Indonesian Journal of Electrical Engineering and Computer Science
Vol. 18, No. 1, April 2020, pp. 1571~1577
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v18.i1.pp1571-1577

Improving graph-based methods for computing qualitative properties of markov decision processes

Mohammadsadegh Mohagheghi[1], Khayyam Salehi[2]
[1]Department of Computer Science, Vali-e-Asr University of Rafsanjan, Iran
[2]Department of Computer Science, University of Tabriz, Iran

Article Info

Article history:

Received Aug 1, 2019
Revised Sep 21, 2019
Accepted Oct 11, 2019

Keywords:

Graph-based pre-computation
Markov decision processes
Probabilistic model checking
Qualitative reachability probabilities

ABSTRACT

Probabilistic model checking is a formal verification method, which is used to guarantee the correctness of the computer systems with stochastic behaviors. Reachability probabilities are the main class of properties that are proposed in probabilistic model checking. Some graph-based pre-computation can determine those states for which the reachability probability is exactly zero or one. Iterative numerical methods are used to compute the reachability probabilities for the remaining states. In this paper, we focus on the graph-based pre-computations and propose a heuristic to improve the performance of these pre-computations. The proposed heuristic approximates the set of states that are computed in the standard pre-computation methods. The experiments show that the proposed heuristic can compute a main part of the expected states, while reduces the running time by several orders of magnitude.

Corresponding Author:

Mohammadsadegh Mohagheghi,
Department of Computer science,
Vali-e-Asr University of Rafsanjan, Iran.
Email: mohagheghi@vru.ac.ir

1. INTRODUCTION

Formal methods are mathematical-based approaches that are used in software engineering and hardware design. The goal of these methods is to guarantee the correctness of the qualitative or quantitative properties of the desired systems [1, 2]. Model checking is an automated formal method that uses graph-based structures for modelling the underlying systems and logic-based propositions to specify the system properties [1, 3]. A model checker is a software tool that decides the satisfiability of the specified properties against the proposed model [1, 4].

Because of the stochastic behaviors of many computer systems, probabilistic structures are more useful for modeling such systems [4-6]. Markov chains and Markov decision processes (MDPs) are well-known structures for modelling stochastic systems and are widely used in artificial intelligence, economy, operations research and software engineering [7-9]. Several examples of the stochastic systems and their modelling are available in [3, 4, 7]. For this class of systems, probabilistic model checking is a good technique to verify the quantitative or qualitative properties of the systems. PRSIM [10], STORM [11] and IscasMC [12] are state of the art probabilistic model checkers.

The main classes of properties that are used in probabilistic model checking contain reachability probabilities, i.e. the maximal or minimal probability of reaching a set of goal states in the MDP model. A standard approach for computing reachability probabilities is to use iterative graph-based and numerical methods [13]. Graph-based methods (also called pre-computation) compute the set of states for which the probability of reaching a goal state is exactly zero or one [4]. An iterative numerical method starts from an initial vector of values and iteratively updates the values until satisfying the stopping criterion [1].

The main challenge of model checking in all variants is the state explosion problem, i.e. the number of the states of the model grows exponentially in the number of its components. Several techniques are proposed to cope with this problem [1, 14-17]. In the case of probabilistic model checking the running time of iterative methods is the main problem that limits the scalability of the method [3, 18, 19]. Several techniques have been proposed to improve the performance of the standard probabilistic model checking methods. These techniques are used to reduce the running time of the graph-based [20, 21] or numerical computations [22-24] or both methods [16, 25]. Although the proposed methods show promising results in the running time of probabilistic model checking, experimental results show more improvement is needed for iterative graph-based of numerical methods [26]. In this paper, we consider the running time of graph-based methods as an important problem in probabilistic model checking and propose a new approach to reduce this running time.

