Runtime Probabilistic Model Checking Based on Incremental Method

Chao He

Information Engineering, Nanjing University of Finance and Economics, Nanjing 210000, China
704218038@qq.com

ABSTRACT. *Nowadays, more and more systems change dynamically during their life cycle, runtime probabilistic model checking is proposed to verify these system. An important challenge of runtime probabilistic model checking is its performance. It should be fast enough to respond to runtime requirements and continuously verify whether the current system meets system requirements when the system changes dynamically. In this paper, in view of the efficiency of the runtime probabilistic model checking, we propose a runtime probabilistic model checking based on incremental method. The method applies the ideal of incremental verification to reuse the calculated value of the previous model to reduce the number of iterations and improve their performance. We implement our method in model checking tool PRISM, and use a benchmark case model to perform model verification on its reachability properties. The results of the experiments show that the method proposed in this paper can reduce the system verification time of standard runtime probabilistic model checking by more than 45% in most of cases.*

KEYWORDS: *Runtime probabilistic model checking, Incremental verification, Stochastic system, Discrete-Time Markov Chain*

1. Introduction

Computer systems are widely used in our lives, our communication, finance, transportation, and aerospace fields are inseparable from computer systems. The widespread use of these systems, combined with their increasing complexity, means that effective methods to ensure their reliability and performance are essential. Model checking[1] is an automated formal verification approach that is used to verify computer systems. In this approach, labeled transition systems are usually used to model systems and temporal logics are used to specify system's properties. Some computer systems have stochastic behaviors. Therefore, we use probabilistic model checking[2] to analyze their quantitative properties. In this domain, the Discrete-time Markov Chain (DTMC) is used to model systems with random behavior Markov. Probabilistic Computation Tree Logic (PCTL) is used to specify system's properties A main class of PCTL properties is the optimal (maximum or minimum) reachability probabilities. In most cases, numerical computations are used to calculate these probabilities [3].

Many computer systems will encounter dynamic changes in their life cycle, these changes usually occur in the structure and components of the computer system [4], and lead to the addition, deletion or modification of its components, which requires us verify the system at runtime. The runtime probabilistic model checking [5] is to verify whether the system with random behavior meets the system's properties at runtime. Figure 1 describes the process of the runtime probabilistic model checking. With the monitoring of the system [6], the system models are different at different times. The model checker needs to continuously verify the newly generated models to check whether the current model meets the system's properties. This continued verification at runtime requires a lot of cost.

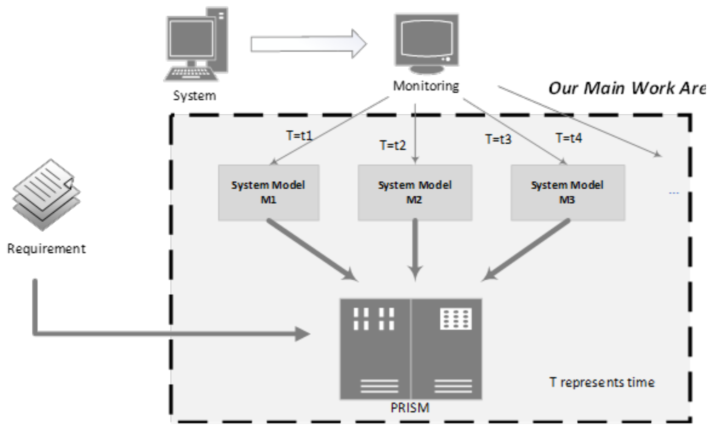

Figure. 1 Runtime probabilistic model checking with PRISM[7]

In order to improve the efficiency [8] of runtime probabilistic model checking, we used the idea of incremental verification [10] [11] to improve its verification efficiency. This paper focuses on the research on the reachability probability [9] of DTMC model. We change the probability transition value between states in the model to simulate the changes in the system at runtime, and finally verify the effectiveness of the proposed method through a benchmark case model in PRISM. The main contributions of this article are as follows:

1. Analyze the principles of the existing runtime probabilistic model checking, and summarize the limitations and performance bottlenecks of some verification methods.

