Automated Software Engineering (2022) 29:29
https://doi.org/10.1007/s10515-022-00327-z

A three-valued model abstraction framework for PCTL* stochastic model checking

Yang Liu[1,3] · Yan Ma[2,3] · Yongsheng Yang[1]

Received: 30 July 2021 / Accepted: 24 January 2022 / Published online: 2 March 2022
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2022

Abstract
Stochastic model checking can automatically verify and analyse the software-driven autonomous systems with stochastic behaviors, which is a formal verification technique based on system models. When coping with large-scale systems, it suffers from state space explosion problem very seriously. Model abstraction is a potential technique for mitigating this problem. At present, only a few properties specified by PCTL (Probabilistic Computation Tree Logic), such as probabilistic safety and probabilistic reachability, can be preserved in the practical model abstraction of stochastic model checking, which are the proper subset of PCTL* (Probabilistic Computation Tree Logic*) properties. For dealing with this, an effective and efficient three-valued model abstraction framework for full PCTL* stochastic model checking is proposed in this paper. We propose a new abstract model to preserve full PCTL* properties for nondeterministic and probabilistic system, which orthogonally integrates interval probability of transition and game for nondeterminism. A game-based three-valued PCTL* stochastic model checking algorithm is developed to verify abstract model, and a BPSO (binary particle swarm optimization) algorithm integrated with sample learning is designed to refine the indefinite result of three-valued PCTL* stochastic model checking abstract model. It is proved that full PCTL* properties are preserved when the result of three-valued stochastic model checking is definite, and the efficiency of this framework is demonstrated by some large cases.

Keywords Stochastic model checking · Three-valued model abstraction · Game · Abstraction-refinement · BPSO

✉ Yan Ma
yanma_nus@126.com

Extended author information available on the last page of the article

1 Introduction

Software-driven autonomous systems are increasingly being used in a lot of domains, e.g., industry automation, transport, finance, medical surgery and so on (Ebert and Weyrich 2019; Shivakumar et al. 2020; Fremont et al. 2020).

They are becoming larger and more complex, some of which are accompanied with the stochastic behaviors (Clarke et al. 2018). The essential reasons for exhibiting stochastic behaviors can be classified as, among others: (a) the system itself contains randomness, e.g., probabilistic algorithms or randomized algorithms are included; (b) the running environment of system is open, dynamic and unmanageable, which may lead to random failures, e.g., message loss or failing to invoke some components; (c) the stochastic variables are artificially employed to capture system performance index for evaluation and analysis, e.g., reliability. As an automatic and formal verification technique, stochastic model checking (a.k.a. probabilistic model checking) (Clarke et al. 2018, 2009; Kwiatkowska et al. 2022; Baier and Katoen 2008) can be naturally applied to quantitatively analyse intelligent software-driven autonomous systems with stochastic behaviors. In practice, stochastic model checking has been exploited to analyze or guarantee the correctness of probabilistic program (Hark et al. 2020), system performance of mobile service robots (Lacerda et al. 2019), reliability of critical communication protocols (Oxford et al. 2020), resource control mechanisms in cloud computing (Evangelidis 2020), software adaptation of an unmanned undersea vehicle (Pfeffer et al. 2019), trustworthiness of deep learning systems (Kwiatkowska 2019), and so on. In the verification process, the autonomous systems are modelled as full probabilistic models, e.g., DTMCs (Discrete-time Markov Chains), PPNs (Probabilistic Petri Nets), or nondeterministic and probabilistic models, e.g., MDPs (Markov Decision Processes), NPPNs (Nondeterministic Probabilistic Petri Nets); and the requirement properties are specified by LTL (Linear Temporal Logic) with probability or PCTL (Probabilistic Computation Tree Logic) (Kwiatkowska et al. 2022).

The biggest obstacle is state space explosion in stochastic model checking the software-driven autonomous systems, especially the large-scale systems. This is rooted in the facts that: (a) the number of states grows double exponentially in the number of variables or components in a software-driven autonomous system; (b) stochastic model checking algorithm combines the classical model checking algorithm and linear equation solving or linear programming algorithms, which computes the probabilistic vector over states rather than bit-vector in classical model checking. Hence, how to fight state space explosion problem is a major challenge in the field of stochastic model checking software-driven autonomous systems. Clarke, Turing Award winner for founding the field of model checking, points out that it is an important direction in the future research of model checking at "Turing Lecture" (Clarke et al. 2009). In recent years, some techniques have been proposed for combating this problem, ranging from the multi-terminal binary decision diagram (MTBDD) (Kwiatkowska et al. 2017), abstraction (Dehnert 2018; Dams and Grumberg 2018), and bounded stochastic model checking algorithm (Hartmanns et al. 2018) to compositional reasoning (He et al. 2016; Ma et al. 2019a). However, it is still an open problem.

1.1 Problem statement of model abstraction for stochastic model checking

As an important means to tackle the state space explosion problem, abstraction (Clarke et al. 2003, 1994a) has been applied in the field of stochastic model checking. It performs stochastic model checking on the abstraction of one or more components in the concrete stochastic model checking. Concrete stochastic model checking is to decide whether $M \vDash \Phi$ holds, where M is the stochastic system model, Φ is the quantitative requirement property specification, and $\vDash$ is the satisfaction relation. Abstraction for stochastic model checking can be denoted as: $M^A \vDash^A \Phi^A$, where M^A is the model abstraction of M, $\vDash^A$ is the relation abstraction of $\vDash$ based on algorithm, Φ^A is the specification abstraction of Φ. In fact, M^A usually leads to $\vDash^A$ and Φ^A; $\vDash^A$ also results in Φ^A; Φ^A may also lead to M^A and $\vDash^A$. In the present research on abstraction for stochastic model checking, the abstraction refers to model abstraction by default; while relation abstraction (Alfaro and Roy 2007; Didier et al. 2010; Filieri et al. 2011; Huang et al. 2019; Younes 2005) and specification abstraction (Dehnert 2018; Dams and Grumberg 2018) are specifically identified, and the related works of them is less. This paper also focuses on model abstraction for stochastic model checking.

Informally, model abstraction for stochastic model checking is omitting details from concrete stochastic system model, which are not relevant for verifying properties under consideration(Liu et al. 2015). It concludes the value of $M \vDash \Phi$ from the result of $M^A \vDash \Phi$. Applying model abstraction in stochastic model checking faces the following four issues, which is named MA4SMC (model abstraction for stochastic model checking) problem: P1) how to construct abstract model, P2) how to verify the abstract model, P3) what quantitative properties can be preserved, P4) how to present the counterexample. Thereinto, the answer to P1) is the key for solving the MA4SMC problem. It has an important effect on the answers to P2), P3) and P4). The goal of model abstraction is to make the abstract model that has a small enough state space, yet contains abundant information of concrete model. In other words, perfect model abstraction technology should meet the following conditions: C1) the state space of abstract model M^A is significantly less than the concrete model M; C2) the results of stochastic model checking M^A is equivalent to the results of stochastic model checking M, i.e., $M^A \vDash \Phi \rightarrow M \vDash \Phi$, and $M^A \nvDash \Phi \rightarrow M \nvDash \Phi$. That is to say, if M^A satisfies Φ, M satisfies Φ; and if M^A doesnot satisfy Φ, M doesnot satisfy Φ. However, abstraction process will cause information loss inevitably. There is a tradeoff in abstraction process between state space and information loss, and various model abstraction techniques choose the different tradeoffs.

The existing model abstraction techniques, discussed elaborately in Sect. 2, can be divided into two categories: (a) accurate abstraction, e.g., probabilistic bisimulation-based abstraction (Larsen and Skou 1991; Milner 1980, 1971; Park 1981; Buchholz 1994; Segala and Lynch 1995; Paige and Tarjan 1987; Huynh and Tian 1992; Hermanns and Katoen 2000; Derisavi 2007; Christian et al. 2013; Zhang et al.

2018; Baier and Hermanns 1997; Philippou et al. 2000; Ferrer et al. 2016; Hermanns and Turrini 2012; Jonsson and Larsen 1991; Clarke et al. 1994b; Baier et al. 2005a; Zhang 2008; Zhang and David 2016; Hashemi et al. 2013), symmetry reduction-based abstraction (Clarke et al. 1996; Emerson and Sistla 1996; Norris and Dill 1996; Miller et al. 2006; Donaldson and Miller 2006; Emerson and Wahl 2003, 2005a, b; Donaldson et al. 2009; Wahl et al. 2008; Kwiatkowska et al. 2006a; Kamaleson 2018; Christopher 2012) and partial order reduction-based abstraction (Gerth et al. 1995; Peled 1993, 1996; Peled et al. 1997; Valmari 1992; Baier et al. 2004, 2005b; D'Argenio and Niebert 2004; Ciesinski 2011; Fernandez-Diaz et al. 2012; Hansen et al. 2011; Kwiatkowska et al. 2011; Hansen and Wang 2011; Winterer et al. 2017); (b) approximate abstraction, e.g., probabilistic CEGAR (counterexample-guided abstraction-refinement) (Clarke et al. 2003; Hermanns et al. 2008; Hahn et al. 2010; Chadha and Viswanathan 2010; Ma et al. 2019b), error-guided abstraction (Ma et al. 2019b; Kwiatkowska et al. 2006b; Kattenbelt et al. 2010; Katoen and Sher 2017; Wachter and Zhang 2010; Winterer et al. 2020; Kwiatkowska et al. 2020), indefinite result-guided abstraction (Fecher et al. 2006; Katoen et al. 2012; Luisa et al. 2017). In accurate abstraction, the abstract model is the quotient of concrete model, verification algorithm and counterexample generation are the same with traditional stochastic model checking. The abstract model of exact abstraction is not small enough, but it preserves almost all the properties. The accurate abstraction is difficult to realize in practical application at technical level, as it cannot reduce state space obviously, and the system models need to have the corresponding special structure. In approximate abstraction, probabilistic CEGAR has a wide range of practical applications, but the preserved properties are too less, only probabilistic safety is preserved. Error-guided abstraction and indefinite result-guided abstraction are the potential model abstraction paradigms for stochastic model checking. The limitations of error-guided abstraction are: (1) the scope of preserved properties is too narrow, only probabilistic reachability, (2) the refinement algorithm is not efficient enough, and (3) counterexample generation is not included. Indefinite result-guided abstraction can preserve PCTL properties, but it isnot complete, which solves the MASMC problem partially, and the limitations of which are: (1) the verified system model is full probabilistic model, (2) the refinement is not involved for steering repartition of abstract model, and (3) counterexample generation is also not included.