The forward and backward approaches are two alternatives for graph-based computations in the probabilistic model checking of MDPs [1]. The forward approach is normally faster than the backward approach, but its main drawback is the memory overhead, which limits its scalability to relatively small models [1, 20]. As a result, most prominent model checkers (such as PRISM and IscasMC) use the forward approach for the pre-computations to avoid memory overhead, which is essential to overcome the state explosion problem [15, 27]. The motivation of this paper is to improve the performance of the graph-based pre-computation methods. To avoid memory overhead, we focus on the forward approach for pre-computation. As the main contribution of the paper, we propose a heuristic to reduce the running time of pre-computation in the forward approach. This heuristic reduces the number of iterations by approximating the set of expected states in an improved order. Although this heuristic does not guarantee to detect all states that the standard pre-computation algorithms do, the experiments show that it can reduce the overall run-time in most cases and improves the performance of probabilistic model checking of MDPs.

2. FORMAL PROBLEM DEFINATION AND BACKGROUND

In this section, we propose a brief review of the main concepts of probabilistic model checking that are used in this paper. More details about probabilistic model checking and their methods are available in [1, 3, 4].

2.1. Probabilistic Models and Reachability Probabilities

Markov decision processes are used in probabilistic model checking to model both nondeterministic and probabilistic aspects of a system [7].

Definition 1. (Markov Decision Process) A Markov Decision Process (MDP) is defined as a tuple $M = (S, \bar{s}, Act, \rightarrow, G)$ where S is a finite set of states, $\bar{s} \in S$ is the initial state, Act is a finite set of actions. For every state $s \in S$ the set of enabled actions of s is denoted by $Act(s)$. We use $|Act(s)|$ as the size of this set and $\rightarrow$ is a probabilistic transition function and is defined as a subset of $S \times Act \times Dist(S)$. For each state $s \in S$ and enabled action $\alpha \in Act(s)$, exactly one transition $s \xrightarrow{\alpha} P$ exists. The notation $s \xrightarrow{\alpha} P$ means that (s, α, P) is an element of $\rightarrow$. We use $P(s, \alpha, s')$ to denote the probability of reaching from s to s'. By the action α. $G \subset S$ is the set of goal states. MDPs are widely used to model decision making problems in stochastic environments. Transitions of an MDP M show the behavior of the related system [1]. For a state $s \in S$, a transition is performed in two steps: First, an action $\alpha \in Act(s)$ is selected non-deterministically. Next, the destination state s' is selected randomly with probability $P(s, \alpha, s')$. We use $Post(s)$, $Pre(s)$, $Post(s, \alpha)$ for the set of successor and predecessor and α-successor states of s:

$$Post(s) = \{s' \in S | \exists \alpha \in Act(s), P(s, \alpha, s') > 0\} \quad (1)$$

$$Pre(s) = \{s' \in S | s \in post(s')\} \quad (2)$$

$$Post(s, \alpha) = \{s' \in S | P(s, \alpha, s') > 0\} \quad (3)$$

We use $|M|$ for the size of M and define it as the number of states and transitions of M. A finite path of M is a possible sequence of actions and transitions of M and is defined as $\pi = s_0 \xrightarrow{\alpha_0} s_1 \xrightarrow{\alpha_1} \ldots \xrightarrow{\alpha_{n-1}} s_n$ where $s_i \in S$, $\alpha_i \in Act(s_i)$ and $s_{i+1} \in Post(s_i, \alpha_i)$ for all $0 \leq i < n$. We use $last(\pi)$ for the last state of π.

Reachability probabilities are one of the main properties of probabilistic models that are computed in probabilistic model checking [3, 4]. A reachability probability is defined as the probability of reaching a goal state from the initial state of the model. In the case of MDPs, reachability probabilities are defined as the optimal (minimal or maximal) probability of reaching one of the goal states from the initial state $\bar{s}$. We use

$pr_s^{min}(reach(G))$ and $pr_s^{max}(reach(G))$ for minimal and maximal reachability probabilities of M. Iterative numerical approaches (such as value iteration [18] and policy iteration [4]) can be used to approximate the reachability probabilities.