2. Apply the ideal of incremental verification to reuse calculation results of the initial model probabilistic values as the initial values of iterative calculation for the changed model.

After this section, we review the work related to incremental method in Section 2. In Section 3, we introduce the basic knowledge and theory of probabilistic model checking. In Section 4, we describe the runtime probabilistic model checking based on incremental method and the implementation of incremental method. In Section 5, we selected two types of DTMC models in the PRISM benchmark case to do the experiments and analyze the experimental results. The last section summarizes this paper.

2. Related work

Research related to incremental methods, for non-probabilistic systems, an incremental algorithm is proposed in model checking for the first time in [12]. The basis of the algorithm is using the changes $\triangle$ of LTS as input for incremental model verification. These changes include the addition and deletion of states. The experimental results show that in the worst case, the time required for the incremental model checking algorithm is linearly related to the size of the LTS. In the best case, the time spent is linearly related to the amount of change $\triangle$. In [13] and [14], the incremental technique is used to accelerate the generation of state space or the inspection of functional attributes, but neither quantitative attributes nor numerical calculations are considered in these works. In the related research on probabilistic systems, [15] researched incremental model construction when the number of system components increases. In order to reduce the running time of the probabilistic model checking, an incremental reduction technology based on strong connected components (SCC) was proposed in [10]. The SCC-based method [16] identifies the strongly connected components of the underlying model and calculates each in sequence. When the model state changes, only needs to recalculate the reachability probability values of the corresponding SCC. In this paper,.we focus on the reachability probability of DTMC model. we propose a runtime probabilistic model checking based on incremental method, which applies the ideal of incremental verification to reuse the calculated value of the previous model to reduce the number of iterations and improve their performance.

3. Background

3.1 Discrete time markov chains

Discrete time Markov chains are discrete stochastic processes with Markov property, according to this process, the probability distribution of future states depend only on the current state. They are defined as Kraske structures with probabilistic transitions between states. The states represent the possible configuration of the system. The transitions between states occur at discrete times and have related probabilities. DTMC is currently widely used to model the reliability of systems with different components (services). In particular, they prove to be useful for early assessment or prediction of reliability.

A DTMC is defined as $M = (S, P, s_0, AP, L)$ where

- S is a finite set of states.
- $P: S \times S \to [0,1]$, $\forall s \in S$, $\Sigma_{s' \in S} P(s,s') = 1$, $P(s,s')$ represents the probability that the next state of process will be s' given that the current state is s.
- s_0 is a set of initial states.
- AP is a set of atomic propositions.
- $L: S \to 2^{AP}$ is a labeling function which assigns to each state the set of atomic propositions which are true in that state.

A state $s \in S$ is called the absorbing state if $P(s,s) = 1$. If a DTMC contains at least one absorbing state, itself is called absorbing DTMC. In the simple model for reliability analysis, DTMC will have two absorbing states, representing the correct completion of the task and the failure of the task. The use of multiple absorbing states can often be extended to model's different fault conditions. For example, different fault states may be associated with different external service calls.

3.2. Probabilistic computation tree logic

PCTL is a probabilistic expansion of the Computation Tree Logic (CTL). The probabilistic operator $P_{\sim p}$ quantitatively expands the CTL. PCTL can describe the quantitative branch time property of DTMC and MDP. The PCTL formula can be defined as:

$$\Phi ::= true | a | \Phi \wedge \Phi | \neg\Phi | P_{\sim p}(\Psi)$$

$$\Psi ::= \mathrm{X}\Phi | \Phi \cup \Phi | \Phi \cup^{\leq n} \Phi$$

PCTL provides the probabilistic operator $P_{\sim p}$, where $p \in [0,1]$ is a probability bound and $\sim \in \{\leq, <, \geq, >\}$. The formula Φ is called a state formula, which can be evaluated as true or false in each state, and the formula Ψ is called a path formula, and its authenticity will be evaluated for each execution path. X stands for *next*, $\cup$ stands for *until*, and their semantics are the same as those of CTL path operators. $\cup^{\leq n}$ $(bounded\ until)$ is a variant of $\cup$, which means that n migrations or less than n migrations satisfy $\cup$ semantics, where n is a non-negative integer. The PCTL formula for each state can be defined as:

$$s \vDash true$$
$$s \vDash a \ \Leftrightarrow \ a \in L(s)$$
$$s \vDash \Phi_1 \wedge \Phi_2 \ \Leftrightarrow \ s \vDash \Phi_1 \ and \ s \vDash \Phi_2$$
$$s \vDash \neg\Phi \ \Leftrightarrow \ s \nvDash \Phi$$

Each path ω can be defined as:

$$\omega \vDash X\Phi \ \Leftrightarrow \ \omega(1) \vDash \Phi$$

$$\omega \vDash \Phi_1 \ \mathrm{U}^{\leq n} \ \Phi_2 \ \Leftrightarrow \ \exists i \leq n, \omega(i) \vDash \Phi_2 \ and \ \forall j < i, \omega(j) \vDash \Phi_1$$

$$\omega \vDash \Phi_1 \ \mathrm{U} \ \Phi_2 \ \Leftrightarrow \ \exists k \geq 0, \omega(k) \vDash \Phi_2 \ and \ \forall i < k, \omega(i) \vDash \Phi_1$$

PCTL is an expressive language that can specify many properties. The most important one is the reachability probability [17], which is one of the basic problems of quantitative analysis by system modeled as DTMC. The calculation of the reachability probability of DTMC can be simplified to the solution of linear equations [18]. The two main solutions are:

(1) Direct method, calculate the exact solution (within the numerical error range) in a fixed number of steps, such as Gaussian elimination, L/U decomposition, etc.

(2) Iterative method, which calculates successive approximations of the solution, and terminates when the solution sequence converges to a predetermined accuracy. Including Power [19], Jacobi [20] [21] and Gauss-Seidel [22] [23] methods.

3.3 PRISM

PRISM is a probabilistic model checker, which is a tool for formal modeling and analysis of systems that exhibit random behavior. It analyzes by establishing an accurate mathematical model of a system, and then formally expresses the property of the system in temporal logic, and automatically analyzes the constructed model. PRISM has been used to analyze systems from many different application areas, including communication and multimedia protocols, random distribution algorithms, security protocols, biological systems, etc. PRISM can build and analyze several types of probability models:

- Discrete-time Markov Chain (DTMC)
- Continuous-time Markov Chain (CTMC)
- Markov Decision Process (MDP)
- Probabilistic Automata (PA)
- Probabilistic Timed Automata (PTA)

Models are described using the PRISM language, a simple, state-based language. PRISM provides support for automated analysis of a wide range of quantitative properties of these models, e.g. "what is the probability of a failure causing the system to shut down within 4 hours?", "what is the worst-case probability of the protocol terminating in error, over all possible initial configurations?", "what is the expected size of the message queue after 30 minutes?", or "what is the worst-case expected time taken for the algorithm to terminate?". The property specification language incorporates the temporal logics PCTL, CSL, LTL and PCTL*.

4. Runtime probabilistic model checking based on incremental method

4.1 The framework of incremental runtime probabilistic checking

The main goal of this paper is to improve the verification efficiency of the runtime probabilistic model checking. The research background is based on the fact that the system model has undergone some changes in the complex and changing runtime environment and needs to be verified again. This kind of scenario also often occurs in practical applications. For example, when we need to investigate the impact of changing model parameters on the overall model performance, and online monitoring of properties of an existing system at runtime. In most cases, the migration structure of the model has not changed, but one or several of the state probability migration values have changed and we need to re-verify the model.

