Bisimulation Minimisation *Still* Mostly Speeds Up Probabilistic Model Checking

Adnan Ahmed, Hiva Karami, Anto Nanah Ji, and Franck van Breugel

DisCoVeri Group, Department of Electrical Engineering and Computer Science
York University, Toronto, Canada

Abstract. Katoen, Kemna, Zapreev, and Jansen have empirically shown that identifying states that are probabilistic bisimilar generally reduces the size of discrete-time Markov chains (DTMCs) and speeds up probabilistic model checking of those DTMCs. In particular, they have demonstrated that the time it takes the probabilistic model checker MRMC to reduce the DTMC and check a property expressed in the logic PCTL for the reduced DTMC is often less than the time MRMC needs to check the property for the original DTMC. In this paper, we reproduce those results. Furthermore, we show that these results also hold for the probabilistic model checkers PRISM, as well as our extension of PRISM with implementations of algorithms to decide probabilistic bisimilarity from the literature.

Keywords: probabilistic model checking · probabilistic bisimilarity · discrete-time Markov chain · PCTL · MRMC · PRISM.

1 Introduction

A *model checker* determines whether a model of a system satisfies a property. A large variety of systems, including those involving software and hardware but also biological as well as chemical systems, are considered. The models of these systems formalize (an abstraction of) the states in which the system can be and how the system can evolve transitioning from one state to another. The property considered by the model checker is often expressed as a formula of a logic.

In this paper we focus on explicit state model checkers. Roughly, these model checkers systematically visit the states of the model searching for a violation of the property. The number of states of a model can be humongous. This is known as the *state space explosion problem*. One approach to tackle this problem is reducing the size of the model by identifying states that satisfy the same properties. As has been shown by Fisler and Vardi [5], for models known as transition systems this approach is ineffective as the cost of performing the reduction of the model outweighs that of checking the property for the model.

For models known as *discrete-time Markov chains* (DTMC), which can model systems that rely on randomness, Katoen, Kemna, Zapreev, and Jansen [10] have empirically shown that the reduction of the size of these probabilistic models

often is effective. They identify states that are *probabilistic bisimilar*, an equivalence relation that captures similarity of behaviour due to Larsen and Skou [13]. Probabilistic bisimilarity preserves PCTL, a logic that can express a large array of properties involving probabilities and was introduced by Hansson and Jonsson [6]. That is, states that are probabilistic bisimilar satisfy the same PCTL formulas.

Katoen et al. use the probabilistic model checker MRMC [11] in their experiments. They consider four types of DTMCs and for each a property expressed in PCTL. In the experiments for each DTMC and PCTL formula they compare (1) the time taken to decide probabilistic bisimilarity for the DTMC, reduce the size of the DTMC by identifying probabilistic bisimilar states, and check the PCTL formula for the reduced DTMC with (2) the time taken to check the PCTL formula for the original DTMC. Their experimental results show that (1) is often less than (2), from which they conclude that bisimulation minimisation, that is, reducing the size of the DTMC by identifying probabilistic bisimilar states, mostly speeds up probabilistic model checking. More details can be found in [11] as well as [18, Chapter 4] and [19].

The results of Katoen et al. are 18 years old. Since 2007, new algorithms to decide probabilistic bisimilarity have been introduced, new probabilistic model checkers have been developed, existing probabilistic model checkers have been improved, and computer hardware has made strides as well. Therefore, we ask the question whether bisimulation minimisation *still* mostly speeds up probabilistic model checking.

To answer this question, we not only consider the probabilistic model checker MRMC, as Katoen et al. did, but also the currently most popular probabilistic model checker PRISM [12]. Furthermore, we also consider our extension of PRISM with implementations of other algorithms to decide probabilistic bisimilarity: the one by Buchholz [1], the original algorithm of Derisavi, Hermanns, and Sanders [3], their algorithm that uses red black trees instead of splay trees, and the algorithm introduced by Valmari and Franceschinis [17].

As we will see, the results of Katoen et al. are reproducible. Furthermore, we will show that their results also hold for other probabilistic model checkers such as PRISM, as well as our extension of PRISM.

2 Probabilistic Model Checkers

In our experiments, we not only consider MRMC[1], but also PRISM[2]. Both support DTMCs as models and PCTL formulas as properties. Next, we briefly discuss these probabilistic model checkers.

In our experiments, we make use of the latest version of MRMC, namely version 1.5. In MRMC, the algorithm of Derisavi, Hermanns, and Sanders [3] to decide probabilistic bisimilarity has been implemented.