2.2. Qualitative Reachability

Some graph-based computations can determine the set of states for which the extremal reachability probabilities are exactly 0 or 1. In the case of maximum reachability probability, these sets are denoted by S^0_{max} and S^1_{max} and are defined as:

$$S^0_{max} = \{s \in S | Pr_s^{max}(reach(G)) = 0\} \quad (4)$$

$$S^1_{max} = \{s \in S | Pr_s^{max}(reach(G)) = 1\} \quad (5)$$

We define $S^?_{max} = S \backslash (S^0_{max} \cup S^1_{max})$ for the remaining states. For the case of minimum reachability probabilities, we use S^0_{min}, S^1_{min} and $S^?_{min}$ and define them in an analogous way [4]. The computations of these sets are called qualitative reachability analysis and are used as pre-computation in probabilistic mode checking. The main aims of pre-computations are to simplify the iterative computations by focusing on $S^?_{max}$ and to increase the precision of computations [4, 18, 20, 25]. Algorithm 1 and Algorithm 2 show the standard pre-computation methods for the S^0_{max} and S^1_{max} sets [4].

Algorithm 1 Pre-Computation for S^0_{max}.

input: an MDP $M = (S, \hat{s}, Act, \rightarrow, G)$
output: The set $S^0_{max} = \{s \in S | Pr_s^{max}(reach(G)) = 0\}$
1. $R := G$;
2. do
3. $R' := R$;
4. $R := R' \cup \{s \in S | Post(s) \in R'\}$;
5. while $R \neq R'$;
6. return $S \backslash R$;

Algorithm 2 Pre-Computation for S^1_{max}.

input: an MDP $M = (S, \hat{s}, Act, \rightarrow, G)$
output: The set $S^1_{max} = \{s \in S | Pr_s^{max}(reach(G)) = 1\}$
1. $R := G$;
2. do
3. $R' := R$;
4. $R := G$;
5. do
6. $R'' := R$;
7. $R := R'' \cup \{s \in S | \exists \alpha \in Act(s).(\forall s' \in S.(P(s, \alpha, s') > 0 \rightarrow s' \in R')) \wedge (Post(s, \alpha) \in R'')\}$;
8. while $R \neq R''$;
9. while $R \neq R'$;
10. return R;

Algorithm 1 iteratively computes the set R of states that can reach to one of the goal states G. In each iteration, the algorithm adds a state $s \in S$ to R if at least one state $s' \in Post(s)$ has been added to R in the previous iteration. For any state $s \in R$, we have $pr_s^{min}(reach(G)) > 0$. The remaining states cannot reach to any of the goal states and are returned as the S^0_{max} set.

Algorithm 2 uses a nested while loop to compute the S^1_{max} set. The outer loop starts from S and successively removes those states $s \in S$ for which we are sure $pr_s^{min}(reach(G)) < 1$. It induces a sequence of R_i sets, where $S = R_0 \supset R_1 \ldots \supset R_i = S^1_{max}$. To compute the R_i sets, the inner loop starts from G (line 6 of Algorithm 2) and iteratively adds each state $s' \in S$ to R_i if s' can reach to one of the goal states with probability one via the states of R_{i-1}. For the remaining states (the states in $S \backslash R_i$), we are sure that they do not belong to S^1_{max}. The proof of correctness of these algorithms is available in [18].

3. RESEARCH METHOD

The time complexity of Algorithm 1 is in $O(|M|)$ and the time complexity of Algorithm 2 is in $O(|M|^2)$. In these cases, we suppose that for each state $s \in R'$, the algorithms can determine any states of

$Pre(s)$ in $O(1)$ [1]. For this purpose, the method should restore the information of the model in the backward approach, i.e. for each state $s \in S$, the list of states in $Pre(s)$ should be restored. The main drawback of restoring the information of a model is its memory overhead which is a main challenge in the state explosion problem. On the other hand, the forward implementation of these algorithms (as is used in PRISM [10]) need not any additional memory, but may increase the running time of the computations. To improve the performance of the pre-computations in the forward manner, we propose a heuristic to reduce their running time. As the first part of the heuristic, we modify Algorithm 1 for the computation of the S^1_{max} set. As the second part (the main contribution of the work), we propose a new approach to approximate the S^1_{max} set.