1.2 Our contributions

For preserving more properties in model abstraction, we argue that error-guided abstraction and indefinite result-guided abstraction can be combined orthogonally to form an ideal abstraction framework, as they have the complementary advantages. Game can characterize the new nondeterminism occurred by abstraction, which can reduce the state space of abstract model. Indefinite result-guided abstraction can over-approximate and under-approximate the concrete system model at the same time, which can get the tight lower and upper bounds. This paper is dedicated to this direction.

In this paper, with a breakthrough in constructing and presenting abstract model, we propose a complete and efficient model abstraction framework, i.e., three-valued abstraction-refinement, for PCTL* (Probabilistic Computation Tree Logic*) stochastic model checking nondeterministic and probabilistic systems. As shown in Fig. 1, we mainly make the following contributions to solve the MA4SMC problem: (1) proposing a new abstract model which orthogonally integrates interval probability of transition and game for nondeterminism, i.e. LGIPPN (Game Interval PPN with Label), and combining sample learning and BPSO (binary particle swarm optimization) algorithm to refine the abstract model; (2) generalizing two-player (*verifier* and *refuter*) stochastic game semantics (Liu et al. 2016; Shoham and Grumberg 2007) for three-valued PCTL* stochastic model checking the abstract model, and exploiting coloring process for strategy solving; (3) preserving the full PCTL* properties which is a proper superset of probabilistic safety, probabilistic reachability, and PCTL properties; (4) extending the evidence for refutation (Liu et al. 2016; Shoham and Grumberg 2007) to express the counterexample. This solution meets the conditions of perfect model abstraction well: (1) the state space of abstract model M^A is significantly less than the concrete model M; C2) $M^A \vDash \Phi \rightarrow M \vDash \Phi$, $M^A \nvDash \Phi \rightarrow M \nvDash \Phi$, except for the indefinite result. As far as we know, this is the first complete framework for three-valued abstraction-refinement in stochastic model checking, and it is also the first abstraction framework for preserving full PCTL*. In this framework, the concrete system model that we deal with is LNPPN (Nondeterministic Probabilistic Petri Net with Label) (Albanese et al. 2008; Liu et al. 2016; Bernemann et al. 2020) which is a high-level formal model for modelling autonomous systems with nondeterministic and probabilistic behaviors. The LNPPN model can be modelled from an existing autonomous system, or it is a design model by designer for developing the autonomous system. Note that this framework can also be used to

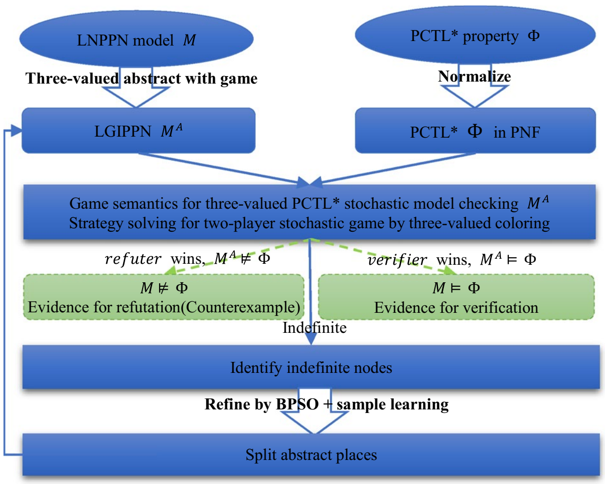
LNPPN model M
PCTL* property Φ
Three-valued abstract with game
Normalize
LGIPPN M^A
PCTL* Φ in PNF
Game semantics for three-valued PCTL* stochastic model checking M^A
Strategy solving for two-player stochastic game by three-valued coloring
refuter wins, $M^A \nvDash \Phi$
verifier wins, $M^A \vDash \Phi$
$M \nvDash \Phi$
Evidence for refutation(Counterexample)
$M \vDash \Phi$
Evidence for verification
Indefinite
Identify indefinite nodes
Refine by BPSO + sample learning
Split abstract places

Fig. 1 Three-valued abstraction-refinement for PCTL* stochastic mode checking

abstract other nondeterministic and probabilistic systems models, such as MDP or probabilistic automaton.

1.3 Outline of the paper

In the next section, we survey the related works of model abstraction techniques for stochastic model checking. We give the necessary preliminaries for LNPPN, PCTL* and stochastic model checking LNPPN in Sect. 3. In Sect. 4, we propose a novel abstract model for nondeterministic and probabilistic system, and describe how to construct the abstract models for LNPPN. Section 5 extends the game-based PCTL* stochastic model checking to verify abstract model, which transforms the three-valued PCTL* stochastic model checking into strategy solving of three-valued stochastic game. We propose the refinement algorithm for refining the abstract model, which integrates BPSO and sample learning in Sect. 6. The experimental results are presented in Sect. 7. We conclude the paper in Sect. 8.

2 Related works

There are some model abstraction techniques have been proposed for stochastic model checking. We classify them according to the methods of presenting and constructing abstract model, summarise their solutions to MA4SMC problem, and analyse the extent to which they satisfy the conditions of perfect abstraction.

2.1 Simulation relation-based model abstraction

(1) Strong probabilistic bisimulation

Strong probabilistic bisimulation (Larsen and Skou 1991), i.e., probabilistic bisimulation, is the quantitative extension of strong bisimulation (Milner 1980; Park 1981) on LTS (labelled transition system). It has been used for model abstraction for DTMC (discrete-time Markov chain), CTMC (discrete-time Markov chain) (Buchholz 1994) and MDP (Markov decision process) (Segala and Lynch 1995). Based on study (Paige and Tarjan 1987), Huynh et al. (1992) gave the first automatic constructing abstract model algorithm for DTMC. Hermanns et al. (2000) adopted compositional method to transform the strong probabilistic bisimulation of model into the strong probabilistic bisimulation of submodel, and used it to quantitatively verify POTS (plain-old telephone system). Derisavi (2007) firstly presented a symbolic algorithm and its implementation for the construct abstract model of CTMC, which was optimal for generating the smallest possible abstract model. Christian et al. (2013) leveraged satisfiability solvers to extract the minimised system from the PRISM modelling language directly, which could generate coarser abstract model and no state space was generated. Song et al. (Zhang et al. 2018) introduced novel notions of strong probabilistic bisimulation relation for PA (probabilistic automaton), which could get the coarsest abstract model among the existing strong probabilistic bisimulation techniques. Strong probabilistic bisimulation-based abstraction solves the MA4SMC problem in the following way: W1) use the strong probabilistic bisimulation minimization to construct a smaller equivalent abstract model, and use the quotient model to present abstract model; W2) use existing stochastic model checking algorithm to verify abstract model; W3) preserve all the quantitative temporal logics; W4) use existing method to generate the counterexample. It meets condition C2) of perfect model abstraction well, and performs badly for condition C1). The limitation of it: the abstract model is so refined that the state space of it does not become smaller obviously, and the time to construct the abstract model may exceed the stochastic model checking time for concrete model.

(2) Weak probabilistic bisimulation

Compared with Strong probabilistic bisimulation, weak probabilistic bisimulation (Zhang et al. 2018; Baier and Hermanns 1997; Philippou et al. 2000) is a coarse simulation relation. It solves the MA4SMC problem in the following way: W1) use the weak probabilistic bisimulation minimization to construct a smaller stuttering equivalent abstract model, and use the quotient model to present abstract model; W2) use existing stochastic model checking algorithm to verify abstract model; W3) preserve all the quantitative temporal logics without next operator; W4) use existing method to generate the counterexample. Although some work (Baier and Hermanns 1997; Ferrer et al. 2016; Hermanns and Turrini 2012) manage to reduce the time complexity of constructing abstract model, it is still subject to the cost of a complex minimization algorithm.

(3) Strong probabilistic simulation

Strong probabilistic simulation (Larsen and Skou 1991; Jonsson and Larsen 1991) can be seen as the quantitative extension of strong simulation (Milner 1971; Clarke et al. 1994b). Baier et al. (2005a) proved that equivalence of strong probabilistic simulation over DTMC or CTMC was in accordance with strong probabilistic bisimulation. For MDP, the quotient under strong probabilistic simulation is strict coarser than strong probabilistic bisimulation. Zhang et al. (2008, 2016) pursued the optimization of constructing abstract model based on strong probabilistic bisimulation, and Huang et al. (2019) generalized probabilistic simulation to probabilistic pushdown automata and finite-state systems. Strong probabilistic simulation-based abstraction solves the MA4SMC problem in the following way: W1) use the strong probabilistic simulation minimization to construct a smaller abstract model, and use the quotient model to present abstract model; W2) use existing stochastic model checking algorithm to verify abstract model; W3) preserve the truth value of quantitative safety temporal logics; W4) use existing method to generate the counterexample. It meets the perfect model abstraction C1) well, but for C2), it just preserves the truth value of quantitative safety temporal logics formula Φ, i.e., $M^A \vDash \Phi \rightarrow M \vDash \Phi$.

(4) Weak probabilistic simulation

Weak probabilistic simulation is the coarsest among all the simulation relations. At present, the related work about weak probabilistic simulation is less (Zhang 2008; Zhang and David 2016; Hashemi et al. 2013). It solves the MA4SMC problem in the following way: W1) use the weak probabilistic simulation minimization to construct a smaller abstract model, and use the quotient model to present abstract model; W2) use existing stochastic model checking algorithm to verify abstract model; W3) preserve the truth value of quantitative safety temporal logics without next operator; W4) use existing method to produce the counterexample. It meets the perfect model abstraction C1) well, but for C2), it just preserves the truth value of quantitative safety temporal logics without next operator.

2.2 Symmetry reduction-based model abstraction

Symmetry reduction (Clarke et al. 1996; Emerson and Sistla 1996; Norris and Dill 1996; Miller et al. 2006) exploits presence of replication within model to be verified. Donaldson et al. (2006) applied symmetry reduction (Emerson and Wahl 2003, 2005a) for non-probabilistic symbolic model checking in PRSIM with MDP or DTMC semantics, translated the SP (symmetric PRISM) program into the reduced form, and developed corresponding tool GRIP. Based on study (Donaldson and Miller 2006), Donaldson et al. (2009) presented a much richer language, which allowed specification of probabilistic systems in a way that guaranteed the applicability of the generic representative technique, together with an extended translation algorithm. Via dynamic symmetry reduction (Emerson and Wahl 2005b; Wahl et al. 2008), Kwiatkowska et al. (2006a) proposed an efficient algorithm for the construction of quotient models of DTMC, CTMC, MDP, which were built PRISM with MTBDD data structure. Kamaleson (2018) and Christopher (2012) applied symmetry reduction to stochastic model checking with explicit states, which could the automated detect component symmetries or arbitrary data in the probabilistic specification. Symmetry reduction-based abstraction uses quotient to present abstract model. The relation between it and concrete model is probabilistic bisimulation, but it needs less cost when constructing abstract model compared with probabilistic bisimulation-based abstraction.