[1] mrmc-tool.org
[2] www.prismmodelchecker.org

We also employ the latest version of PRISM, version 4.8.1.dev. This probabilistic model checker contains an implementation of a variation on Derisavi's algorithm to decide probabilistic bisimilarity [2]. We have made one change to PRISM so that the reduced DTMCs produced by MRMC and PRISM have the same size: we set the desired accuracy when comparing sums of probabilities in the probabilistic bisimilarity decision procedure to 10^{-5}. We have extended PRISM[3] with implementations of other algorithms to decide probabilistic bisimilarity: the one by Buchholz [1], the original algorithm of Derisavi, Hermanns, and Sanders [3], their algorithm that uses red black trees instead of splay trees, and the algorithm introduced by Valmari and Franceschinis [17].

To check that our implementations of these minimisation algorithms are correct, we generated millions of random DTMCs. For each random DTMC, ran the five minimisation algorithms. We checked that the five resulting minimized DTMCs have the same number of states and transitions.

3 Models and Properties

In our experiments, we use most of the DTMCs and PCTL properties that have been considered in [10]. Below, we briefly introduce them.

3.1 Crowds Protocol

This protocol was introduced by Reiter and Rubin in [15]. It protects anonymity of users of the web, to some degree, by grouping some internet users, known as a crowd, for collectively issuing requests on behalf of its members. A request to a web server from a crowd member is first passed to another random group member who can either submit the request or delegate the handling of the request to yet another randomly chosen crowd member. As a result, the request is eventually submitted by a random member of the crowd.

Shmatikov [16] modelled the protocol in PRISM and distinguished between honest and corrupt crowd members. The protocol has two parameters: N, the size of the crowd, and R, the number of runs of the protocol. PRISM was used to determine whether the probability of corrupt members eventually, say within 10000 steps, observing messages along different routing paths originate from a particular member is smaller than 0.5 — the probability of identifying the sender increases as this number of paths and probability grow (see [16] for details). This property can be expressed in PCTL's original syntax as $F^{\leq 10000}_{<0.5}$ `observe`, where the atomic proposition `observe` captures that a corrupt crowd member observes messages from a particular member more than once. In MRMC's syntax, the property is expressed as `P{<0.5}[tt U[0,10000] observe]` and the property can be expressed in PRISM as `P<0.5[F<=10000 "observe"]`.

[3] Our extension of PRISM is available at github.com/Lassonde-Undergraduate-Research-2024/PRISM-V2.

3.2 Leader Election Protocol

Given a ring of indistinguishable processors, the protocol of Itai and Rodeh [9] elects a leader among the processors by sending messages synchronously along the ring. To break symmetries, the protocol relies on randomness. In particular, each processor randomly chooses an integer between one and K, where K is a parameter of the protocol. The protocol has one other parameter: N, the number of processors.

The probability that the protocol elects a leader within a particular number of steps is a property of interest. For example, one can express in PCTL that the probability that the protocol elects a leader within 5000 steps is greater than 0.5 by $F^{\leq 5000}_{>0.5}$ `elected`, where the atomic proposition `elected` captures that the protocol has elected a leader. In MRMC's syntax, the property is expressed as `P{>0.5}[tt U[0,5000] elected]` and the property can be expressed in PRISM as `P>0.5[F<=5000 "elected"]`.

3.3 Randomised Mutual Exclusion

Pnueli and Zuck [14] present a mutual exclusion protocol of processes. It was designed so that it gives rise to a small state space and is easy to verify. The protocol has a single parameter: N, the number of processes.

Probabilistic model checkers such as MRMC and PRISM allow us to determine the probability that process 1 is the first to enter the critical section. For example, we can express the property that the probability that process 1 is the first to enter the critical section within 10000 steps is greater than 0.5 can be expressed in PCTL as `notEnter1` $U^{\leq 10000}_{>0.5}$ `enter1`, where the atomic proposition `enter1` captures that process 1 enters the critical section and the atomic proposition `notEnter1` represents that the other processes do not enter the critical section. In MRMC's syntax, the property is expressed as `P{> 0.5}[notEnter1 U[0,10000] enter1]` and the property can be expressed in PRISM as `P>0.5["notEnter1" U<=10000 "enter1"]`.

The above three sample DTMCs were also considered in [10]. A fourth sample DTMC considered in op. cit. was obtained from a continuous-time Markov chain (CTMC) via uniformization. We were unable to obtain a DTMC with the number of states reported in op. cit.