3.1. Improving Method for Computing S^0_{max}

The idea of Algorithm 1 (which is used in PRISM and IscasMC) is to add a state $s \in S$ to R if at least one states in $Post(s)$ has been added in the previous iteration. To reduce the number of iterations of this algorithm in the forward approach, it can use only one set (R) to store the set of states that can reach to G. In each iteration k, a state $s \in S$ is added to R if at least one state in $Post(s)$ has been added to R in the iteration k or $k-1$. Note that in Algorithm 1, s is added to R only if one state in $Post(s)$ has been added to R in the previous iteration and not in the current one.

3.2. Improving Method for Approximating S^1_{max}

In this section we propose our heuristic to compute a set $T \subseteq S^1_{max}$as an approximation of the S^1_{max} set. This heuristic starts from $T_1 = G$ and iteratively compute the sets $T_1 \subset T_2 \subset \cdots \subset T_{n-1} \subset T_n \subseteq S^1_{max}$ where T_n is the fixed point set of the computations. For each iteration i of the heuristic, a state $s \in S$ should be added to T_i if there exists an action $\alpha \in Act(s)$ for which $Post(s,\alpha) \subseteq T_{i-1}$. In this case, all α-successor states of s are in T_{i-1}, which means for all of these states, we are sure that their maximum reachability probability is one. This heuristic is explained in Algorithm 3.

Algorithm 3 ApproximatingS^1_{max}.

Input: An MDP $M = (S, \bar{s}, Act, \rightarrow, G)$
Output: The set $S^1_{max} = \{s \in S | Pr_s^{max}(reach(G)) = 0\}$
1. $T \coloneqq G$;
2. do
3. $R \coloneqq T$;
4. $T \coloneqq T \cup \{s \in S | \exists \alpha \in Act(s).(\forall s' \in S.(P(s,\alpha,s') > 0 \rightarrow s' \in R))\}$;
5. while $R \neq T$;
6. return T;

The running time of Algorithm 3 is $n \times |M|$ where n the number of iterations of the algorithm is and $|M|$ is the size of the model. In worst case, $n = |S| - 1$ and the worst case time complexity of the algorithm is in $O(|S| \times |M|)$. However, in most cases, the algorithm terminates after a few numbers of iterations. Note that the proposed heuristics in this section (Subsections 3.1 and 3.2) do not need any additional memory.

4. RESULTS AND ANALYSIS

To compare the running time of our heuristic for pre-computation with the standard method, we have implemented them in PRISM. We use 5 classes of standard case studies which are used in several previous works [4, 15, 19, 20, 26, 28]. For each class, we consider 4 models. More details about these case studies are available in [10]. In Table 1 we propose the name, parameter values, the number of states of each model and the experimental results. We use the sparse engine of PRISM for running the standard and improved pre-computation methods. We propose the number of states in the S^1_{max} and S^0_{max} sets and the approximation of S^1_{max} (columns $|S^0_{max}|$ and $|S^1_{max}|$ in Table 1. We also propose the running times of standard pre-computations and our proposed methods. All times are in seconds.

For most cases (except the coin ones) our heuristic finds all states of the S^1_{max} set. On the other hand, our heuristic reduces the running time of pre-computations for all cases. The best result is for CSMA (n=3,k=6) which the running time of the standard pre-computation method is more than 8 hours (24603 seconds), while the running time of our improved method is less than 30 seconds. For other cases of CSMA and most cases of zeroconf, wlan and wlan collide our methods reduce the running times of pre-computation by two orders of magnitude. These results show that our heuristic presents a significant improvement in the performance of graph-based iterative methods in probabilistic model checking.