2.3 Partial order reduction-based model abstraction

Partial order reduction methods (Gerth et al. 1995; Peled 1993; Peled et al. 1997; Valmari 1992) rely on expanding a state space only partially, exploring representatives of sets of executions of a system. Baier et al. (2004) firstly investigated partial order reduction for LTL without next operator model-checking MDP via a variant of Peled's ample set method (Peled 1993, 1996). Argenio et al. (2004) enhanced ample set conditions of study (Baier et al. 2004) to make the abstraction preserve maximum and minimum probabilities of next-free LTL properties, which was implemented in LiQuor (Ciesinski 2011). Baier et al. (2005b) presented the partial order reduction criteria for verifying branching time properties formalized by PCTL. Fernandez et al. (2012) regarded the work of Baier et al. (2004, 2005b), D'Argenio and Niebert (2004) as the dynamic partial order reduction methods, the drawback of which was that they can hardly be combined with other techniques to tackle the state space explosion problem. It studied partial order reduction realized by a static analysis which injected the reduction criteria into the control flow graph. Different from partial order reduction via ample set, Hansen et al. (2011) and Kwiatkowska et al. (2011) firstly proposed a weak variant of the stubborn set (Hansen and Wang 2011) to reduce the state space of MDP, which would preserve the maximal probabilities of reaching bottom end components under far schedulers and realizability of unconditional fairness. This work is then extended in Winterer et al. (2017). Partial order reduction-based abstraction can be seen as the weak probabilistic bisimulation-based abstraction.

2.4 Probabilistic counterexample-guided model abstraction

CEGAR (counterexample-guided abstraction-refinement) (Clarke et al. 2003) has been en vogue for the automatic verification of very large systems in the past years. At present, there are two methods for applying CEGAR in stochastic model checking, i.e., probabilistic CEGAR (Hermanns et al. 2008; Chadha and Viswanathan 2010), which are based on over-approximation of concrete model. Hermanns et al. (2008) firstly applied CEGAR for stochastic model-checking probabilistic automaton, and developed the corresponding tool PASS (Hahn et al. 2010). It solves the MA4SMC problem in the following way: W1) use counterexample-guided refinement to construct abstract model, and use the abstract quotient automaton to present abstract model; W2) use existing stochastic model checking algorithm to verify abstract model; W3) preserve the truth value of reachability properties; W4) use existing method to produce the counterexample expressed by a finite Markov chain. It meets condition C1) of perfect model abstraction well, and performs badly for condition C2), for reachability property Φ, i.e., $M^A \vDash \Phi \rightarrow M \vDash \Phi$. Chadha et al. (2010) pointed out that DTMC could not serve as counterexample for the richer class of properties and argued that no formal statement characterizing process based on the refinement algorithm outlined in Hermanns et al. (2008). This was supplemented by Ma et al. (2019b), in which counterexample was represented with subgraph. They can be seen as the second solution for MA4SMC problem in the following way: W1) use counterexample-guided refinement to construct abstract model, and use the another MDP or sub-graph to present abstract model; W2) use existing stochastic model checking algorithm to verify abstract model; W3) preserve the truth value of safety and liveness fragments of PCTL; W4) use existing method with heuristics to generate the counterexample expressed by a sub-MDP or sub-graph. It meets condition C1) of perfect model abstraction well, and does not perform well for condition C2), i.e., for safety and liveness fragments of PCTL formulae Φ, i.e., $M^A \vDash \Phi \rightarrow M \vDash \Phi$.

2.5 Error-guided model abstraction

Error-guided abstraction-refinement (game-based abstraction) proposed by Marta, Mark, et al. (Kwiatkowska et al. 2006b; Kattenbelt et al. 2010; Katoen and Sher 2017) is a new framework for model abstraction in stochastic model checking. Error-guided abstraction-refinement framework comprises an abstraction based on stochastic 2-player games, two refinement methods (strategy-based refinement, value-based refinement) and an efficient algorithm for an abstraction-refinement loop. Recent extensions to the PASS tool (Wachter and Zhang 2010) use this framework and demonstrate that it is faster and yields smaller abstractions. Winterer et al. (2020) extended and complemented the work (Katoen and Sher 2017) for strategy synthesis for POMDPs in robot planning. The solution of game-based abstraction to the MA4PMC problem is as follows: W1) use error-guided refinement to construct abstract model, and use stochastic game to present abstract model; W2) use existing stochastic game algorithm (Kwiatkowska et al. 2006b, 2020) to analyze abstract model; W3) allows lower and upper bounds to be computed for the values of reachability properties of the MDP; W4) do not involve the counterexample generation. It meets condition C1) of perfect model abstraction well, but the properties to be verified is limited to reachability properties.

2.6 Indefinite result-guided model abstraction

In recent years, indefinite result-guided abstraction-refinement (three-valued abstraction) (Fecher et al. 2006) is used in stochastic model checking, which is both over-approximation and under-approximation of concrete model. Fecher et al. (2006) considered three-valued abstraction for DTMC firstly. It solves the MA4SMC problem partially in the following way: W1) use abstract Markov chain to present abstract model, do not involve refinement method; W2) use a dedicated three-valued stochastic model checking to verify abstract model; W3) preserve PCTL of the DTMC, except for the indefinite results; W4) do not include the counterexample generation. Katoen et al. (2012) extended the work of Fecher et al. (2006) to CTMC, and laid down the theoretical underpinnings of three-valued abstraction for DTMC and CTMC. Luisa et al. (2017) extended the work of Fecher et al. (2006), Katoen et al. (2012) to Spatio-Temporal Logic for stochastic systems, and Belardinelli et al. (2019) generalized the three-valued abstraction to verify strategic properties in multi-agent systems with imperfect information. It solves the MA4SMC problem partially in the following way: W1) use interval Markov chain to present abstract model, do not involve refinement method; W2) use a dedicated three-valued stochastic model checking to verify abstract model; W3) preserve PCTL/CSL of the DTMC/CTMC, except for the indefinite results; W4) do not include the counterexample generation. The work of Katoen et al. (2012), Luisa et al. (2017), Belardinelli et al. (2019) meets condition C1) and C2) of perfect model abstraction well, but they are incomplete, and there is not yet an implementation to test this on practical examples.

3 Preliminaries

3.1 LNPPN

Petri net is a high-level formal method for modeling, analyzing, and verifying the complex systems. It was proposed by C. A. Petri in his dissertation "Communication with Automata" in 1962. Today, it has been widely used in all aspects of software or system engineering to ensure the quality. For modeling the complex system with stochastic behaviors, probability measurement theory is introduced in Petri net (Albanese et al. 2008; Liu et al. 2016; Bernemann et al. 2020), which is under the guidance of general net theory (Petri 1979). At present, there are 4 types of probabilistic Petri nets for modelling stochastic systems, as shown in Table 1. We consider the nondeterministic probabilistic system models, i.e., NPPN. PPN can be seen as the special case of NPPN without nondeterminism. LNPPN (Liu et al. 2016) is the NPPN with label functions.

Definition 1 LNPPN (NPPN with Label). The LNPPN can be defined as a 7-tuple $M = (S,\ T; F,\ f; C; AP,\ L)$, where: (1) $T = (Transition, Prt)$, $Transition$ denotes the transition act, $Prt \in [0,1]$ denotes the success probability of the transition; T is the act transition (AT) with the probability equals 1, or the probabilistic act transition (AT, Prt) with an act satisfying a certain probability distribution, or the pure probability transition (PT) without any act. If $Prt = 0$, it means that the transition is invalid; (2) S is the set of places,$S \cap T = \phi$, $S \cup T \neq \phi$, $F \subseteq S \times T \cup T \times S$, which is the flow relation of net, and $N = (S, AT, F)$ is the pure net, where AT is the act transition with probability equals 1; (3) $f = f_T \cup f_S \cup f_{S\times T} \cup f_{T\times S}$, $f_T : T \rightarrow 2^{Transition \times Prt}$, $f_S = S \rightarrow [0,1]$, $f_{S\times T} = S \times T \rightarrow [0,1]$, $f_{T\times S} = T \times S \rightarrow [0,1]$, and the value of $f_{S\times T}$ is determined by the nondeterminism of transitions, the value of $f_{T\times S}$ can be obtained from the probability value of $f_T(t)$, i.e., $\sigma_{Prt}\left(f_T(t)\right)$, the value of f_S except initial place can be computed according to the value of $f_{S\times T}$ and $f_{T\times S}$; (4) $\forall t \in T$, $\exists s \in S, f_{T\times S} = 1 - \sigma_{Prt}\left(f_T(t)\right). f_{T\times S}(t \times s) = 0$, if $\sigma_{Prt}\left(f_T(t)\right) = 1$, which represents transition (t, s) and place s are invalid; (5) C is the set of nondeterminism classes, and each nondeterminism class is a set comprised of $\left(s, t_i\right)$. If $\left\{\left(s, t_1\right), \left(s, t_1\right) \dots \left(s, t_1\right)\right\} \in C$, then $\sum_{i=1}^{n} f_{S\times T}\left(s, t_i\right) = 1$; (6) AP is a set of atomic propositions; (7) $L : S \rightarrow 2^{AP}$ is the labeling function, which can express the requirements of users, i.e., the properties.

Table 1 Petri nets with probability measurement theory

	Fully probabilistic	Nondeterministic + probabilistic
Discrete time	Probabilistic Petri net (PPN)	Nondeterministic probabilistic Petri net (NPPN)
Continuous time	Stochastic Petri net (SPN)	Probabilistic timed Petri net (PTPN)

A LNPPN M is finite, if S and T are finite. The size of M is the number of places and transitions plus the number of pairs (s, t) with $f_{S\times T} > 0$ and (t, s) with $f_{T\times S} > 0$. A LNPPN model is measurable with probability (Liu et al. 2016).

Definition 2 Adversary (Policy, or Scheduler) of LNPPN. An adversary of a LNPPN model M is a function $Adv : S^+ \to T$, such that $Adv\left(s_0s_1s_2 \dots s_n\right) \in C$ for all $s_0s_1s_2 \dots s_n \in S^+$. The path $\pi = s_0 \to^{t_1} s_1 \to^{t_2} \cdots \to^{t_i} s_i \cdots$ is called a Adv-path if $t_i = Adv\left(s_0 \dots s_{i-1}\right)$ for all $i > 0$.

3.2 PCTL* in release-PNF

Every PCTL* (Baier and Katoen 2008) formula can be transformed as a canonical form. Release-PNF (positive normal form) of PCTL* is a kind of canonical forms for PCTL*, which is described by a restriction that negation operator only occurs adjacent to atomic propositions. It can avoid the exponential blowup in transforming the PCTL* formulae into PNF.

