Backwards Reachability for Probabilistic Timed Automata: A Replication Report*

Arnd Hartmanns and Bram Kohlen

University of Twente, Enschede, The Netherlands

Abstract. Backwards reachability is an efficient zone-based approach for model checking probabilistic timed automata w.r.t. PTCTL properties. Current implementations, however, are restricted to maximum probabilities of reachability properties. In this paper, we report on our new implementation of backwards reachability as part of the Modest Toolset. Its support for minimum and maximum probabilities of until formulas makes it the most general implementation available today. We compare its behaviour to the experimental results reported in the original papers presenting the backwards reachability technique.

1 Introduction

Probabilistic timed automata (PTA) [9] combine the capabilities of Markov decision processes (MDPs) [12] to model decision-making under uncertainty with the real-time features of timed automata (TA) [1]. As the state space of a PTA is uncountably infinite, model checkers transform PTA to equivalent finite-state models. The region graph [9], zone-based forwards reachability [9], zone-based backwards reachability [11], and digital clocks [8] transformations map to MDPs. Since region graphs are too large, and the forwards approach merely delivers upper bounds on maximum reachability probabilities, only the latter two approaches are relevant today. Although limited to closed clock constraints and reachability properties, the digital clocks approach is easy to implement and also supports expected-reward properties. Backwards reachability supports the full logic PTCTL; and while its complexity is doubly exponential in the number of clocks, it tends to be efficient in practice and competitive with digital clocks [6]. The game-based abstraction technique [6] maps to stochastic games.

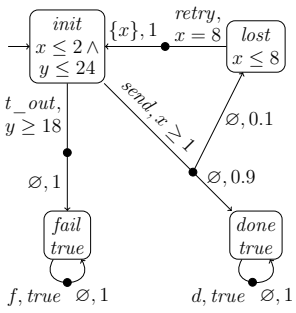
In this paper, we focus on the backwards reachability approach: it poses no restrictions on the PTA, supports all of PTCTL, and maps to MDPs. The latter is favourable for methods like parametric model checking that currently work on MDPs only [5]. However, there is currently no tool that fully implements backwards reachability. PRISM's [7] backwards engine is restricted to maximum probabilities and the "eventually" operator (lacking support for "until"), and does

* Authors are listed in alphabetical order. This work was funded by NWO grant OCENW.KLEIN.311, NWO VENI grant 639.021.754, and the EU's Horizon 2020 research and innovation programme under MSCA grant agreement no. 101008233.

not support models with global variables. The prototype implementation of [11] also supported minimum probabilities, but is no longer available.

In this paper, we report on our replication of backwards reachability in the Modest Toolset [4] that computes minimum and maximum probabilities of until formulas. We compare its behaviour to the original prototype based on the experiments of [11] and its earlier conference version [10]. While we obtain the same probabilities, our implementation's behaviour and performance is notably different. We in particular found that the original papers omit the algorithm for timed predecessors, which is central to the approach, and rather involved.

2 Probabilistic Timed Automata



We show an example PTA $\mathcal{P}$ representing a basic communication protocol with message loss on the right. Its initial *location* is *init*; it has two clocks x and y. The state of a PTA pairs the current location ℓ with the current *clock valuation* v, which maps every clock to its non-negative real value. All paths start in $\langle init, v_0 \rangle$ where v_0 assigns value 0 to x and y. Letting $t > 0$ time units elapse in a location ℓ corresponds to the transition from $\langle \ell, v \rangle$ to $\langle \ell, v + t \rangle$ where $(v + t)(z) = v(z) + t$ for each clock z. Every location has an *invariant* constraining the passage of time: time can only pass while the invariant remains satisfied. The invariant of *init* is $x \leq 2 \wedge y \leq 24$. Location *init* has two *edges* labelled with actions *send* and *t_out*. Picking action *send* takes us to *done* with probability 0.9 and to *lost* with probability 0.1. Action *send* is only available when the edge's *guard*, $x \geq 1$, is satisfied. Action *retry* brings us back to *init* with probability 1 and *resets* the value of x to zero.