4 Experiments and Results

The experiments were run on an Intel machine with an i7-8700T CPU and 15 GB of RAM. We run the probabilistic model checkers on the above mentioned models with the above described properties *with* and *without* bisimulation minimisation. Rather than just measuring the time it takes to do the bisimulation minimisation and checking the property as done in [10], we also consider the time to read the files that contain the transitions and labelling, build the model, etc. As we will see, even if we include this additional time, bisimulation minimisation still speeds

up probabilistic model checking for these examples. We run each experiment 50 times and report the average speed up (the amount it takes without minimisation / the amount of time it takes with minimisation) or slow down (− the amount of time it takes with minimisation / the amount it takes without minimisation), as well as the standard deviation. We only report the results for those experiments that took more than 0.01 second and less than ten minutes. Since some of the models have a large number of states, we set the maximum heap size for Java to 6G.

The PRISM model of the crowds protocol labels some states with `deadlock`. This label is treated differently from the other labels. To ensure that states labelled with `deadlock` are treated in MRMC as they are in PRISM, we add self-loops in states labelled with `deadlock` and subsequently we remove the `deadlock` label in the MRMC model. As a result, the number of states of the reduced DTMC for the crowds protocols differ slightly from those reported in [10].

4.1 Crowds Protocol

As we can see in the table below, the reduced DTMCs are significantly smaller than the original DTMCs. As we already mentioned, these numbers differ slightly from those reported in [10]. For example, in case $N = 5$ and $R = 3$, in our case the reduced model has 72 states and 105 transitions whereas op. cit. reported 53 states.

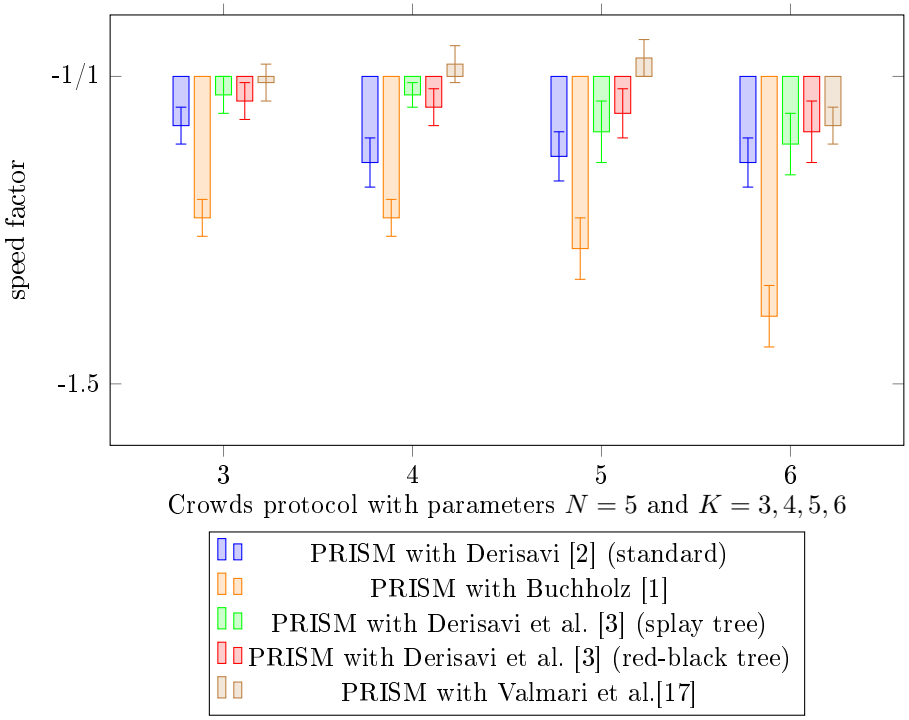
		original DTMC		reduced DTMC		reduction factor	
N	R	states	transitions	states	transitions	states	transitions
5	3	1198	2038	72	105	16.6	19.4
5	4	3515	6035	140	209	25.1	28.9
5	5	8653	14953	230	348	37.6	43.0
5	6	18817	32677	333	507	56.5	64.5
10	3	6563	15143	81	126	81.0	120.2
10	4	30070	70110	160	256	187.9	273.9
10	5	111294	261444	265	431	420.0	606.6
10	6	352535	833015	396	651	890.2	1279.6
15	3	19228	55948	81	126	237.4	444.0
15	4	119800	352360	160	256	748.8	1376.4
15	5	592060	1754860	265	431	2234.2	4071.6
15	6	2464168	7347928	396	651	6222.6	11287.1

In the table below, we present MRMC's speed up factors for the crowds protocol. We do not report on the cases that $N = 5$ and $R = 3$ or $R = 4$, because those experiments took less than 0.01 second. For all experiments we observe an average speed up factor between 4 and 5, which is significant.