Table 1. Running Time of Pre-Computation Methods for MDP Models

Model	Parameter(s)	$\|S\|$	$\|S^1_{max}\|$	$\|S^0_{max}\|$	Time	$\|S^1_{max}\|$	Time
Coin	n=2,K=45	5,776	12	30	3.24	12	0.01
	n=4,K=12	32,056	6,156	294	10.96	324	0.03
	n=4,K=20	53,048	9,996	294	29.2	324	0.08
	n=6,K=9	68,914	18,202	694	28.48	988	0.07
Zeroconf	K=8	1,870,338	171,749	611,330	77.75	171,749	0.66
	K=10	3,001,911	197,004	957,807	94.6	197,004	1.16
	K=12	3,754,386	189,372	1,082,145	113.5	189,372	1.46
	K=14	4,427,159	171,851	1,160,964	141.1	171,851	1.8
CSMA	n=3,k=4	249,678	118,544	7,726	22.72	118,544	0.24
	n=3,k=6	14,222,529	10,120,379	169,206	24603	10,120,379	25.26
	n=4,k=2	39,481	6,312	1,972	5.7	6,312	0.05
	n=4,k=4	5,874,853	514,457	171,960	1324.9	514,457	22.12
Wlan	TTM=1500,n=5	3,634,518	2,734,164	847,967	60.98	2,734,164	1.51
	TTM=3000,n=5	5,989,518	4,300,164	1,594,967	125.81	4,300,164	2.63
	TTM=250,n=6	5,755,628	5,083,436	641,581	45.7	5,083,436	3.56
	TTM=450,n=6	6,379,028	5,497,036	844,381	60.62	5,497,036	4.13
Wlan_collide	TTM=1000,n=5	2,851,619	2,212,165	601,067	65.95	2,212,165	1.04
	TTM=2500,n=5	5,209,619	3,778,165	1,351,067	140.36	3,778,165	2.5
	TTM=200,n=6	5,600,280	4,980,037	591,382	54.85	4,980,037	3.26
	TTM=400,n=6	6,224,080	5,393,637	794,582	84.99	5,393,637	3.89

To compare the overall running time of probabilistic model checking, we consider the running times of graph-based computations of the S^1_{max} and S^0_{max} sets and the running time of the iterative numerical methods. We select one sample model from each class of case studies and propose the results in Figures 1 to 3. Each figure presents the running time of the standard and improved methods for computing the S^1_{max} and S^0_{max} sets. We also use the SCC-based topological and the learning-based methods as two well-known improved iterative graph-based and numerical methods [16, 19, 20, 23, 25]. These methods are now available in the explicit engine of PRISM. For the Coin class, we select two sample models to study the impact of our heuristic on the overall running time of the probabilistic model checking. The results of these figures show that for all cases, the running time of probabilistic model checking with our heuristic is less than the running time of the other methods. The results show that the running times are reduced for the computations of both S^1_{max} and S^0_{max} sets. In most cases, the total running time of probabilistic model checking is reduced to less than 50% of the running time of the best previous method, which shows a significant improvement in our proposed heuristic. For CSMA(n=3,k=6) and CSMA(n=4,k=4), which is not presented here, the total running time is reduced to less than 1% of the running time of the best previous method.

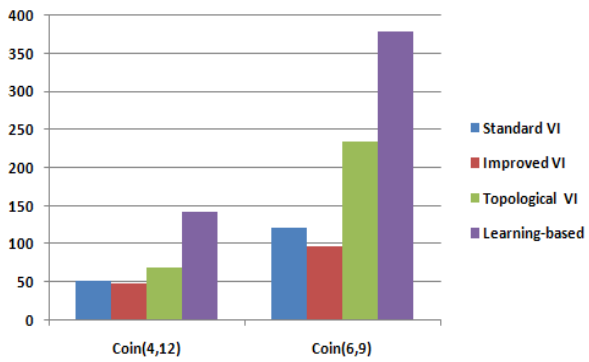

Figure 1. Running times of probabilistic model checking for Coin sample models

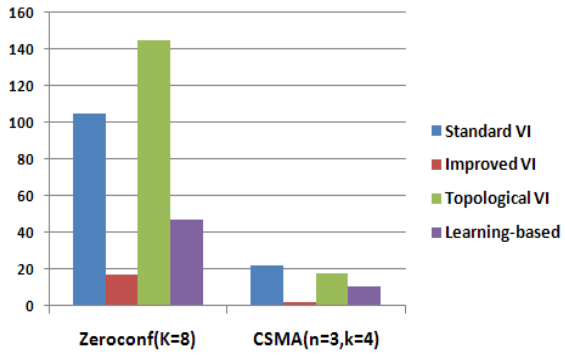

Figure 2. Running times of probabilistic model checking for *Zeroconf* and *CSMA* sample models