The main feature of the runtime probabilistic model checking based on incremental method proposed in this paper is introducing the incremental ideal to the verification process of the runtime probabilistic model checking. Reuse the reachability probabilistic values from the previous model to accelerate the process of calculating and improving the efficiency of the runtime probabilistic model checking.

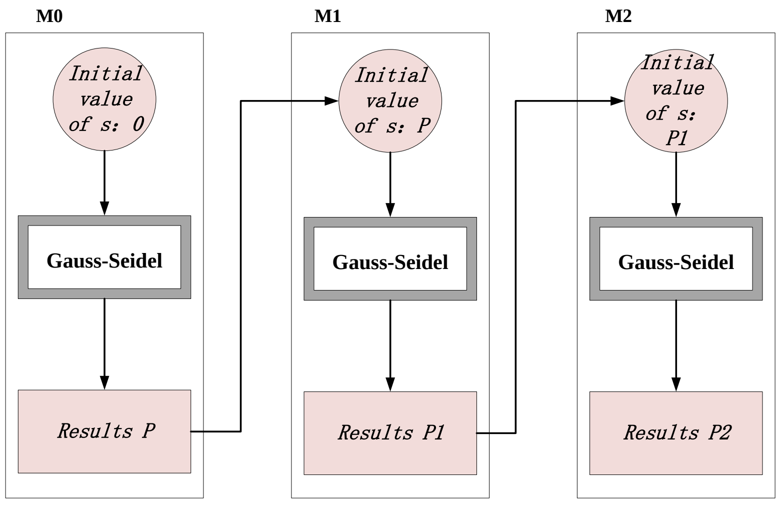

Figure. 2 Incremental runtime probabilistic checking framework

Figure 2 shows the incremental runtime probabilistic model checking framework From the figure, it can be seen that our main work is implemented in the model verification process. *M*, *M1* and *M2* indicate that some state probability transition values have changed during the runtime and thus different system models have been generated. In the traditional runtime probabilistic model checking, when a new model is generated, the new model and system requirements will be re-verified.

When the number of model changes is large and the number of model states is large, this repetitive work will waste a lot of time and energy. This paper combines incremental method in the verification process after the model changes, which is used to reduce the number of numerical iterations in the verification process and improve the efficiency of verification.

4.2 Incremental value iteration

We have developed an incremental value iteration method during the verification process of the probabilistic model checking. The method is based on the Gauss-Seidel iteration method. In the calculation of the reachability properties of stochastic models, the Gauss-Seidel iterative method is a more commonly used iterative method. The Gauss-Seidel iteration is a variant of value iteration. Compared with value iteration, it can improve computing efficiency significantly. The main factor that affects the efficiency of iterative calculation is the number of iterations in the calculation process. It is the most effective method to improve the verification efficiency by reducing the number of iterations. Our method is to reduce the number of iterations is to reuse the calculated value of the previous model. The standard Gauss-Seidel iterative methods normally use the zero vector for the first iteration and iteratively update the vector of values until satisfying the convergence criterion. From the perspective of linear algebra, we know that any other vector of probability values can be used for the first iteration if for every Eigen value λ of the DTMC matrix we have $|\lambda|<1$. In this case, the iterative method will converge to the solution of the reachability probability values. Fortunately, this condition is guaranteed for every matrix of DTMCs. We use this fact to select better start vector for the Gauss-Seidel iterative method. In our approach, the computed vector of reachability values for each version of the model is used as the start vector of values for the new version(after change).

We use P_s^{max} to represent the reachability probability value of a state s in DTMC *M*, and $\bar{P}_s^{max}$ to represent the value of the model *M1* that needs to be verify again after some probability transition values change. In our method, when we need to recalculate $\bar{P}_s^{max}$, the calculated value P_s^{max} (may be a single value or a set of values) of the model *M* before the change can be used as the starting vector of iterative method. We use $P_s^{max,0}$ to represent the iteration initial value of the iterative method for model M, and $\bar{P}_s^{max,0}$ to represent the iteration initial value of the model *M1*. In the first method, $\bar{P}_s^{max,0}$ is updating as follow:

$$\bar{P}_s^{max,0} = \begin{cases} 1 & s \in S^{yes} \\ 0 & s \in S^{no} \\ P_s^{max} & \text{otherwise} \end{cases}$$

5. Experiments

We perform the experiments in a Ubuntu 18.04 LTS system computer, the CPU is i5 and the memory is 8GB. The experimental tool is the probabilistic model checker PRISM, the version is 4.5, and the basic configuration of PRISM is shown in Table 1.