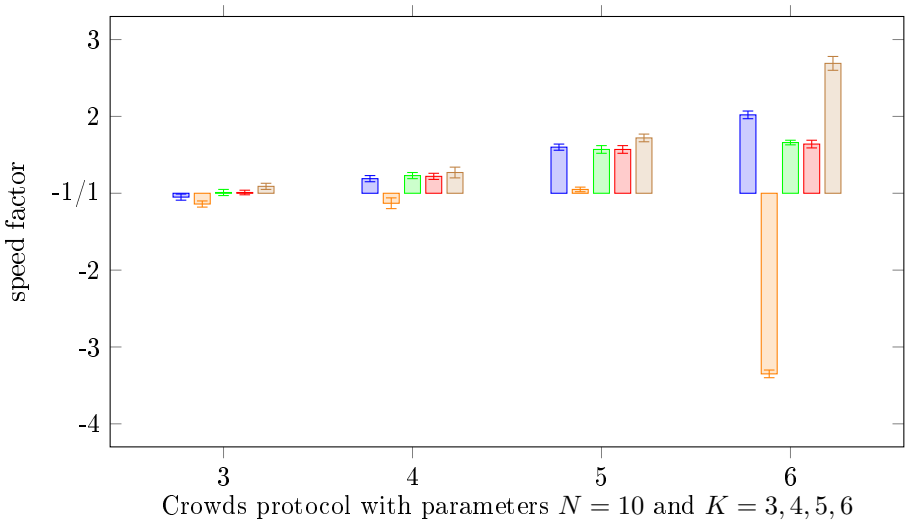
N	5		5		10		10		10	
R	5		6		3		4		5	
	avg	std	avg	std	avg	std	avg	std	avg	std
	4.22	0.42	4.34	0.74	4.80	0.49	4.50	0.65	4.13	0.23

N	10		15		15		15		15	
R	6		3		4		5		6	
	avg	std	avg	std	avg	std	avg	std	avg	std
	4.44	0.16	4.09	0.27	4.41	0.43	4.43	0.32	4.06	0.05

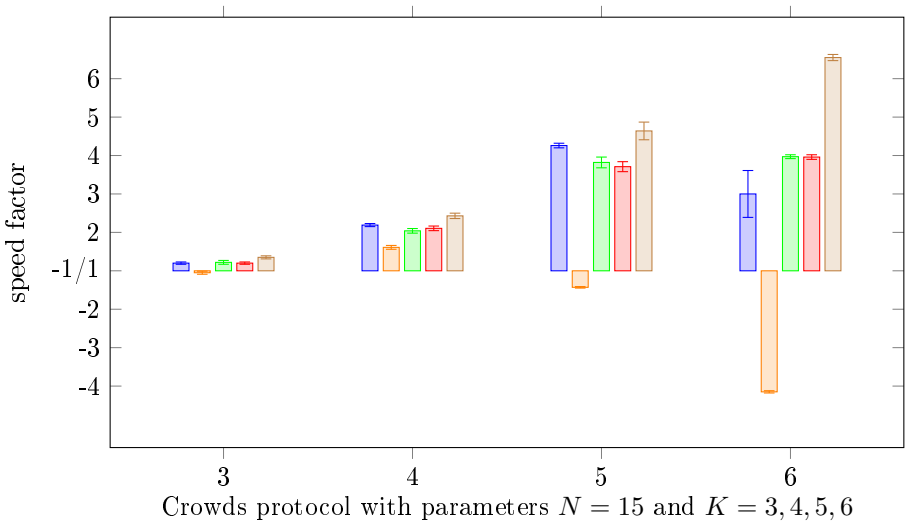
In the graph below, we present PRISM's average speed up or slow down as well as the standard deviation for the crowds protocol where the crowd has size 5. If the bar goes up, bisimulation minimisation speeds up probabilistic model checking, whereas if the bar goes down, bisimulation minimisation slows it down. None of the algorithms show any significant speed up. The Buchholz minimisation algorithm performs worst with an average slow down factor of 1.39 for $R = 6$.



Next, we present the results when the crowd has size 10. Apart from the Buchholz minimisation algorithm, all others speed up.



Finally, we consider a crowd size of 15 for this protocol. Also in this case, the Buchholz minimisation algorithm does not speed up in most cases. The minimisation algorithm by Valmari et al. performs very well with speed up factors even larger than those achieved by MRMC.