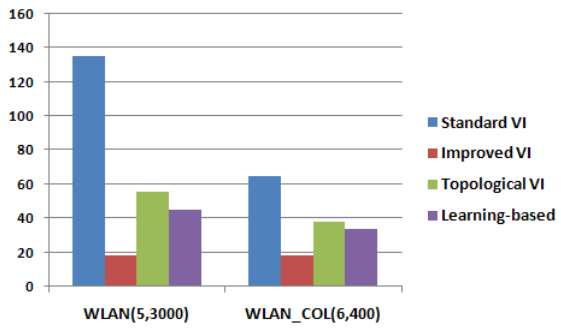

Figure 3. Running times of probabilistic model checking for wlan and wlan_col models

5. CONCLUSION

The paper proposed a heuristic to reduce the running time of pre-computation for probabilistic model checking of MDPs. The idea of the proposed methods in subsection 3.1 and 3.2 is to reduce the number of iterations of the graph-based pre-computations. Experimental results show that the proposed heuristic reduces the running times by several orders of magnitude and outperforms the standard and previous improved methods for probabilistic model checking. For future works, other probabilistic structures (such as probabilistic timed automata or stochastic hybrid automata) can be considered and one can study the impact of the proposed heuristic on the performance of model checking methods for these structures.

REFERENCES

[1] Christel Baier and Joost-Pieter Katoen. ”Principles of model checking”. *MIT press*, 2008.
[2] Mungkasi, Sudi. ”Formal expansion method for solving an electrical circuit model.” *TELKOMNIKA (Telecommunication, Computing, Electronics and Control)*, vol. 17(3). pp. 1338-1343, 2019.
[3] Christel Baier, et al.,”Probabilistic Model Checking”. *Dependable Software Systems Engineering*. 2016, vol. 45, pp: 1-23.
[4] V. Forejt et al.,”Automated Verification Techniques for Probabilistic Systems”. In *SFM* 2011, Vol. 11, pp. 53-113.
[5] Alostad, Jasem M.”Improved probabilistic distance based locality preserving projections method to reduce dimensionality in large datasets.” *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9(1), pp. 593-599, 2019.
[6] EzatulAkma Abdullah, SitiMeriamZahari, S.SarifahRadiahShariff, Muhammad Asmui Abdul Rahim. "Modelling volatility of Kuala Lumpur composite index (KLCI) using SV and garch models”. *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*. Vol. 13(3), 2019.
[7] ML. Puterman”Markov decision processes: discrete stochastic dynamic programming”, *John Wiley & Sons*; 2014 Aug 28.
[8] Lalaoui, Mohamed, Abdellatif El Afia, and RaddouaneChiheb. ”A self-tuned simulated annealing algorithm using hidden markov model.” *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 8(1), pp. 291-298, 2018.
[9] Marbun, Musa Partahi, NgapuliIrmeaSinisuka, and NanangHariyanto. ”The use of Markov Chain method to determine spare transformer number and location.”*International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9(1), pp. 1-8, 2019.
[10] Kwiatkowska M, Norman G, and Parker D. “*PRISM 4.0: Verification of probabilistic real-time systems*”. InInternational conference on computer aided verification 2011 Jul 14 (pp. 585-591). Springer, Berlin, Heidelberg.
[11] Dehnert C, et al., “*A storm is coming: A modern probabilistic model checker*”. InInternational Conference on Computer Aided Verification 2017 Jul 24 (pp. 592-600). Springer, Cham.
[12] Hahn, Ernst Moritz, Yi Li, Sven Schewe, Andrea Turrini, and Lijun Zhang. ”ISCASMC: A Webbased Probabilistic Model Checker.” *In International Symposium on Formal Methods*, pp. 312-317. Springer, Cham, 2014.
[13] G. Norman and D. Parker. Quantitative verification:”Formal guarantees for timeliness, reliability and performance”. *Technical report, The London Mathematical Society and the Smith Institute*, 2014.
[14] Hartmanns. ”On the analysis of stochastic timed systems”. PhD thesis, Saarland University, 2015.
[15] Joachim Klein, et al.,”Advances in probabilistic model checking with PRISM: variable reordering, quantiles and weak deterministic Bchi automata”. *International Journal on Software Tools for Technology Transfer*, vol. 20(2), pages 179-194. 2018.
[16] Mateusz Ujma. ”On Verification and Controller Synthesis for Probabilistic Systems at Runtime” PhD thesis, University of Oxford, 2015.
[17] Guldstrand Larsen. ”Statistical Model Checking the 2018 Edition!” In: Margaria T., Steffen B. (Eds) Leveraging Applications of Formal Methods, Verification and Validation. Verification. ISoLA 2018. Lecture Notes in Computer Science, vol 11245. *Springer*, Cham.

