method when computing reachability probabilities and expected accumulated rewards with either mcsta or Storm.

References

1. Bellman, R.: A Markovian decision process. J. Math. and Mech. **6**, 679–684 (1957)
2. Haddad, S., Monmege, B.: Interval iteration algorithm for MDPs and IMDPs. Theor. Comput. Sci. **735**, 111–131 (2018). https://doi.org/10.1016/j.tcs.2016.12.003
3. Hartmanns, A., Kaminski, B.L.: Optimistic value iteration. In: CAV (2). LNCS, vol. 12225, pp. 488–511. Springer (2020). https://doi.org/10.1007/978-3-030-53291-8_26
4. Quatmann, T., Katoen, J.: Sound value iteration. In: CAV (1). LNCS, vol. 10981, pp. 643–661. Springer (2018). https://doi.org/10.1007/978-3-319-96145-3_37