Definition 3 Release-PNF of PCTL*. Let AP be a set of atomic propositions, the release-PNF of PCTL* state formulae Φ are defined as follows: $\Phi ::= \mathrm{true}|\mathrm{false}|a|\neg a|\Phi \wedge \Phi|\Phi \vee \Phi|\mathrm{P}_{\sim p}(\Psi)$, where $a \in AP$,Ψ is the path formula, $\sim \in \{\langle, \leq, \rangle, \geq\}$, and $p \in [0, 1]$ is the rational bound; the release-PNF of PCTL* path formulae Ψ are defined as follows: $\Psi ::= \Phi|\Psi \wedge \Psi|\Psi \vee \Psi|\mathrm{X}\Psi|\Psi\mathrm{U}\Psi|\Psi\mathrm{R}\Psi$, where the temporal modality R is dual to the until operator U. $\Psi_0\mathrm{R}\Psi_1$ is “true” over a path, if Ψ_1 always holds, a requirement that is released no sooner than Ψ_0 becomes valid.

3.3 Stochastic model checking LNPPN

The descriptive powers of PPN, NPPN, SPN and PTPN are the same with low-lever formal model DTMC, MDP, CTMC (Continuous-time Markov Chain) and CTMDP (Continuous-time Markov Decision Process), respectively. State-of-the-art stochastic model checker, e.g., PRISM, PAT, can verify LNPPN, Markov process or probabilistic automaton, indiscriminately. Actually, PCTL* stochastic model checking NPPN is a very complex process, as shown in Fig. 2, because the PCTL* is a combination of PCTL and LTL with probability. The time complexity of it is proved to be double exponential in $|\Psi|$ and polynomial in the size of NPPN M (Clarke et al. 2018; Liu et al. 2016), where Ψ is the path formula in PCTL* Φ. Liu et al. (2016) proposed a game (Shoham and Grumberg 2007)-based PCTL* stochastic model checking algorithm, which yielded a single exponential time complexity in $|\Psi|$ for PCTL* stochastic model checking LNPPN. They proved that this cannot be improved further, as the LTL with probability stochastic model checking LNPPN is PSPACE-completeness. Moreover, in reference Liu et al. (2016), the counterexample was generated by evidence of winning strategy of player refuter *refuter*. It is the first work to generate counterexamples for PCTL* stochastic model checking.

From the point of view of time complexity, abstraction for PCTL* stochastic model checking LNPPN is also very important and necessary. The model abstraction

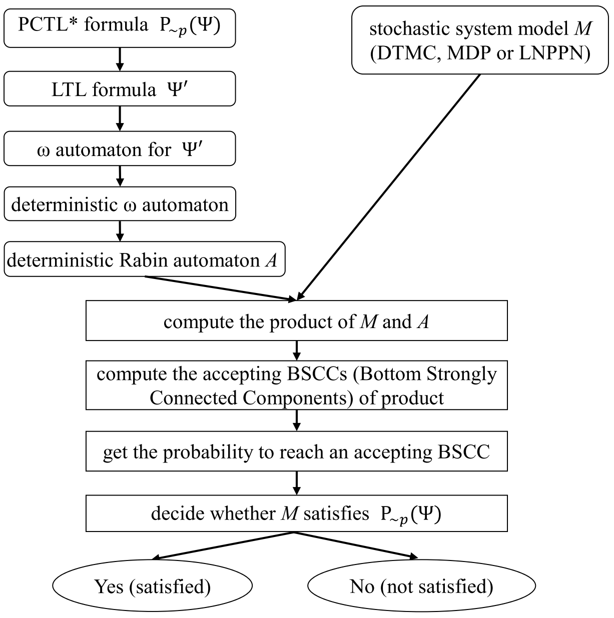
PCTL* formula $P_{\sim p}(\Psi)$
LTL formula Ψ'
ω automaton for Ψ'
deterministic ω automaton
deterministic Rabin automaton A
stochastic system model M (DTMC, MDP or LNPPN)
compute the product of M and A
compute the accepting BSCCs (Bottom Strongly Connected Components) of product
get the probability to reach an accepting BSCC
decide whether M satisfies $P_{\sim p}(\Psi)$
Yes (satisfied)
No (not satisfied)

Fig. 2 PCTL* stochastic model checking

methods in Sect. 2 can also be used to LNPPN, if they don't take the LNPPN as the concrete model, originally. It should be noted that there are currently no model abstraction methods to preserve full PCTL* properties for LNPPN.

4 Three-valued abstraction for LNPPN

In the fields of classical model checking, three-valued abstraction (Fecher et al. 2006; Shoham and Grumberg 2007) has been advocated as a better method for fighting state space explosion problem. It uses *may* and *must* transitions between abstract states to over- and under-approximate the concrete model. When lifting the three-valued abstraction to nondeterministic and probabilistic system model (i.e., LNPPN in this paper), it should be considered to handle the new nondeterminism and the probabilistic transition for over- and under-approximation in abstraction process.

5 Abstract model

We firstly present two kinds of extended LNPPN, which are used later for defining the abstract model of LNPPN.

Definition 4 Interval NPPN with Label (LINPPN). The interval LNPPN (LINPPN) can also be defined as a 7-tuple $M = (S, T; F, f; C; AP, L)$, where: (1) $T = \left(Transition, \left[Pt_l, Pt_u\right]\right)$, *Transition* denotes the transition act, $Pt_l, Pt_u \in [0, 1]$

denotes the success probability interval of the transition, and $Pt_l \leq Pt_u$; (2) $f = f_T \cup f_S \cup f_{S\times T} \cup f_{T\times S}$, $f_T : T \rightarrow 2^{Transition \times [Pt_l, Pt_u]}$, and $Pt_l, Pt_u : T \rightarrow [0,1]$ describe the lower and upper bounds for the success probabilities of T, $f_S = S \rightarrow [0,1]$, $f_{S\times T} = S \times T \rightarrow [0,1]$ describe the probabilities between places and transitions, the value of $f_{T\times S}$ can be obtained from the probability interval value of $f_T(t)$, i.e., $[\sigma_{Pt_l}(f_T(t)), \sigma_{Pt_u}(f_T(t))]$, the value of f_S except initial place can be computed according to the value of $f_{S\times T}$ and $f_{T\times S}$; (3) $\forall t \in T$, $\exists s \in S, f_{T\times S} = [1 - \sigma_{Pt_u}(f_T(t)), 1 - \sigma_{Pt_l}(f_T(t))].f_{T\times S}(t \times s) = [0,0]$, if the probability interval value of $f_T(t)$ is $[1,1]$, which represents transition (t,s) and place s are invalid; (4) C is the set of nondeterminism classes, and each nondeterminism class is a set comprised of (s, t_i) and $\sum \sigma_{Pt_l}(f_T(t_i)) \leq 1 \leq \sum \sigma_{Pt_u}(f_T(t_i))$. If $\{(s,t_1), (s,t_1) \ldots (s,t_1)\} \in C$, then $\sum_{i=1}^{n} f_{S\times T}(s, t_i) = 1$; (5) the others are the same with LNPPN.

An INPPN can be regarded as the extension of NPPN in transition probability, which permits the probability value of transition is the interval. The standard and complete LNPPN is a special case of LINPPN with $Pt_l = Pt_u$.

Definition 5 Game Probabilistic Petri Net with Label (LGPPN). A Game Probabilistic Petri Net with Label (LGPPN) is a tuple $M = (S, (S_0, S_1, S_p), T; F, f; AP, L)$ where: (1) (S_0, S_1, S_p) is a partition of S, and places in S_0, S_1 and S_p are called 'Player 0', 'Player 1' and 'Probabilistic' places, respectively; (2) if place $s \in S_0 \cup S_1$ (s is called player place), $f_{S\times T}(s,t) \in [0,1]$ for every $t \in T$, and if $s \in S_p$, $\sum_{t \in T} f_{S\times T}(s,t) = 1$; (3) the others are the same with LNPPN.

On the one hand, a LGPPN can be seen as the turn-based stochastic 2-player game played on NPPN. On the other hand, we can say that LGPPN extends the LNPPN with another non-deterministic choice among the enabled transitions in place s occurred. Moreover, either the non-deterministic choice has been made for reaching place s, or the non-deterministic choice has to be made after the enabled transitions in place s having been occurred. The standard and complete LNPPN can be defined as $(S, (S_0, \emptyset, S_p), T; F, f)$ in LGPPN form, which substitutes the set of nondeterminism classes C with $(S_0, \emptyset, S_p)$.

To contain more information in abstract model for both negative and affirmative results of three-valued PCTL* stochastic model checking, we extend both LGPPN and LINPPN orthogonally to present abstract model that is named LGIPPN. As defined in Definition 5, 'Player 0' making choices in S_0 corresponds to the new nondeterminism caused by abstraction, 'Player 1' making choices in S_1 corresponds to the original nondeterminism in LNPPN, lower bound and upper bound of the interval under- and over-approximate probabilistic transition, respectively.

Definition 6 Game Interval PPN with Label (LGIPPN). A LGIPPN can be defined as $M = (S, (S_0, S_1, S_p), T; F, f; AP, L)$ where: (1) (S_0, S_1, S_p) is a partition of$S$, and places in S_0, S_1 and S_p are called 'Player 0', 'Player 1' and 'Probabilistic' places, respectively; (2) $f = f_T \cup f_S \cup f_{S\times T} \cup f_{T\times S}$, $f_T : T \to 2^{Transition\times[Pt_l, Pt_u]}$, and $Pt_l, Pt_u : T \to [0,1]$ describe the lower and upper bounds for the success probabilities of T,$f_S = S \to [0,1]$, $f_{S\times T} = S\times T \to [0,1]$ describe the probabilities between places and transitions, the value of $f_{T\times S}$ can be obtained from the probability interval value of $f_T(t)$, i.e., $[\sigma_{Pt_l}(f_T(t)), \sigma_{Pt_u}(f_T(t))]$, the value of f_S except initial place can be computed according to the value of $f_{S\times T}$ and $f_{T\times S}$; (3) $\forall t \in T$, $\exists s \in S, f_{T\times S} = \left[1 - \sigma_{Pt_u}(f_T(t)), 1 - \sigma_{Pt_l}(f_T(t))\right]. f_{T\times S}(t \times s) = [0,0]$, if the probability interval value of $f_T(t)$ is$[1,1]$, which represents transition (t, s) and place s are invalid; (4) if place $s \in S_0 \cup S_1$ (s is called player place), $f_{S\times T}(s,t) \in \{0,1\}$ for every$t \in T$, if$s \in S_p$, $\sum_{t\in T} f_{S\times T}(s,t) = 1$, and for every$t_i \in s^{\cdot}$, $\sum \sigma_{Pt_l}(f_T(t_i)) \le 1 \le \sum \sigma_{Pt_u}(f_T(t_i))$ and $\sigma_{Pt_l}(f_T(t_i)) \le \sigma_{Pt_u}(f_T(t_i))$; (5) $L : S \times AP \to \{true, ?, flase\}$ is a labeling function that assigns a truth value in $\{true, ?, flase\}$ to each pair of *place* in S and proposition in AP; (6) the others are the same with LNPPN.