Due to the delay transitions, a PTA's state space is uncountably infinite. To make the analysis of PTA tractable, we group clock valuations satisfying a constraint into *zones*. For example, the zone over constraint $x \leq 2 \wedge y \leq 24$ contains v with $v(x) = 1.5$ and $v(y) = 23.9$, but not v' with $v'(x) = 1.5$ and $v'(y) = 24.1$. Given zone ζ, the *symbolic state* $\langle \ell, \zeta \rangle$ contains all states $\langle \ell, v \rangle$ where $v \in \zeta$. From now on, we use the notation for constraints to denote zones, too. A common data structure to represent a zone is the *difference bound matrix* (DBM) [2]. A DBM's entries are pairs of an integer and an operator $\lesssim \in \{ <, \leq \}$. Entry $A_{i,j} = \langle d, \lesssim \rangle$ means that $x_i - x_j \lesssim d$, given a bijection between the PTA's n clocks and $x_1, \ldots, x_n$. x_0 represents the clock that always has value zero.

We focus on model checking PTA w.r.t. *until formulas*. Property $\neg \mathit{avoid}\ \mathsf{U}\ \mathit{target}$ characterises the paths that eventually reach a state satisfying formula *target* over locations and clock constraints without entering an *avoid* state before. If *avoid* is *false*, then we have a *reachability* formula of the form $\Diamond \mathit{target}$. The properties we aim to check are statements about the probability of the set of paths characterised by a formula being smaller or larger than a given bound. For example, property $\mathbb{P}_{\geq 0.99}(\Diamond\ done)$ tests whether the probability to eventually

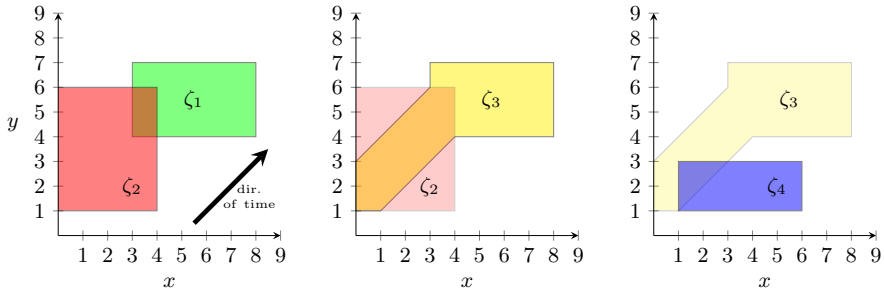


Fig. 1. Time and discrete predecessors of zones.

reach location *done* for *any* choice of actions is greater than or equal to 0.99. That is, the *minimal probability* $\mathbb{P}_{\min}$ must be at least 0.99. This is true for $\mathcal{P}$, as it must perform *send* at least twice before *t_out* is enabled. Conversely, $\mathbb{P}_{\leq 0.99}(\Diamond\, done)$ states that the *maximal probability* $\mathbb{P}_{\max}$ to reach location *done* is at most 0.99. This is false for $\mathcal{P}$ it is possible to perform *send* three times which has a probability of 0.999 to reach *done*. However, $z.\mathbb{P}_{\leq 0.99}(\Diamond\, done \wedge z \leq 10)$, where z is a *property clock*, is true as we can only send twice in 10 time units.

3 Backwards Reachability

The *backwards reachability* approach transforms a PTA into an MDP via a backwards search starting at the *target* symbolic state. It then discovers those symbolic states that can reach *target* by performing a transition and letting time pass. This procedure is iterated until all backwards-reachable states are discovered. We refer to this as the *MaxU* algorithm [11, Fig. 5]; it builds an MDP whose states are symbolic states of the PTA. It needs to compute *time predecessors* and *discrete predecessors*. The former operation, denoted $tpre_{\langle \ell, \zeta' \rangle}(\langle \ell, \zeta \rangle)$, calculates the set of clock valuations that eventually reach ζ without leaving ζ'. In Fig. 1, we visualise zones $\zeta_1 = (3 \leq x \leq 8 \wedge 4 \leq y \leq 7)$ and $\zeta_2 = (x \leq 4 \wedge 1 \leq y \leq 6)$ on the left. In the middle, we show $\zeta_3 = tpre_{\langle \ell, \zeta_2 \rangle}(\langle \ell, \zeta_1 \rangle)$. The discrete predecessor operation, denoted $dpre(\langle \ell', a, X, \ell \rangle, \langle \ell, \zeta_3 \rangle)$, makes a backwards transition over the specified edge and branch from ℓ' to ℓ. It performs a *backwards reset* on the clocks in X. On the right of Fig. 1, we show $\langle \ell', \zeta_4 \rangle = dpre(\langle \ell', \alpha, \{x\}, \ell \rangle, \langle \ell, \zeta_3 \rangle)$ for invariant $x \leq 6$ of ℓ' and guard $x \geq 1$ of α.