4.2 Leader Election Protocol

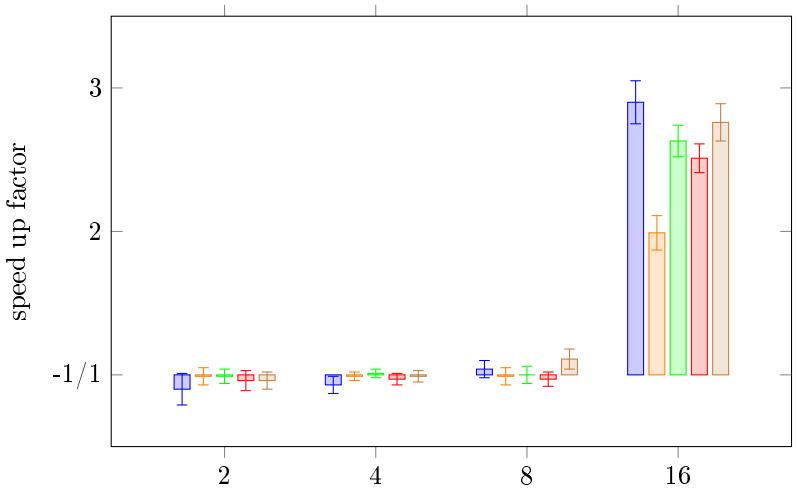
Also for this example, the reduced DTMCs are much smaller than the original DTMCs, as can be seen in the table below. Here, the numbers exactly match the ones reported in [10].

		original DTMC		reduced DTMC		reduction factor	
N	K	states	transitions	states	transitions	states	transitions
4	2	55	70	10	11	5.5	6.4
4	4	782	1037	10	11	78.2	94.3
4	8	12302	16397	10	11	1230.2	1490.6
4	16	196622	262157	10	11	19662.2	23832.5
5	2	162	193	12	13	13.5	14.8
5	4	5122	6145	12	13	426.8	472.7
5	6	38882	46657	12	13	3240.2	3589.0
5	8	163842	196609	12	13	13653.5	15123.8

In the table below, we present MRMC's speed up factors for the leader election protocol. Minimisation gives rise to average speed up factors larger than 25.

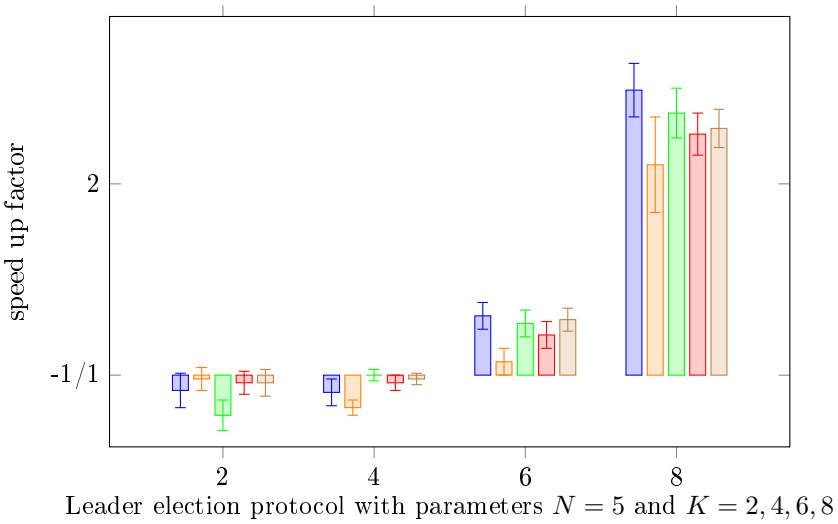
N	4		4		5		5	
K	8		16		6		8	
	avg	std	avg	std	avg	std	avg	std
	27.47	2.22	24.10	2.21	21.51	6.01	24.47	1.02

Let us now turn to PRISM. First, we consider the leader election protocol for 4 processors. As can be concluded from the graph below, PRISM sees a speed up, especially for $K = 16$. Although PRISM's original minimisation algorithm shows the largest speed up, our implementations of the other minimisation algorithms show speed as well. For this example, MRMC achieves a much larger speed up than PRISM.



Leader election protocol with parameters $N = 4$ and $K = 2, 4, 8, 16$

Next, we consider the leader election protocol for 5 processors. The results are similar.