Note that we don't consider the place without any outgoing transition, which can be implemented by augmenting a transition to itself with probability 1. $L(s,a) = ?$ means that the truth value of an atomic proposition a is indefinite in the place s. Player 0, probability interval and the third value "?" are used to model explicitly a loss of information due to abstraction of place and probabilistic transition properties of the LNPPN, respectively. We call a LGIPPN model a LNPPN if $S_1 = \emptyset$, $Pt_l = Pt_u$, and there is no proposition taking value ? in any place. The relationship among LNPPN, ILNPPN, GLPPN and LGIPPN is shown in Fig. 3.

We formalize the notion of the 3-valued abstraction in Definition 7 on the level of interval LGIPPN, because: (1) the LNPPN is a special case of LGIPPN; (2) generally speaking, appropriate abstract model LGIPPN cannot be constructed at the first time, when abstraction is used after the first time, the concrete model is just LGIPPN which is constructed in abstract process at last time.

Definition 7 Three-valued abstraction of LGIPPN. Let $M = \left(S, \left(S_0, S_1, S_p\right), T; F, f; AP, L\right)$ be a concrete model and $\mathcal{P} = \left\{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_n\right\} \subseteq 2^S$ a partition of$S$, i.e., $\mathcal{P}_i \ne \emptyset$, $\mathcal{P}_i \cap \mathcal{P}_j = \emptyset$, for $i \ne j$, $1 \le i,j \le n$, and $\cup_{i=1}^n \mathcal{P}_i = S$. Then the three-valued abstract model $M^A = \left(S^A, \left(S_0^A, S_1^A, S_p^A\right), T^A; F^A, f^A; AP, L^A\right)$ of M induced by $\mathcal{P}$ can be defined as:

(1) $S^A = S_0^A \cup S_1^A \cup S_p^A$

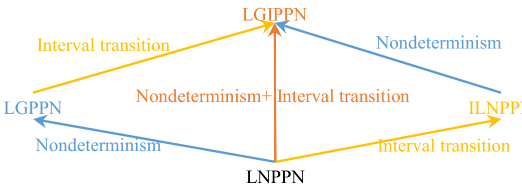

Fig. 3 Relationship between LNPPN, LGPPN, LINPPN and LGIPPN

(2) $S_0^A = \mathcal{P}$; S_1^A is composed of place s from S, whose next places reached by $s^{\cdot}$ are identical after lifting; S_p^A is set of probabilistic transitions emanating from each element of S_1^A, in the same style as M.

(3) For $\forall t_i^A$ between $(s_1^A)_m$ and $\left(s_0^A\right)_k$, $\sigma_{Pt_l}\left(f_{T^A}\left(t_i^A\right)\right) = \inf_{\cdot(t_j)\in(s_1^A)_m}\left(\sum_{(t_j)\cdot\in\left(s_0^A\right)_k} \sigma_{Pt_l}\left(f_T\left(t_j\right)\right)\right)$ and

$$\sigma_{Pt_u}\left(f_{T^A}\left(t_i^A\right)\right) = \min\left\{1,\ \sup_{\cdot(t_j)\in(s_1^A)_m}\left(\sum_{(t_j)\cdot\in\left(s_0^A\right)_k} \sigma_{Pt_u}\left(f_T\left(t_j\right)\right)\right)\right\}$$

(4) T^A is the set of t_i^A

(5) F^A and f^A can be got from S^A, $\left(S_0^A, S_1^A, S_p^A\right)$ and T^A according to Definition 5.

(6) $L^A\left((s_0^A)_n,\ a\right) = \begin{cases} true, & if\ L(s,a) = true\ for\ all\ s \in (s_0^A)_n \\ false, & if\ L(s,a) = false\ for\ all\ s \in (s_0^A)_n \\ ?, & otherwise \end{cases}$

(7) Initial place of M^A is $s_{init}^A = \mathcal{P}_i$, where initial place of M $s_{init} \in \mathcal{P}_i$. Initial distribution of M^A is $\mu_0^A\left((s_0^A)_n\right) = \sum_{s\in(s_0^A)_n} \mu_0(s)$.

Note that partition $\mathcal{P}$ is usually depending on where M are used. In this paper, partition $\mathcal{P}$ is based on invisible variables (Clarke et al. 2002) which means that an abstract place ($\mathcal{P}_i$) "agrees" on all the variables that are visible. Intuitively speaking, the Player 0 place is an element of the partition of places in concrete M. Player 0 selects a concrete place in the set $\mathcal{P}$ firstly, then Player 1 selects a probability distribution over the concrete t_i, which is a distribution over abstract places rather than concrete places.

Theorem 1 *For any LGIPPN M and partition $\mathcal{P}$, the three-valued abstract model M^A constructed by Definition 7 is also a LGIPPN.*

Proof It is obvious that S^A, $\left(S_0^A, S_1^A, S_p^A\right)$, F^A, f^A, AP and L^A are all in accordance with the definition of LGIPPN. To prove M^A is a LGIPPN, it is just to prove: (1) $\sigma_{Pt_l}\left(f_{T^A}\left(t_i^A\right)\right) \in [0,1]$, $\sigma_{Pt_u}\left(f_{T^A}\left(t_i^A\right)\right) \in [0,1]$, and $\sigma_{Pt_l}\left(f_{T^A}\left(t_i^A\right)\right) \le \sigma_{Pt_u}\left(f_{T^A}\left(t_i^A\right)\right)$; (2) $\sum \sigma_{Pt_l}\left(f_{T^A}\left(t_i^A\right)\right) \le 1 \le \sum \sigma_{Pt_u}\left(f_{T^A}\left(t_i^A\right)\right)$. The former obligation (1) follows by easy derivation from (3) of Definition 7. We show that obligation (2) holds: $\sum \sigma_{Pt_l}\left(f_{T^A}\left(t_i^A\right)\right) \sum_{(t_j)\cdot\in S_0^A}\left(\sum_{\cdot(t_j)\in(S_1^A)_m,(t_j)\cdot\in\left(S_0^A\right)_k} \sigma_{Pt_l}\left(f_T\left(t_j\right)\right)\right) \le 1$; $\sum \sigma_{Pt_u}\left(f_{T^A}\left(t_i^A\right)\right) = \sum_{(t_j)\cdot\in S_0^A}\left(\min\left\{1,\ \sup_{\cdot(t_j)\in(S_1^A)_m}\left(\sum_{(t_j)\cdot\in\left(S_0^A\right)_k} \sigma_{Pt_u}\left(f_T\left(t_j\right)\right)\right)\right\}\right) \ge \sum_{(t_j)\cdot\in S_0^A}\left(\sum_{\cdot(t_j)\in(S_1^A)_m,(t_j)\cdot\in\left(S_0^A\right)_k} \sigma_{Pt_u}\left(f_T\left(t_j\right)\right)\right) \ge 1$. So, M^A is a LGIPPN. □

5.1 Simulation relation on LGIPPN

To compare the behavior between LGIPPN model and its abstract model, we will discuss simulation relation on LGIPPN model. Simulation relations have been extensively studied both in classical model checking and stochastic model checking. The state-based simulation for probabilistic models (Jonsson and Larsen 1991) is a preorder on a state space. It requires that if state u simulates state v, u can mimic the stepwise behavior of v but may have more behavior that cannot be matched by u. This state-based simulation can be employed to distributions over places of LGIPPN model via weight function (Jonsson and Larsen 1991).

Definition 8 Distribution-based simulation. Let S, S' be the set of places in LGIPPN model, and $\mu \in distr(S)$, $\mu' \in distr\left(S'\right)$. For $R \subseteq S \times S'$, μ' simulates μ w.r.t. R, denoted $\mu R \mu'$, if there exists a weight function $\Delta : S \times S' \to [0,1]$ such that for all $u \in S, v \in S'$: (1) $\Delta(u,v) > 0 \Rightarrow uRv$, (2) $\sum\limits_{s' \in S'} \Delta\left(u,s'\right) = \mu(u)$, (3)$\sum\limits_{s \in S} \Delta(s,v) = \mu'(v)$

Distribution-based simulation can be computed by reducing them to a maximum-flow problem (Zhang et al. 2018). It plays an important role in defining simulation relation on LGIPPN model.

Definition 9 Probabilistic simulation. Let M, M' be LGIPPN models, μ_0, $\mu_0^{'}$ be the initial distributions. $R \subseteq S \times S'$ is a probabilistic simulation relation for $\left(M,M'\right)$, which holds that: (1) $\mu_0 R \mu_0^{'}$, (2) for $\forall\left(s,s'\right) \in R$: (2–1) $L'\left(s',a\right) \neq ? \Rightarrow L'\left(s',a\right) = L(s,a)$, (2–2) $s \to^{t} \mu$ implies $s' \to^{t'} \mu'$ such that $\mu R \mu'$, where $t = \left(Transition, \left[Pt_l, Pt_u\right]\right)$, $t' = \left(Transition, \left[Pt_l^{'}, Pt_u^{'}\right]\right)$

If there exists a probabilistic simulation relation for $\left(M,M'\right)$, we say that LGIPPN model M is simulated by M', denoted $M \preceq M'$.

Theorem 2 *Given a LGIPPN model M, the LGIPPN model M^A obtained by applying Definition 7 is such that $M \preceq M^A$.*

Proof In order to prove$M \preceq M^A$, we just show that $R = \{(s,\mathcal{P}_i) | s \in \mathcal{P}_i, \mathcal{P}_i \in \mathcal{P}\}$ is a probabilistic simulation in Definition 8. According to Definition 7-(7), $\mu_0^A\left((s_0^A)_n\right) = \sum_{s \in (s_0^A)_n} \mu_0(s)$, it directly follows $\mu_0 R \mu_0^A$ and fulfils condition (1) of Definition 8. It can also show that L^A fulfills condition (2–1) of Definition 8 by the Definition 7-(6). Let$\left(s, (s_0^A)_n\right) \in R$, $s \overset{t}{\to} \mu$ and$\mathcal{P}_i \overset{t^A}{\to} \mu^A$, then we construct a weight function Δ and show it fulfills the condition (2–2) of Definition 8, i.e., the condition (1) (2) (3) of Definition 7. (1) for $v \in \mathcal{P}$ let$\mu^A(v) = \sum_{u \in v} \mu(u)$, and for $u \in S$ let$\Delta(u,v) = \begin{cases} \mu(u) & if (u,v) \in R \\ 0 & otherwise \end{cases}$, function μ^A is a probability distribution if μ is:$\sum_{v \in \mathcal{P}} \mu^A(v) = \sum_{v \in \mathcal{P},\, u \in v} \mu(u) = \sum_{u \in S} \mu(u) = 1.$ Then for$v \overset{t^A}{\to} \mathcal{P}_i$, $\sigma_{Pt_l}(f_T(t^A)) \le \mu^A(v) \le \sigma_{Pt_u}(f_T(t^A))$ is hold according to Definition 6. Condition (1) of Definition 7 is fulfilled trivially, since $\Delta(u,v) = 0$ if $(u,v) \notin R$. (2) $\sum\limits_{\mathcal{P}_i \in \mathcal{P}} \Delta\left(u,\mathcal{P}_i\right) = \mu(u)$, condition (2) of Definition 7 is fulfilled.