Table 1 Configuration of PRISM

Name	Parameter
Engine	Sparse
Solution Method	Gauss-Seidel
Memory	1g

The paper focus on the DTMC model and reachability property, we used Crowds (Crowds Protocol) model from the PRISM benchmark case. Crowds protocol is an anonymous communication protocol. By storing and forwarding data at the application layer, the sender's data reaches the receiver through a routing path composed of multiple relay nodes, because each node on the path is difficult to judge Whether the predecessor node is the initial initiator of the message or an intermediate forwarding node, so as to hide the sender information and achieve anonymity. The property of Crowds model that needs to be verified is R1: the probability that the adversary observes the real sender more than once. The property can be translated into PCTL as shown in Table 2.

Table 2 Properties translation in PCTL

Model	Property	PCTL
Crowds	*R1*	"positive": $P = ?\ [\,F\ observe > 1\,]$

For each type of model, we complete the four DTMC model experiments by setting parameters, and record the relevant model information, including the parameter settings of each model, the number of states and transitions in the model. As shown in Table 3.

Table 3 Information of models

Model	**Parameter Values**	**Number of States**	**Number of Transitions**
Crowds	TotalRuns=5, Crowdsize=20	2061951	7374951
	TotalRuns=6, Crowdsize=10	352535	833015
	TotalRuns=6, Crowdsize=15	2464168	7347928
	TotalRuns=8, Crowdsize=10	2529567	6030207

In order to verify the effectiveness of our method, we implement the two methods for each model in PRISM: The standard incremental verification method, the incremental verification method combining with the incremental value iteration method. In the experiment, considering the uncertainty of the model change in actual system, we use the parameter Beta to represent the number of states which transition values have changed. We have considered Beta=300 in the experiments. In the Table 4. we record the running time of each model, and all recorded times are in seconds.

Table 4 Time of model checking (Beta=300)

Model	Parameter Values	Standard runtime method	Incremental value iteration
Crowds	TotalRuns=5, Crowdsize=20	7.324	3.678
	TotalRuns=6, Crowdsize=10	2.72	1.452
	TotalRuns=6, Crowdsize=15	13.658	7.055
	TotalRuns=8, Crowdsize=10	25.88	14.28

According to the above experimental results, it can be seen that the incremental method we proposed can effectively reduce the time of the model checking. For the Crowds model, in most cases, combining the incremental method reduces the verification time to less than 56% of the standard method.

6. Conclusion

We have proposed a runtime probabilistic model checking based on incremental method in the paper. The main work of the paper includes: First, analyze the principles of the existing runtime probabilistic model checking, and summarize the limitations of some verification methods. Secondly, introduce the idea of incremental value iteration and propose the runtime probabilistic model checking based on incremental method. Finally, we implement the proposed method through a benchmark case model in PRISM.

References

[1] Baier C, Katoen J P. Principles of model checking [M]. MIT press, 2008.

[2] LIU Yang, LI Xuan-Dong, MA Yan, WANG Lin-Zhang. Survey for stochastic model checking [J]. Chinese Journal of Computers, 2015, 000 (011): 2145-2162.

[3] Kwiatkowska M, Norman G, Parker D. PRISM 4.0: Verification of probabilistic real-time systems [C] //International conference on computer aided verification. Springer, Berlin, Heidelberg, 2011: 585-591.

[4] Dehnert C, Junges S, Katoen J P. A storm is coming: A modern probabilistic model checker [C] //International Conference on Computer Aided Verification. Springer, Cham, 2017: 592-600.