4.3 Randomised Mutual Exclusion

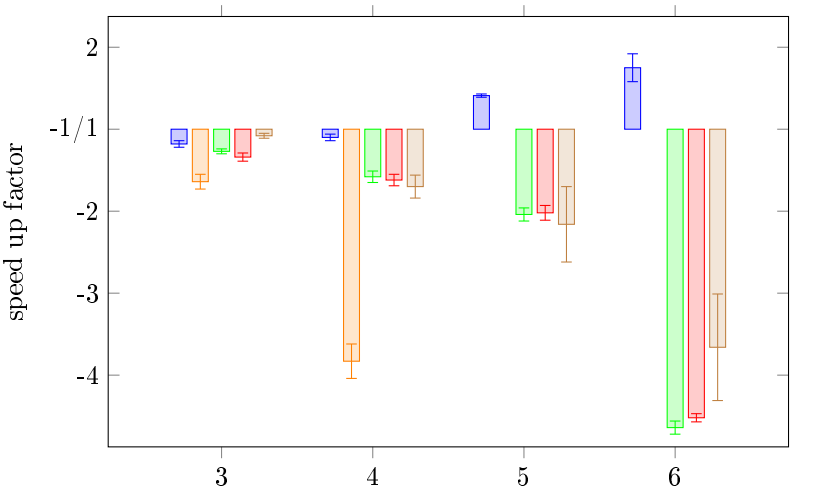
Again we see a reduction is the size of the DTMC. Also for this example, our numbers match those reported in [10].

	original DTMC		reduced DTMC		reduction factor	
N	states	transitions	states	transitions	states	transitions
3	2368	8272	1123	3846	2.1	2.1
4	27600	123883	5224	21796	5.3	5.7
5	308800	1680086	18501	88391	16.7	19.0
6	3377344	21514489	54706	289022	61.7	74.4

In the table below, we present MRMC's speed up factors for the randomized mutual exclusion protocol. Minimisation gives rise to some speed up, although the average speed up factors are much smaller than for the other two protocols.

4		5		6	
avg	std	avg	std	avg	std
1.47	0.13	1.63	0.04	1.53	0.01

For this final protocol, we see that only PRISM's original minimisation algorithm achieves a speed up. For that algorithm, the average speed up factors are comparable to those of MRMC. We do not depict the average slow down factor for the Buchholz minimisation algorithm for $N = 5$, which is 27.18, as it would distort the graph. We do not report the average slow down factor for the Buchholz minimisation algorithm for $N = 6$ since each experiment takes more than ten minutes.



Random mutual exclusion protocol with parameters $N = 3, 4, 5, 6$

5 Conclusion

Not only have we been able to reproduce the results of Katoen et al. of 18 years ago, demonstrating that bisimulation minimisation speeds up probabilistic model checking, but we have also shown that these results stand the test of time, even though probabilistic model checkers have been improved, new algorithms to decide probabilistic bisimilarity have been developed, and computer hardware has changed.

We consider three examples: the crowds protocol, the leader election protocol, and randomized leader election. MRMC's bisimulation minimisation performs well for all three. For PRISM the situation is less clear cut. PRISM's original minimisation algorithm performs fairly well, but is sometimes outperformed by the minimisation algorithm of Valmari et al. The two versions of the minimisation algorithm of Derisavi et al., the one using splay trees and the other using red-black trees, show very similar performance. The Buchholz minimisation algorithm often shows the poorest performance.

In the future, we would like to extend our work in two directions. First of all, we plan to also consider the probabilistic model checker Storm[4] [8] in our experiments. Furthermore, we intend to consider numerous other examples of probabilistic protocols and randomized algorithms such as the ones that can be found in the Quantitative Verification Benchmark Set[5] [7] as well as the examples accompanying jpf-probabilistic[6] [4].

[4] www.stormchecker.org
[5] github.com/ahartmanns/qcomp
[6] github.com/javapathfinder/jpf-probabilistic

Acknowledgements. The authors would like to thank the referee for their constructive feedback. This research was supported by the Natural Sciences and Engineering Research Council of Canada.