(3)$\sum_{u \in S} \Delta(u, v) = \sum_{u \in v} \Delta(u, v) = \sum_{u \in v} \mu(u) = \mu^A(v)$, condition (3) of Definition 7 is fulfilled. So, $M \preccurlyeq M^A$. □

5.2 Preservation of PCTL*

When evaluating a PCTL* formula over the LGIPPN model, a place may no longer just satisfy or refuse the formula, i.e., "true" or "false", but indefinitely satisfy or refuse it, i.e., value "?". If a formula is evaluated to "true" or "false" in a place, we say that the result is definite. Otherwise, we say that it is indefinite. We generalize Kleene's strong 3-valued propositional logic (Paoli and Prabaldi 2020) to interpret propositional operators on the LGIPPN model. Operator conjunction $\sqcap$ is defined as follows: the operation value is "true", if both arguments of $\sqcap$ are "true"; the operation value is "false", if either of arguments is "false"; and the operation value is "?" (i.e., the value is indefinite), otherwise. Operator negation $\neg$ maps "true" to "false", "false" to "true", and "?" to "?". Operator disjunction $\bigsqcup$ can be derived from the operator $\sqcap$ with De Morgan's laws.

Given a LGIPPN model $M = \left(S,\ \left(S_0,\ S_1,\ S_p\right),\ T;\ F,\ f;\ AP,\ L\right)$ a PCTL* state formula Φ, the three-valued semantics of Φ in a place s, denoted as $\left[(M, s) \models_3 \Phi\right]$, can be defined inductively:

$$\left[(M, s) \models_3 \text{true}\right] = \text{true}$$

$$\left[(M, s) \models_3 a\right] = \begin{cases} \text{true} & \text{if } a \in L(\text{s}) \\ \text{false} & \text{if } \neg\, a \in L(s) \\ ? & \text{otherwise} \end{cases}$$

$$\left[(M, s) \models_3 \neg\Phi\right] = \begin{cases} \text{true} & \text{if } s \not\models_3 \Phi \\ \text{false} & \text{if } s \models_3 \Phi \\ ? & \text{otherwise} \end{cases}$$

$$\left[(M, s) \models_3 \Phi_0 \wedge \Phi_1\right] = \begin{cases} \text{true} & \text{if } \left[(M, s) \models_3 \Phi_0\right] = \text{true} \sqcap \left[(M, s) \models_3 \Phi_1\right] = \text{true} \\ \text{false} & \text{if } \left[(M, s) \models_3 \Phi_0\right] = \text{false} \sqcup \left[(M, s) \models_3 \Phi_1\right] = \text{false} \\ ? & \text{otherwise} \end{cases}$$

$$\left[(M,\ s) \models_3 \mathrm{P}_{\geq p}(\Psi)\right] = \begin{cases} \text{true} & \text{if } Prob_{\inf}(s,\ \Psi,\ \text{true}) \geq p \\ \text{false} & \text{if } Prob_{\sup}(s,\ \Psi,\ \text{false}) < p \\ ? & \text{otherwise} \end{cases}$$

$$\left[(M,s) \vDash_3 \mathrm{P}_{\leq p}(\Psi)\right] = \begin{cases} \text{true} & \text{if } Prob_{\sup}(s,\Psi,\text{false}) \leq p \\ \text{false} & \text{if } Prob_{\inf}(s,\Psi,\text{true}) > p \\ ? & \text{otherwise} \end{cases}$$

where $Prob_{\inf}(s,\Psi,\text{true})$ is the probability measure of path set such that $\pi \vDash \Psi$, i.e., $Prob(s,\ \Psi)= Pr_s(\{\pi \in Path(s)\ |\pi \vDash \Psi\})$, $Path(s)$ is the path set which starts with place$s$. Case $\left[(M,s) \vDash_3 \Phi_0 \vee \Phi_1\right]$ can be derived from $\left[(M,s) \vDash_3 \Phi_0 \wedge \Phi_1\right]$ according to De Morgan's laws. Cases $\left[(M,s) \vDash_3 \mathrm{P}_{>p}(\Psi)\right]$ and $\left[(M,s) \vDash_3 \mathrm{P}_{<p}(\Psi)\right]$ are similar to the cases $\left[(M,s) \vDash_3 \mathrm{P}_{\geq p}(\Psi)\right]$ and $\left[(M,s) \vDash_3 \mathrm{P}_{\leq p}(\Psi)\right]$, respectively, but we exchange $\geq$ by $>$ and vice versa. For a path $\pi = s_0 \rightarrow^{t_0} s_1 \rightarrow^{t_1} s_2 \rightarrow^{t_2} s_3 \ldots$ (abbr.$\pi = s_0 s_1 s_2 s_3 \ldots$, if it is not related to$t$), the three-valued semantics of a path formula Ψ onπ, denoted $\left[(M,\pi) \vDash_3 \Psi\right]$, is defined inductively as follows:

$$\left[(M,\pi) \vDash_3 \Phi\right] = \begin{cases} \left[\left(M,s_0\right) \vDash_3 \Phi\right] & \text{if } |\pi| > 0 \\ ? & \text{otherwise} \end{cases}$$

$$\left[(M,\pi) \vDash_3 \mathrm{X}\Psi\ \right] = \begin{cases} \left[\left(M,s_1\right) \vDash_3 \Psi\right] & \text{if } |\pi| > 1 \\ ? & \text{otherwise} \end{cases}$$

$$\left[(M,\pi) \vDash_3 \neg\Psi\right] = \begin{cases} \text{true} & \text{if } \pi \nvDash_3 \Psi \\ \text{false} & \text{if } \pi \vDash_3 \Psi \\ ? & \text{otherwise} \end{cases}$$

$$\left[(M,\pi) \vDash_3 \Psi_0 \wedge \Psi_1\right] = \begin{cases} \text{true} & \text{if } \left[(M,\pi) \vDash_3 \Psi_0\right] = \text{true} \sqcap \left[(M,\pi) \vDash_3 \Psi_1\right] = \text{true} \\ \text{false} & \text{if } \left[(M,\pi) \vDash_3 \Psi_0\right] = \text{false} \sqcup \left[(M,\pi) \vDash_3 \Psi_1\right] = \text{false} \\ ? & \text{otherwise} \end{cases}$$

$$\left[(M,\pi) \vDash_3 \Psi_0 \mathrm{U} \Psi_1\ \right] = \begin{cases} \text{true} & \begin{array}{l} \text{if } \exists\, 0 \leq k < |\pi| : [(\left[\left(M,s_k\right) \vDash_3 \Psi_1\right] = \text{true}) \sqcap \\ (\forall j < k : \left[\left(M,s_j\right) \vDash_3 \Psi_0\right] = \text{true})] \end{array} \\ \text{false} & \begin{array}{l} \text{if } (\forall\, 0 \leq k < |\pi| : ([(\forall j < k : \left[\left(M,s_j\right) \vDash_3 \Psi_0\right] \neq \text{false}) \\ \Rightarrow (\left[\left(M,s_k\right) \vDash_3 \Psi_1\right] = \text{false})])\sqcap \\ ((\forall 0 \leq k < |\pi| : \left[\left(M,s_k\right) \vDash_3 \Psi_0\right] \neq \text{false}) \Rightarrow |\pi| = \infty) \end{array} \\ ? & \text{otherwise} \end{cases}$$

The other path operators can be derived from above cases.

If $\forall s_0 \in S_0 : \left[\left(M,s_0\right)\vDash_3\Phi\right] = \text{true}$, we can conclude that $\left[M\vDash_3\Phi\right] = \text{true}$, i.e., $M\vDash_3\Phi$. If $\exists s_0 \in S_0 : \left[\left(M,s_0\right)\vDash_3\Phi\right] = \text{false}$, we can conclude that $\left[M\vDash_3\Phi\right] = \text{false}$, i.e., $M\nvDash_3\Phi$. $\left[M\vDash_3\Phi\right] = ?$, otherwise. Formally, the three-valued semantics of PCTL* characters a preorder relation over a LGIPPN model that reflects the degree of completeness. Let $\lesssim$ denote an information ordering over the truth values, in which $? \lesssim \text{true}$, $? \lesssim \text{false}$, and $x \lesssim x$ $(x \in \{\text{true},\text{false},?\})$. The operators *comp*, *min* and *max* are monotonic on$\lesssim$, i.e., if $x_1 \lesssim x_2$ and $y_1 \lesssim y_2$, then $comp(x_1) \lesssim comp(x_2)$, $min(x_1) \lesssim min(x_2)$ and $max(x_1) \lesssim max(x_2)$. We can conclude that $[(M^A, s^A_{init}) \models_3 \Phi] \lesssim [(M, s_{init}) \models_3 \Phi]$. Informally, if a LGIPPN model is more abstract with respect to $\preccurlyeq$, it has less definite properties that are either "true" or "false" with respect to $\lesssim$. Moreover, a formula of PCTL* is evaluated to "true" of "false" on an abstract LGIPPN model, then it has the same truth value on any refinement models. In other words, the result of three-valued PCTL* stochastic model-checking an abstract LGIPPN model agrees with the concrete model, if the results are definite. This is asserted in Theorem 3.