MaxU works for maximal probabilities only. We restructure properties requiring minimal probabilities to properties with maximal probabilities: To obtain, say, $\mathbb{P}_{\max}(\Diamond\, target)$, we compute $1 - \mathbb{P}_{\max}(\neg target\, \mathsf{U}\, \phi)$ using *MaxU* where $\phi = \mathbb{P}_{\max}(\Box \neg target) \geq 1$. The symbolic states satisfying ϕ can be precomputed via a graph analysis using the $MaxV_{\geq 1}$ algorithm [11, Fig. 7]. It iteratively removes those symbolic states that do not have a path inside $\neg target$ of duration $\geq c$, until a fixpoint is reached. $MaxV_{\geq 1}$ is parameterised by c, which can be any positive integer; the value of c influences the number of iterations and therefore

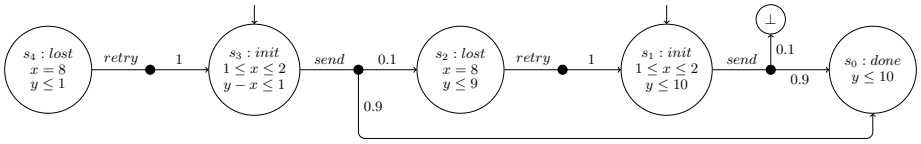


Fig. 2. Backwards reachability MDP for $\mathcal{P}$ and property $\mathbb{P}_{\leq 0.99}(\Diamond\, done \wedge y \leq 10)$.

the runtime. The optimal value of c is model- and property-dependent. Each iteration of $MaxV_{\geq 1}$ in turn calls the $MaxU_{\geq 1}$ algorithm [11, Fig. 4], which calculates whether an until formula is satisfied with maximal probability 1.

For property $\mathbb{P}_{\leq 0.99}(\Diamond\, done \wedge y \leq 10)$, we show the backwards reachability MDP of PTA $\mathcal{P}$ in Fig. 2. The target state is s_0. To obtain its predecessors we apply *tpre* followed by *dpre*. We have $s_0 = tpre(s_0)$ (omitting the subscript because it is only needed for until properties). We then apply *dpre* for the one incoming edge of *done*, i.e. $dpre(\langle init, send, \varnothing, done\rangle, tpre(s_0))$. No clocks are reset, so we only need to intersect zone $y \leq 10$ with guard $x \geq 1$ and invariant $x \leq 2 \wedge y \leq 24$ of *init*, which yields zone $1 \leq x \leq 2 \wedge y \leq 10$, making up state s_1. Next, $tpre(s_1) = \langle init, x \leq 2 \wedge y \leq 10 \wedge y - x \leq 9\rangle$ and for $dpre(\langle lost, retry, \{\, x \,\}, init\rangle, tpre(s_1))$, we perform a backwards reset on x, which will be any clock valuation where $y \leq 9$. Intersecting with guard and invariant yields s_2. The same principle applies to s_3 and s_4. However, since the zone of s_3 is a subset of the zone of s_1, it inherits the successors of s_1. Therefore, there is an edge directly to s_0. For s_4, the predecessor would have an empty zone, which means all predecessors have been discovered and the algorithm terminates.

4 Implementation

For our implementation of backwards reachability in the Modest Toolset, we followed the pseudocode of [11] as closely as possible, with some optimizations to eliminate a few obviously redundant calculations. We use DBMs to represent convex zones, which suffice for maximal reachability probabilities, and lists of DBMs whenever non-convex zones occur for until properties and minimum probabilities or due to disjunctions in constraints.