References

1. Buchholz, P.: Efficient computation of equivalent and reduced representations for stochastic automata. International Journal of Computer Systems Science and Engineering **15**(2), 93–103 (Apr 2000)
2. Derisavi, S.: Signature-based symbolic algorithm for optimal Markov chain lumping. In: Proceedings of the 4th International Conference on the Quantitative Evaluation of Systems. pp. 141–150. IEEE, Edinburgh, UK (Sep 2007)
3. Derisavi, S., Hermanns, H., Sanders, W.: Optimal state-space lumping in Markov chains. Information Processing Letters **87**(6), 309–315 (Sep 2003)
4. Fatmi, S., Chen, X., Dhamija, Y., Wildes, M., Tang, Q., Breugel, F.v.: Probabilistic model checking of randomized Java code. In: Laarman, A., Sokolova, A. (eds.) Proceedings of the 27th International Symposium on Model Checking Software. Lecture Notes in Computer Science, vol. 12864, pp. 157–174. Springer-Verlag (Jul 2021)
5. Fisler, K., Vardi, M.: Bisimulation minimization in an automata-theoretic verification framework. In: Gopalakrishnan, G., Windley, P. (eds.) Proceedings of the 2nd International Conference on Formal Methods in Computer-Aided Design. Lecture Notes in Computer Science, vol. 1522, pp. 115–132. Springer-Verlag, Palo Alto, CA, USA (Nov 1998)
6. Hansson, H., Jonsson, B.: A framework for reasoning about time and reliability. In: Proceedings of the Real-Time Systems Symposium. pp. 102–111. IEEE, Santa Monica, CA, USA (Dec 1989)
7. Hartmanns, A., Klauck, M., Parker, D., Quatmann, T., Ruijters, E.: The quantitative verification benchmark set. In: Vojnar, T., Zhang, L. (eds.) Proceedings of the 25th International Conference on Tools and Algorithms for the Construction and Analysis of Systems. Lecture Notes in Computer Science, vol. 11427, pp. 344–350. Springer-Verlag, Prague, Czech Republic (Apr 2019)
8. Hensel, C., Junges, S., Katoen, J.P., Quatmann, T., Volk, M.: The probabilistic model checker Storm. International Journal on Software Tools for Technology Transfer **24**(4), 589–610 (Aug 2022)
9. Itai, A., Rodeh, M.: Symmetry breaking in distributed networks. In: Proceedings of the 2nd Annual Symposium on Foundations of Computer Science. pp. 150–158. IEEE, Nashville, TN, USA (Oct 1981)
10. Katoen, J.P., Kemna, T., Zapreev, I., Jansen, D.: Bisimulation minimisation mostly speeds up probabilistic model checking. In: Grumberg, O., Huth, M. (eds.) Proceedings of the 13th International Conference on Tools and Algorithms for the Construction and Analysis of Systems. Lecture Notes in Computer Science, vol. 4424, pp. 87–101. Springer-Verlag, Braga, Portugal (Mar/Apr 2007)
11. Katoen, J.P., Khattri, M., Zapreev, I.: A Markov reward model checker. In: Proceedings of the 2nd International Conference on the Quantitative Evaluation of Systems. pp. 243–244. IEEE, Torino, Italy (Sep 2005)
12. Kwiatkowska, M., Norman, G., Parker, D.: Prism 4.0: Verification of probabilistic real-time systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) Proceedings of the 23rd International Conference on Computer Aided Verification. Lecture Notes in

Computer Science, vol. 6806, pp. 585–591. Springer-Verlag, Snowbird, UT, USA (Jul 2011)

13. Larsen, K., Skou, A.: Bisimulation through probabilistic testing. In: Proceedings of the 16th Annual ACM Symposium on Principles of Programming Languages. pp. 344–352. ACM, Austin, TX, USA (Jan 1989)
14. Pnueli, A., Zuck, L.: Verification of multiprocess probabilistic protocols. In: Kameda, T., Misra, J., Peters, J., Santoro, N. (eds.) Proceedings of the 3rd Annual ACM Symposium on Principles of Distributed Computing. pp. 12–27. ACM, Vancouver, Canada (Aug 1984)
15. Reiter, M., Rubin, A.: Crowds: anonymity for web transactions. ACM Transactions on Information and System Security **1**(1), 66–92 (Nov 1998)
16. Shmatikov, V.: Probabilistic analysis of anonymity. In: Proceedings of the 15th IEEE Computer Security Foundations Workshop. pp. 119–128. IEEE, Cape Breton, Canada (Jun 2002)
17. Valmari, A., Franceschinis, G.: Simple $O(m \log n)$ time Markov chain lumping. In: Esparza, J., Majumdar, R. (eds.) Proceedings of the 16th International Conference on Tools and Algorithms for the Construction and Analysis of Systems. Lecture Notes in Computer Science, vol. 6015, pp. 38–52. Springer-Verlag, Paphos, Cyprus (Mar 2010)
18. Zapreev, I.: Model Checking Markov Chains: Techniques and Tools. Ph.D. thesis, University of Twente, Enschede, the Netherlands (Mar 2008)
19. Zapreev, I., Jansen, C.: MRMC test suite, version 1.5 (Jan 2011), available at mrmc-tool.org