Theorem 3 *Let $R \subseteq S \times S^A$ be a mixed probabilistic simulation relation from a LGIPPN model M to a LGIPPN model M^A, i.e., $M \preccurlyeq M^A$. Then for all PCTL* formulas Φ:$\left[M^A \models_3 \Phi\right] \neq$?implies$\left[M^A \models_3 \Phi\right] = [M \models \Phi]$.*

Proof $\left[M^A \models_3 \Phi\right] = [(M^A, s^A_{init}) \models_3 \Phi]$ and $[M \models \Phi] = [(M, s_{init}) \models_3 \Phi]$, so, we can prove the following proposition for proving it. For every $(s, s^A) \in R$ and all PCTL* formulas Φ: $\left[\left(M^A, s^A\right) \models_3 \Phi\right] \neq ?$ implies $\left[\left(M^A, s^A\right) \models_3 \Phi\right] = [(M, s) \models \Phi]$. It can be proved by the induction method: (1) $\Phi = \text{true}$: $\left[\left(M^A, s^A\right) \models_3 \text{true}\right] = \text{true} = [(M, s) \models \text{true}]$, since $s \preccurlyeq s^A$. (2) PCTL* formulae Φ is atomic proposition a: $\left[\left(M^A, s^A\right) \models_3 a\right] = L^A\left(s^A, a\right) = L(s, a) = [(M, s) \models a]$. Induction hypothesis: suppose Φ' represents all sub-formulas of Φ, $\left[\left(M^A,\ s^A\right) \models_3 \Phi'\right] \neq ?$ implies that $\left[\left(M^A,\ s^A\right) \models_3 \Phi'\right] = \left[(M, s) \models \Phi'\right]$. (3) $\Phi = \neg\Phi'$: $\left[\left(M^A,\ s^A\right) \models_3 \neg\Phi'\right] = \neg\left[\left(M,\ s^A\right) \models_3 \Phi'\right] = \neg\left[(M, s) \models \Phi'\right] = \left[(M, s) \models \neg\Phi'\right]$. (4) $\Phi = \Phi'_0 \wedge \Phi'_1$: $\left[\left(M^A, s^A\right) \models_3 \Phi'_0 \wedge \Phi'_1\right] = \left[\left(M^A, s^A\right) \models_3 \Phi'_0\right] \sqcap \left[\left(M^A, s^A\right) \models_3 \Phi'_1\right] = \left[(M, s) \models \Phi'_0\right] \wedge \left[(M, s) \models \Phi'_1\right] = \left[(M, s) \models \Phi'_0 \wedge \Phi'_1\right]$. (5) $\Phi = \mathrm{P}_{\leq p}(\Psi)$, where $\Psi = \Phi'/X\Psi'/\neg\Psi'/\Psi_0 \wedge \Psi_1/\Psi_0 \mathrm{U} \Psi_1$: if $\left[\left(M^A, s^A\right) \models_3 \Phi\right] = \text{true} \Rightarrow Prob_{\sup}(s, \Psi, \text{true}) \leq Prob_{\sup}\left(s^A, \Psi, \text{true}\right) \leq p \Rightarrow [(M, s) \models \Phi] = \text{true}$; if $\left[\left(M^A, s^A\right) \models_3 \Phi\right] = \text{false} \Rightarrow Prob_{\inf}(s, \Psi, \text{true}) \geq Prob_{\inf}\left(s^A, \Psi, \text{true}\right) > p \Rightarrow [(M, s) \models \Phi] = \text{false}$. (6) $\Phi = \mathrm{P}_{\geq p}(\Psi)$, where $\Psi = \Phi'/X\Psi'/\neg\Psi'/\Psi_0 \wedge \Psi_1/\Psi_0 \mathrm{U} \Psi_1$: if $\left[\left(M^A, s^A\right) \models_3 \Phi\right] = \text{true} \Rightarrow Prob_{\inf}(s, \Psi, \text{true}) \geq Prob_{\inf}\left(s^A, \Psi, \text{true}\right) \geq p \Rightarrow [(M, s) \models \Phi] = \text{true}$; if $\left[\left(M^A, s^A\right) \models_3 \Phi\right] = \text{false} \Rightarrow Prob_{\sup}(s, \Psi, \text{true}) \leq Prob_{\sup}\left(s^A, \Psi, \text{true}\right) < p \Rightarrow [(M, s) \models \Phi] = \text{false}$. □

The above theorem states that our abstraction framework can indeed be used as a solution to MA4SMC problem, which constructs three-valued abstract model according to Definition 7, and preserves the full PCTL* properties. Intuitively speaking, in this framework, the results of PCTL* stochastic model-checking an abstract model agree with PCTL* stochastic model-checking a concrete model, unless the result is indefinite.

6 Three-valued stochastic model checking

Given a three-valued abstract model (LGIPPN model M^A) and a PCTL* formula Φ in release-PNF, deciding whether $M^A \models_3 \Phi$, i.e., $\left(M^A, s^A_{init}\right) \models_3 \Phi$, holds or not, is named the three-valued PCTL* stochastic model checking. Theoretically speaking, it can be reduced to two instances of the traditional PCTL* stochastic model checking, as the three-valued model checking to traditional model checking. In this section, we give a direct game-based three-valued PCTL* stochastic model checking algorithm.

6.1 Game semantics for three-valued stochastic model checking

The game-based operational semantics for PCTL* stochastic model checking is an intuitive and succinct approach for stochastic model checking, and it can provide the succinct evidences for corresponding results. In this section, we generate it to three-valued PCTL* stochastic model checking.

The game $\mathrm{G}^{\Phi}_{M^A}(player, board, rule)$ for a three-valued abstract model (LGIPPN model M^A) and a PCTL* formula Φ in release-PNF can be defined as in Liu et al. (2016). However, some move rules of the player and the winning criteria need to be changed. The differences are caused by the fact that LGIPPN model has the interval transitions. Since the transitions are considered only in configurations with sub-formulae of $\mathrm{P}_{\geq p}(\mathrm{X}\Psi)$ or $\mathrm{P}_{\leq p}(\mathrm{X}\Psi)$, the new move rules for game consist of move rules in Liu et al. (2016), with exception that rule (6) are adapted as follows:

(0) $p = 0$, the play finishes and player *verifier* wins.
(1) $Con_i = \left(player, s, \text{true}/\text{false}/a/\neg a/\mathrm{P}_{\geq p}(\text{true}/\text{false}/a/\neg a), \Omega\right)$, the play finishes.
(2) $Con_i = \left(verifier, s, \Phi_0 \wedge \Phi_1, \Omega\right)$, player *refuter* chooses $\Phi_j, j \in \{0, 1\}$, and $Con_{i+1} = \left(verifier, s, \Phi_j, \left\{\Phi_{1-j}\right\} \cup \Omega\right)$.
(3) $Con_i = \left(verifier, s, \Phi_0 \vee \Phi_1, \Omega\right)$, player *verifier* chooses $\Phi_j, j \in \{0, 1\}$, and $Con_{i+1} = \left(verifier, s, \Phi_j, \Omega\right)$.
(4) $Con_i = \left(refuter, s, \Phi_0 \wedge \Phi_1, \Omega\right)$, player *refuter* chooses$\Phi_j, j \in \{0, 1\}$, and$Con_{i+1} = \left(refuter, s, \Phi_j, \Omega\right)$.
(5) $Con_i = \left(refuter, s, \Phi_0 \vee \Phi_1, \Omega\right)$, player *verifier* chooses$\Phi_j, j \in \{0, 1\}$, and$Con_{i+1} = \left(refuter, s, \Phi_j, \left\{\Phi_{1-j}\right\} \cup \Omega\right)$.
(6) $Con_i = \left(verifier, s^A, \mathrm{P}_{\geq p}(\mathrm{X}\Psi), \Omega\right)$: player *verifier* chooses some transitions t^A in the minimum nondeterministic class by an adversary, and $Con_{i+1} = \left(verifier, s^{A\prime}, \mathrm{P}_{\geq ps^{A\prime}}(\Psi)\right)$, where $s^A \rightarrow^{t^A} s^{A\prime}$, and where $\sum P_l\left(s^A, s'\right) \cdot ps' \geq p$ where $P_l\left(s^A, s'\right)$ is the lower probability from the place s^A to $s^{A\prime}$ and $ps^{A\prime}$ is the probability of $s^{A\prime}$ satisfies Ψ; or player *refuter* chooses some transitions t^A in the maximum nondeterministic class by an adversary, and $Con_{i+1} = \left(refuter, s^{A\prime}, \mathrm{P}_{<ps^{A\prime}}(\Psi)\right)$, where $s^A \rightarrow^{t} s^{A\prime}$, and where $\sum P_u\left(s^A, s^{A\prime}\right) \cdot ps^{A\prime} \geq p$ where $P_u\left(s^A, s^{A\prime}\right)$ is the upper probability from place s^A to $s^{A\prime}$ and $ps^{A\prime}$ is the probability of $s^{A\prime}$ satisfies Ψ.
(7) $Con_i = \left(verifier, s, \mathrm{P}_{\geq p}\left(\Psi_1 \wedge \Psi_2\right), \Omega\right)$, player *refuter* chooses $\Psi_j, j \in \{0, 1\}$, and $Con_{i+1} = (refuter, s, \mathrm{P}_{\geq pj}\left(\Psi_j\right))$, and $p0 + p1 - 1 < p$.
(8) $Con_i = \left(verifier, s, \mathrm{P}_{\geq p}\left(\Psi_0 \vee \Psi_1\right), \Omega\right)$, player *verifier* chooses Ψ_j, and $Con_{i+1} = \left(verifier, s, \mathrm{P}_{\geq pj}\left(\Psi_j\right)\right)$, and $p0 + p1 \geq p$.
(9) $Con_i = \left(verifier, s, \mathrm{P}_{\geq p}\left(\Psi_0 \mathrm{U}\, \Psi_1\right), \Omega\right)$
:$Con_{i+1} = \left(verifier, s, \mathrm{P}_{\geq p}\left(\Psi_1 \vee \left(\Psi_0 \wedge X\left(\Psi_0 \mathrm{U}\, \Psi_1\right)\right)\right)\right)$.

(10) $Con_i = \left(verifier, s, P_{\geq p}\left(\Psi_0 R \Psi_1\right), \Omega\right)$ $:Con_{i+1} = \left(verifier, s, \mathrm{P}_{\geq p}\left(\Psi_1 \wedge \left(\Psi_0 \vee \mathrm{X}\left(\Psi_0 \mathrm{R} \Psi_1\right)\right)\right)\right)$.

(11) $Con_i = (player, s, \Phi, \{\mathrm{P}_{\geq p}(\Psi)\} \cup \Omega)$ $:Con_{i+1} = (player, s, \Phi, \Omega)$.

(12) $Con_i = (player, s, \Phi, \{\mathrm{true/false}/a/\neg a\} \cup \Omega)$ $:Con_{i+1} = (player, s, \Phi, \Omega)$.

(13) $Con_i = \left(verifier, s, \Phi_0, \left\{\Phi_1\right\} \cup \Omega\right)$, player *refuter* chooses, and $Con_{i+1} = \left(verifier, s, \Phi_1, \left\{\Phi_0\right\} \cup \Omega\right)$.

(14) $Con_i = \left(refuter, s, \Phi_0, \left\{\Phi_1\right\} \cup \Omega\right)$, player *verifier* chooses, and $Con_{i+1} = \left(refuter, s, \Phi_1, \left\{\Phi_0\right\} \cup \Omega\right)$.