A major difficulty when working with lists of DBMs is the lack of a canonical form as highlighted in the left and middle parts of Fig. 3, where $[\zeta_1, \zeta_2, \zeta_3]$ contains the same valuations as $[\zeta_4, \zeta_5, \zeta_6]$. As a consequence, our implementation sometimes makes unnecessary iterations before recognising a fixpoint. Among the difficulties that we encountered in replicating [11] was that the pseudocode for *MaxU* is missing some operations. For example, line 14 adds a new predecessor state to the set of discovered states Z, but the journal version [11] does not add the corresponding edge to the set of discovered edges, whereas the 2004 paper version [10] correctly lists this step (line 15). Furthermore, [10] only gives a superficial intuitive explanation of *tpre*, omitting detailed pseudocode and implying that it should be trivial to implement. As illustrated on the right of Fig. 3

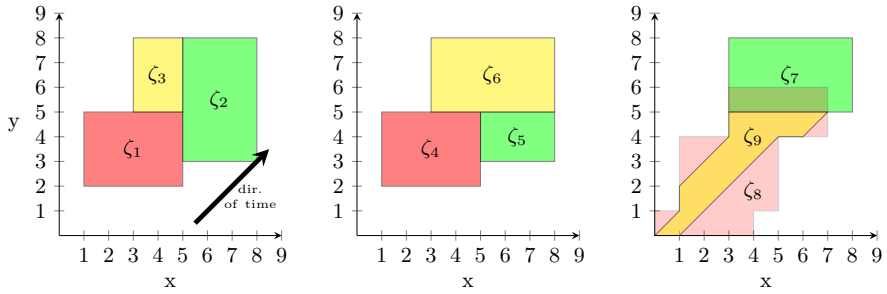


Fig. 3. Non-canonicity of lists of DBMs (left, middle) and *tpre* complications (right).

where $tpre_{\langle \ell, \zeta_8 \rangle}(\langle \ell, \zeta_7 \rangle) = \langle \ell, \zeta_7 \cup \zeta_9 \rangle$, however, it is a rather involved procedure. Finally, the PTA drawn for the CSMA benchmark model in [11] appear to be inaccurate: they include a clock x in some states that is never reset; for the model to make sense, we assumed this to be an error and replaced it with clock y.

5 Experiments

In [11], backwards reachability was benchmarked using models of the CSMA/CD and Firewire protocols for varying values of c and different time bounds. Additional benchmarks concerned a property transformation, and compared the state space sizes and generation times to forwards reachability and digital clocks.

We have attempted to replicate these results. Additionally, we benchmarked the entire model checking procedure, to evaluate the overall competitiveness of backwards reachability. However, we did not compare against forwards reachability and we did not test the property transformation. We have tested all the parameter values and properties considered in [11] and some additional ones, but we only list an interesting selection below. Since the original PRISM PTA models are not available, we recreated them from the visual representations in [11]. To compare with the digital clocks approach, we used the automatic digital clocks transformation in the mcsta tool of the Modest Toolset. This might yield models that are slightly different to the manually transformed ones used in the original benchmarks. All runtimes that we report are in seconds.

Table 1 lists the results of the $MaxV_{\geq 1}$ graph analysis necessary to calculate minimal reachability probabilities. This replicates [11, Table 1]. We tested additional values for c, because we noticed that our implementation behaved best for values between 400 and 800. The original sees the best results at $c = 50$, which was relatively slow in our benchmarks. We suspect that the differences can be mostly attributed to small differences in the recreated model.

Table 2 (partly) replicates [11, Table 2]. We checked $z.\mathbb{P}_{\sim\lambda}(\Diamond\, done \wedge z \leq D)$ and fixed c to 400. This is the value of c that produces the best results for our implementation; the value used for the original experiments was not specified. We notice a significant difference in the number of states. Our implementation

Table 1. Graph analysis data for CSMA/CD.

		$\mathbb{P}_{\geq 1}(\Diamond\, done)$			$\mathbb{P}_{\geq \lambda}(\Diamond\, done \wedge z \leq 2000)$		
			iterations			iterations	
bcmax	c	time	$MaxV_{\geq 1}$	$MaxU_{\geq 1}$	time	$MaxV_{\geq 1}$	$MaxU_{\geq 1}$
1	50	45.2	39	100	30.4	39	100
	400	18.2	7	20	9.1	7	20
	500	20.3	6	18	10.3	6	18
	600	23.7	6	18	11.4	6	18
	700	22.6	5	15	10.3	5	15
	800	24.6	5	15	10.9	5	15
2	50	534.4	41	106	394.0	41	106
	400	103.1	7	21	58.9	7	21
	500	110.2	6	18	59.1	6	18
	600	114.0	6	18	59.1	6	18
	700	111.2	5	15	55.5	5	15
	800	111.0	5	15	52.7	5	15

detects cases where the probability to reach the target is 0; this happens for lower values of D, resulting in very small state counts. For the other cases, the our implementation can explore more than twice as many states as the original.