[5] Filieri A, Ghezzi C, Tamburrelli G. Run-time efficient probabilistic model checking [C] //2011 33rd International Conference on Software Engineering (ICSE). IEEE, 2011: 341-350.

[6] Christoph Prybila,Stefan Schulte,Christoph Hochreiner,Ingo Weber. Runtime verification for business processes utilizing the Bitcoin blockchain [J]. Future Generation Computer Systems,2020,107.

[7] Klein J, Baier C, Chrszon P. Advances in probabilistic model checking with PRISM: variable reordering, quantiles and weak deterministic Büchi automata

[J]. International Journal on Software Tools for Technology Transfer, 2018, 20 (2): 179-194.

[8] Abrahám E, Jansen N, Wimmer R. DTMC model checking by SCC reduction [C] //2010 Seventh International Conference on the Quantitative Evaluation of Systems. IEEE, 2010: 37-46.

[9] Brázdil T, Chatterjee K, Chmelik M. Verification of Markov decision processes using learning algorithms [C] //International Symposium on Automated Technology for Verification and Analysis. Springer, Cham, 2014: 98-114.

[10] Kwiatkowska M, Parker D, Qu H. Incremental quantitative verification for Markov decision processes [C] //2011 IEEE/IFIP 41st International Conference on Dependable Systems & Networks (DSN). IEEE, 2011: 359-370.

[11] Forejt V, Kwiatkowska M, Parker D. Incremental runtime verification of probabilistic systems [C] //International Conference on Runtime Verification. Springer, Berlin, Heidelberg, 2012: 314-319.

[12] Sokolsky O V, Smolka S A. Incremental model checking in the modal mu-calculus [C] //International Conference on Computer Aided Verification. Springer, Berlin, Heidelberg, 1994: 351-363.

[13] Conway C L, Namjoshi K S, Dams D. Incremental algorithms for inter-procedural analysis of safety properties [C] //International Conference on Computer Aided Verification. Springer, Berlin, Heidelberg, 2005: 449-461.

[14] Heljanko K, Junttila T, Latvala T. Incremental and complete bounded model checking for full PLTL [C] //International Conference on Computer Aided Verification. Springer, Berlin, Heidelberg, 2005: 98-111.

[15] Wongpiromsarn T, Ulusoy A, Belta C. Incremental temporal logic synthesis of control policies for robots interacting with dynamic agents [C] //2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2012: 229-236.

[16] Abrahám E, Jansen N, Wimmer R. DTMC model checking by SCC reduction [C] //2010 Seventh International Conference on the Quantitative Evaluation of Systems. IEEE, 2010: 37-46.

[17] Baier C, Klein J, Leuschner L. Ensuring the reliability of your model checker: Interval iteration for Markov decision processes [C] //International Conference on Computer Aided Verification. Springer, Cham, 2017: 160-180.

[18] Katoen J P. The probabilistic model checking landscape [C] //Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science. 2016: 31-45.

[19] Elman H C, Saad Y, Saylor P E. A hybrid Chebyshev Krylov subspace algorithm for solving nonsymmetric systems of linear equations [J]. SIAM Journal on Scientific and Statistical Computing, 1986, 7 (3): 840-855.

[20] Sleijpen G L G, Van der Vorst H A. A Jacobi--Davidson iteration method for linear eigenvalue problems [J]. SIAM review, 2000, 42 (2): 267-293.

[21] Milaszewicz J P. Improving jacobi and gauss-seidel iterations [J]. Linear Algebra and Its Applications, 1987, 93: 161-170.

[22] Kohno T, Kotakemori H, Niki H, et al. Improving the modified Gauss-Seidel method for Z-matrices [J]. Linear Algebra and its Applications, 1997, 267: 113-123.

[23] Ortega J M, Rheinboldt W C. Monotone iterations for nonlinear equations with application to Gauss-Seidel methods [J]. SIAM Journal on Numerical Analysis, 1967, 4 (2): 171-190.