[18] Luca De Alfaro. ”Formal verification of probabilistic systems”. PhD thesis, Stanford University, 1997.
[19] Christel Baier, et al.,”*Ensuring the reliability of your model checker: Interval iteration for Markov Decision Processes*.” In International Conference on Computer Aided Verification, pp. 160-180. Springer, Cham, 2017.
[20] M, Kwiatkowska et al.,”*Incremental quantitative verification for Markov decision processes*”. In Dependable Systems & Networks (DSN), 2011 IEEE/IFIP 41st International Conference on 2011 Jun 27 (pp. 359-370). IEEE.
[21] Chatterjee K, Lacki J. “*Faster algorithms for Markov decision processes with low treewidth*”. In International Conference on Computer Aided Verification 2013 Jul 13 (pp. 543-558). Springer, Berlin, Heidelberg.
[22] Haddad, Serge, and Benjamin Monmege. ”Interval iteration algorithm for MDPs and IMDPs.” Theoretical *Computer Science*, 735 (2018): 111-131.
[23] Klein, Joachim, ChristelBaier, Philipp Chrszon, Marcus Daum, Clemens Dubslaff, SaschaKlppelholz, Steffen M? Rcker, and David Mller. ”Advances in probabilistic model checking with PRISM: variable reordering, quantiles and weak deterministic Bchi automata.” *International Journal on Software Tools for Technology Transfer*, vol. 20(2), pp. 179-194, 2018.
[24] Mateescu, Radu, and Jos Ignacio Requeno. ”On-the-fly model checking for extended actionbased probabilistic operators.” *International Journal on Software Tools for Technology Transfer*. Vol 20(5), pp. 563-587, 2018.
[25] Wang, Jingyi, Jun Sun, Qixia Yuan, and Jun Pang. ”Learning probabilistic models for model checking: an evolutionary approach and an empirical study.” *International Journal on Software Tools for Technology Transfer*. Vol 20(6), pp. 689-704, 2018.
[26] Ernst Moritz, et. al.”The 2019 comparison of tools for the analysis of quantitative formal models.” In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pp. 69-92. Springer, Cham, 2019.
[27] Von Essen, Christian, Barbara Jobstmann, David Parker, and Rahul Varshneya. ”Synthesizing efficient systems in probabilistic environments.” *ActaInformatica*, vol 53(4), pp. 425-457, 2016.
[28] Mohammadsadegh Mohagheghi, Jaber Karimpour, Ayaz Isazadeh, Prioritizing Methods to Accelerate Probabilistic Model Checking of Discrete-Time Markov Models, *The Computer Journal*, 2019.

BIOGRAPHIES OF AUTHORS

Mohammadsadegh Mohagheghi received his Ph.D in computer science from University of Tabriz in 2019, Ms in computer science from Sharif university of technology in 2008 and Bs in software engineering from Shahidbeheshty university in 2006. He is cuurently a faculty member of computer science in Vali-e-asr university of Rafsanjan, Iran. His main research interests include formal verification of stochastic and real-time systems, probabilistic model checking and machine learning.

Khayyam Salehi received his M.Sc. degree in Computer Science from Sharif University of Technology in 2010 and his Ph.D. in Computer Science at University of Tabriz in 2019. His research interests include quantification of information leakage, formal methods, and AI in computer security.