One thing to note is that our backwards implementation sometimes outperforms digital clocks. As opposed to the original work, we benchmarked the time of the entire model checking procedure. For maximal probabilities, backwards reachability appears to outperform digital clocks especially for lower time bounds D. Digital clocks appears to scale better with a higher time bound. The claim of [11] that backwards reachability yields a much smaller state space than digital clocks does not hold any more due to mcsta implementing an unrolling-free technique for time-bounded reachability [3]. Such a method is not straightforwardly applicable to backwards reachability. We also note that the time benefit of backwards reachability is largely obtained by having a close to optimal value for c, which is not trivial to find.

Table 3 replicates the data of [11, Table 3]. These results are closer to the original. The differences in times can be attributed to us using a much faster

Table 2. States and model checking times of $z.\mathbb{P}_{\sim\lambda}(\Diamond\, done \wedge z{\leq}D)$ for CSMA/CD.

		backwards reachability				digital clocks		
bcmax	D	states (max)	time (max)	states (min)	time (min)	states	time (max)	time (min)
1	1200	1	0.0	3	13.2	10085	0.5	0.4
	1600	1	0.1	3	15.4	10085	0.6	0.5
	2000	1025	0.1	909	20.0	10085	0.7	0.6
	2400	1929	0.2	1741	20.9	10085	0.8	0.7
	2800	2833	0.5	2521	22.0	10085	0.9	0.8
2	1200	1	0.8	3	147.7	128553	4.7	4.5
	1600	1	5.9	3	169.3	128553	6.1	5.9
	2000	21750	9.5	28839	206.3	128553	7.7	7.4
	2400	27384	11.7	31783	220.2	128553	9.1	8.9
	2800	31744	16.2	34543	239.8	128553	10.6	10.4

processor. The number of iterations for $MaxV_{\geq 1}$ on property $\mathbb{P}_{\geq 1}(\Diamond\, done)$ corresponds exactly to the original results. However, the number of iterations of $MaxU_{\geq 1}$ is different. We are unsure of the cause; in particular, our implementation does produce the correct probabilities (compared to digital clocks). Our only explanation are differences in the model again. We also notice that the number of iterations for the time-bounded property $\mathbb{P}_{\geq \lambda}(\Diamond\, done \wedge z \leq 10000)$ in our implementation is equal to the number of iterations for its non-bounded counterpart. We came to the conclusion that this is intended as the input to $MaxV_{\geq 1}$ is equal for both properties, with the added time constraint. Our implementation keeps the time bound separate from the PTA, so it does not influence the number of iterations. We speculate that the original prototype or model may have added the time bound to each invariant and guard in order to ensure that it is not violated.

Table 3. Graph analysis data for the Firewire model.

	$\mathbb{P}_{\geq 1}(\Diamond\, done)$				$\mathbb{P}_{\geq \lambda}(\Diamond\, done \wedge z \leq 10000)$		
		iterations			iterations		
c	time	$MaxV_{\geq 1}$	$MaxU_{\geq 1}$	time	$MaxV_{\geq 1}$	$MaxU_{\geq 1}$
10	0.7	372	780	0.2	372	780
100	0.1	39	82	0.0	39	82
360	0.0	13	27	0.0	13	27
1670	0.0	5	11	0.0	5	11
2000	0.0	4	9	0.0	4	9
3000	0.0	4	8	0.0	4	8
			...			
10000	0.0	3	6	0.0	3	6

We also compare to digital clocks using the Firewire model in Table 4 for property $z.\mathbb{P}_{\sim\lambda}(\Diamond\, done \wedge z \leq D)$ as in [11, Table 4]. We fix c to 4000. We observe a similar pattern to the one in Table 2, where backwards reachability is faster than digital clocks for lower time bounds. Again digital clocks scales better with higher time bounds in both runtime and state-space. We also observe a larger state space once again of up to three times the number of states in [11].