A Detailed Results

Below, we provide the details for the five PRISM minimisation algorithms. As in the main part, we use negative speed ups to indicate that probabilistic model checking without bisimulation minimisation is faster than with bisimulation minimisation. For example, for the crowds protocol, for $N = 5$ and $R = 6$, probabilistic model checking without bisimulation minimisation is faster than probabilistic model checking with bisimulation minimisation using Buchholz's algorithm by a factor 1.39. If probabilistic model checking with bisimulation minimisation is faster than without bisimulation minimisation, then we report a positive speed up. For example, for the crowds protocol, for $N = 10$ and $R = 6$, probabilistic model checking with bisimulation minimisation using Valmari et al.'s algorithm is faster than probabilistic model checking without bisimulation minimisation by a factor 2.76. For each row, we use red for the algorithm with the best average.

A.1 Crowds Protocol

		PRISM		Buchholz		Derisavi				Valmari	
						splay		red-black			
N	R	avg	std	avg	std	avg	std	avg	std	avg	std
5	3	-1.08	0.03	-1.23	0.03	-1.03	0.03	-1.04	0.03	-1.01	0.03
5	4	-1.14	0.04	-1.23	0.03	-1.03	0.02	-1.05	0.03	1.02	0.03
5	5	-1.13	0.04	-1.28	0.05	-1.09	0.05	-1.06	0.04	1.03	0.03
5	6	-1.14	0.04	-1.39	0.05	-1.11	0.05	-1.09	0.05	-1.08	0.03
10	3	-1.05	0.04	-1.14	0.04	1.01	0.04	1.01	0.03	1.09	0.04
10	4	1.19	0.04	-1.13	0.07	1.23	0.04	1.22	0.04	1.27	0.07
10	5	1.60	0.04	1.05	0.03	1.57	0.05	1.57	0.05	1.72	0.05
10	6	2.02	0.05	-3.35	0.05	1.66	0.03	1.64	0.05	2.69	0.09
15	3	1.20	0.03	-1.05	0.04	1.22	0.05	1.20	0.03	1.35	0.04
15	4	2.19	0.04	1.61	0.05	2.04	0.06	2.05	0.06	2.43	0.07
15	5	4.26	0.06	-1.43	0.02	3.82	0.14	3.71	0.13	4.64	0.23
15	6	3.00	0.61	-4.15	0.03	3.97	0.05	3.96	0.06	6.55	0.08

A.2 Leader Election Protocol

		PRISM		Buchholz		Derisavi				Valmari	
						splay		red-black			
N	K	avg	std	avg	std	avg	std	avg	std	avg	std
4	2	-1.10	0.11	-1.01	0.06	-1.01	0.05	-1.04	0.07	-1.04	0.06
4	4	-1.07	0.06	-1.01	0.03	1.01	0.03	-1.03	0.04	-1.01	0.04
4	8	1.04	0.06	-1.01	0.06	-1.00	0.06	-1.03	0.05	1.11	0.07
4	16	2.90	0.15	1.99	0.12	2.63	0.11	2.51	0.10	2.76	0.13
5	2	-1.08	0.09	-1.02	0.06	-1.21	0.08	-1.04	0.06	-1.04	0.07
5	4	-1.09	0.07	-1.17	0.04	-1.00	0.03	-1.04	0.04	-1.02	0.03
5	6	1.31	0.07	1.07	0.07	1.27	0.07	1.21	0.07	1.29	0.06
5	8	2.49	0.14	2.10	0.25	2.37	0.13	2.26	0.11	2.29	0.10

A.3 Randomized Mutual Exclusion

	PRISM		Buchholz		Derisavi				Valmari	
					splay		red-black			
N	avg	std	avg	std	avg	std	avg	std	avg	std
3	-1.18	0.04	-1.64	0.09	-1.27	0.03	-1.34	0.05	-1.08	0.03
4	-1.10	0.04	-3.82	0.21	-1.58	0.07	-1.62	0.07	-1.70	0.14
5	1.41	0.02	-27.18	1.04	-2.04	0.08	-2.02	0.09	-2.16	0.46
6	1.75	0.17			-4.64	0.08	-4.52	0.05	-3.66	0.65

There is no entry for the Buchholz minimisation algorithm for $N = 6$ as the experiment took longer than ten minutes.