6 Conclusion

We attempted to replicate the work on backwards reachability for PTA of [11] via an implementation in the Modest Toolset that closely follows the original pseudocode. Our replication was partially successful: The implementation works and computes the same probabilities as the digital clocks approach where applicable. However, we see rather different patterns for runtime and iteration counts in several cases, especially when varying c for the CSMA/CD model.

Table 4. State space sizes and model checking times for the Firewire model.

	backwards reachability				digital clocks		
D	states (max)	time (max)	states (min)	time (min)	states	time (max)	time (min)
2000	50	0.0	35	0.0	7670	0.5	0.5
4000	125	0.0	83	0.0	7670	0.5	1.1
8000	455	0.0	256	0.0	7670	0.5	2.1
10000	725	0.0	379	0.0	7670	0.5	2.6
20000	2655	2.0	1373	0.4	7670	0.5	5.1
40000	10300	192.9	5152	21.9	7670	0.5	7.8

References

1. Alur, R., Dill, D.L.: A theory of timed automata. Theor. Comput. Sci. **126**(2), 183–235 (1994). doi.org/10.1016/0304-3975(94)90010-8
2. Dill, D.L.: Timing assumptions and verification of finite-state concurrent systems. In: International Workshop on Automatic Verification Methods for Finite State Systems. Lecture Notes in Computer Science, vol. 407, pp. 197–212. Springer (1989). doi.org/10.1007/3-540-52148-8_17
3. Hahn, E.M., Hartmanns, A.: A comparison of time- and reward-bounded probabilistic model checking techniques. In: Second International Symposium on Dependable Software Engineering: Theories, Tools, and Applications, (SETTA). Lecture Notes in Computer Science, vol. 9984, pp. 85–100 (2016). doi.org/10.1007/978-3-319-47677-3_6
4. Hartmanns, A., Hermanns, H.: The Modest Toolset: An integrated environment for quantitative modelling and verification. In: 20th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS). vol. 8413, pp. 593–598. Springer (2014). doi.org/10.1007/978-3-642-54862-8_51
5. Hartmanns, A., Katoen, J.P., Kohlen, B., Spel, J.: Tweaking the odds in probabilistic timed automata. In: 18th International Conference on Quantitative Evaluation of Systems (QEST). Lecture Notes in Computer Science, vol. 12846, pp. 39–58. Springer (2021). doi.org/10.1007/978-3-030-85172-9_3
6. Kwiatkowska, M.Z., Norman, G., Parker, D.: Stochastic games for verification of probabilistic timed automata. In: 7th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS). Lecture Notes in Computer Science, vol. 5813, pp. 212–227. Springer (2009). doi.org/10.1007/978-3-642-04368-0_17
7. Kwiatkowska, M.Z., Norman, G., Parker, D.: PRISM 4.0: Verification of probabilistic real-time systems. In: 23rd International Conference on Computer Aided Verification (CAV). Lecture Notes in Computer Science, vol. 6806, pp. 585–591. Springer (2011). doi.org/10.1007/978-3-642-22110-1_47
8. Kwiatkowska, M.Z., Norman, G., Parker, D., Sproston, J.: Performance analysis of probabilistic timed automata using digital clocks. Formal Methods Syst. Des. **29**(1), 33–78 (2006). doi.org/10.1007/s10703-006-0005-2
9. Kwiatkowska, M.Z., Norman, G., Segala, R., Sproston, J.: Automatic verification of real-time systems with discrete probability distributions. Theor. Comput. Sci. **282**(1), 101–150 (2002). doi.org/10.1016/S0304-3975(01)00046-9
10. Kwiatkowska, M.Z., Norman, G., Sproston, J., Wang, F.: Symbolic model checking for probabilistic timed automata. In: Joint International Conferences on Formal Modelling and Analysis of Timed Systems (FORMATS) and Formal Techniques in Real-Time and Fault-Tolerant Systems (FTRTFT). Lecture Notes in Computer Science, vol. 3253, pp. 293–308. Springer (2004). doi.org/10.1007/978-3-540-30206-3_21
11. Kwiatkowska, M.Z., Norman, G., Sproston, J., Wang, F.: Symbolic model checking for probabilistic timed automata. Inf. Comput. **205**(7), 1027–1077 (2007). doi.org/10.1016/j.ic.2007.01.004
12. Puterman, M.L.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. Wiley Series in Probability and Statistics, Wiley (1994). doi.org/10.1002/9780470316887