For $Con_i = \left(verifier, s, \mathrm{P}_{\leq p}(\mathrm{X}\Psi), \Omega\right)$, the move rule can be defined analogously. Intuitively speaking, each player can use both lower and upper bound of transition, and the players use lower bond of transition in order to win, while they use upper bound of transition in order to prevent the other player from wining. Therefore, the new winning criteria are:

(1) The *verifier* wins the play if and only if one of the following conditions holds: (a) the play ends with rule (0); (b) the play ends with rule (1), and the configuration is $(player, s, \mathrm{true}/\mathrm{P}_{\geq p}(\mathrm{true}), \Omega)$, or $(player, s, a/\neg a/\mathrm{P}_{\geq p}(a/\neg a), \Omega)$ and $a/\neg a \in L(s)$; (c) the play iterates infinitely with rule (10); (d) the rule (13) is used for second time.

(2) The *refuter* wins the play if and only if one of the following conditions holds: (a) the play ends with rule (0); (b) the play ends with rule 1), and the configuration is $(player, s, \mathrm{false}/\mathrm{P}_{\geq p}(\mathrm{false}), \Omega)$, or $(player, s, a/\neg a/\mathrm{P}_{\geq p}(a/\neg a), \Omega)$ and $a/\neg a \notin L(s)$; (c) the play infinitely with rule (9); (d) the rule (14) is used for second time.

(3) Either *verifier* or *refuter* cannot win the play, and the play ends with a tie.

With the winning criteria in $\mathrm{G}^{\Phi}_{M^A}$, we can capture the operational semantics for three-valued PCTL* stochastic model checking.

Theorem 4 *Let M^A be a three-valued abstract model (LGIPPN), Φ a PCTL* formula Φ in release-PNF and $s^A \in S^A$. Then, for $\forall s$: $\left[\left(M^A, s^A\right) \models_3 \Phi\right] = true$ if and only if verifier has a winning strategy for game $G^{\Phi}_{M^A}$ with start configuration $(player, s^A, \Phi, \Omega)$; $\left[\left(M^A, s^A\right) \models_3 \Phi\right] = false$ if and only if refuter has a winning strategy for game $G^{\Phi}_{M^A}$ with start configuration $(player, s^A, \Phi, \Omega)$; $\left[\left(M^A, s^A\right) \models_3 \Phi\right] = ?$ if and only if either of the players has a winning strategy with start configuration $(player, s^A, \Phi, \Omega)$. In addition, it is independent of initial player which is verifier or refuter.*

Proof The "if" part is obvious, it is sufficient to prove the "only if" part which can be done by constructing the winning strategy of player *verifier* or *refuter* for $\left[\left(M^A, s^A\right) \models_3 \Phi\right] = \mathrm{true}$ or $\left[\left(M^A, s^A\right) \models_3 \Phi\right] = \mathrm{false}$, as for $\left[\left(M^A, s^A\right) \models \Phi\right] = \mathrm{true}$ or $\left[\left(M^A, s^A\right) \models \Phi\right] = \mathrm{false}$ in game for stochastic model checking. So, we just show the result for the truth value "?".

(1) $\Phi = \mathrm{true}/\mathrm{false}/a/\neg a/\mathrm{P}_{\geq p}(\mathrm{true}/\mathrm{false}/a/\neg a)$: if$\left[\left(M^A, s^A\right)\models_3 \Phi_0 \wedge \Phi_1\right] = ?$, every play ends with a tie, and either of them has a winning strategy.

(2) $\Phi = \Phi_0 \wedge \Phi_1$: if $\left[\left(M^A, s^A\right)\models_3 \Phi_0 \wedge \Phi_1\right] = ?$, then, according to the denotational semantics, $\left[\left(M^A, s^A\right) \models_3 \Phi_j\right] = ?$ or false at least for one of $j \in \{0,1\}$. So, by the induction hypothesis, whatever strategies *verifier* (or *refuter*) chooses, *refuter* (or *verifier*) can always choose to proceed to (*player*, s^A, Φ_j, Ω) in which *verifier* (or *refuter*) has no winning strategy.

(3) $\Phi = \Phi_0 \vee \Phi_1$: if $\left[\left(M^A, s^A\right)\models_3 \Phi_0 \vee \Phi_1\right] = ?$, according to the denotational semantics, $\left[\left(M^A, s^A\right)\models_3 \Phi_j\right] = ?$ at least for one of $j \in \{0,1\}$, and $\left[\left(M^A, s^A\right)\models_3 \Phi_j\right] = ?$ or false for the one $k \in \{0,1\}$ $k \neq j$. So, by the induction hypothesis, whatever strategies *verifier* (or *refuter*) chooses, *refuter* (or *verifier*) can always choose to proceed to (*player*, s^A, Φ_j, Ω) in which *verifier* (or *refuter*) has no winning strategy.

(4) $\Phi = \mathrm{P}_{\geq p}(\mathrm{X}\Psi)$: if$\left[\left(M^A, s^A\right)\models_3 \mathrm{P}_{\geq p}(\mathrm{X}\Psi)\right] = ?$, according to the denotational semantics, the value of $Pr_{s^A}\{\pi \in Path(s^A) | \pi \models \mathrm{X}\Psi\} \geq p$ is ? or false, i.e., the value of $\mathrm{Pr}_{s^A}\{\pi \in Path\left(s^A\right) | \pi \models \mathrm{X}\Psi\} \geq p$ is ?$_{t^A}$or false, where $P(s^A, s^{A\prime})$ is the lowest probability from the place s^A to $s^{A\prime}$, $s^A \to s^{A\prime}$ and $ps^{A\prime}$ is the probability of $s^{A\prime}$ satisfying Ψ, and the value of $\mathrm{Pr}_{s^A}\left\{\pi \in Path\left(s^A\right) | \pi \nvDash X\Psi\right\} > 1 - p$ is ? or false, i.e., the value of $\sum P\left(s^A, s^{A\prime\prime}\right) \cdot ps^{A\prime\prime} > 1 - p$ is ?$_{t^A}$or false, where $P(s^A, s^{A\prime\prime})$ is the lowest probability from the place s^A to $s^{A\prime\prime}$, $s^A \to s^{A\prime\prime}$ and $ps^{A\prime\prime}$ is the probability of $s^{A\prime\prime}$ satisfying $\neg\Psi$. So, by the induction hypothesis, whatever strategies *verifier* (or *refuter*) chooses, *refuter* (or *verifier*) can always choose to proceed to (*player*, s^A, $\mathrm{P}_{\geq p}(\mathrm{X}\Psi)$, Ω) in which *verifier* (or *refuter*) has no winning strategy.

(5) $\Phi = \mathrm{P}_{\geq p}(\Psi_0 \wedge \Psi_1)$: if $\left[\left(M^A, s^A\right)\models_3 \mathrm{P}_{\geq p}(\Psi_0 \wedge \Psi_1)\right] = ?$, according to the denotational semantics, the value of $Pr_{s^A}\{\pi \in Path(s^A) | \pi \models \Psi_0 \wedge \Psi_1\} \geq p$ is ? under the lowest probability, and value of $Pr_{s^A}\{\pi \in Path(s^A) | \pi \models \mathrm{X}\Psi\} \geq p$ is true or ? under the upperest probability. So, by the induction hypothesis, whatever strategies *verifier* (or *refuter*) chooses, *refuter* (or *verifier*) can always choose to proceed to (*player*, s^A, $\mathrm{P}_{\geq p}(\Psi_0 \wedge \Psi_1)$, Ω) in which *verifier* (or *refuter*) has no winning strategy.

(6) $\Phi = \mathrm{P}_{\geq p}(\Psi_0 \vee \Psi_1)$: if $\left[\left(M^A, s^A\right)\models_3 \mathrm{P}_{\geq p}(\Psi_0 \vee \Psi_1)\right] = ?$, according to the denotational semantics, the value of $Pr_{s^A}\{\pi \in Path(s^A) | \pi \models \Psi_0 \vee \Psi_1\} \geq p$ is ? under the lowest probability, and value of $Pr_{s^A}\{\pi \in Path(s^A) | \pi \models \mathrm{X}\Psi\} \geq p$ is true or ? under the upperest probability. So, by the induction hypothesis, whatever strategies *verifier* (or *refuter*) chooses, *refuter* (or *verifier*) can always choose to proceed to (*player*, s^A, $\mathrm{P}_{\geq p}(\Psi_0 \vee \Psi_1)$, Ω) in which *verifier* (or *refuter*) has no winning strategy.

(7) $\Phi = \mathrm{P}_{\geq p}(\Psi_0\ \mathrm{U}\ \Psi_1)$: if $\left[\left(M^A, s^A\right)\models_3 \mathrm{P}_{\geq p}(\Psi_0 \mathrm{U} \Psi_1)\right] = ?$, according to the denotational semantics, the value of $Pr_{s^A}\{\pi \in Path\ (s^A) | \pi \models \Psi_0\ \mathrm{U}\ \Psi_1\} \geq p$ is ? under the lowest probability, and value of $Pr_{s^A}\{\pi \in Path\ (s^A) | \pi \models \mathrm{X}\ \Psi\} \geq p$ is true or ? under the upperest probability. So, by the induction hypothesis, whatever strategies *verifier* (or *refuter*) chooses, *refuter* (or *verifier*) can always choose to proceed to (*player*, s^A, $\mathrm{P}_{\geq p}(\Psi_0 \mathrm{U}\ \Psi_1)$, Ω) in which *verifier* (or *refuter*) has no winning strategy.

(8) $\Phi = \mathrm{P}_{\geq p}(\Psi_1\ \mathrm{R}\ \Psi_2)$: if$\left[\left(M^A, s^A\right)\models_3 \mathrm{P}_{\geq p}(\Psi_0\ \mathrm{R}\ \Psi_1)\right] = ?$, according to the denotational semantics, the value of $Pr_{s^A}\{\pi \in Path\ (s^A) | \pi \models \Psi_0\ \mathrm{R}\ \Psi_1\} \geq p$ is ? under

the lowest probability, and value of $Pr_{s^A}\{\pi \in Path\ (s^A)|\pi \vDash \Psi_0 \mathrm{R}\ \Psi_1\} \geq p$ is true or ? under the upperest probability. So, by the induction hypothesis, whatever strategies *verifier* (or *refuter*) chooses, *refuter* (or *verifier*) can always choose to proceed to $(player, s^A, \mathrm{P}_{\geq p}(\Psi_0 \mathrm{R}\ \Psi_1),\ \Omega)$ in which *verifier* (or *refuter*) has no winning strategy.

The Theorem 4 states three-valued PCTL* stochastic model checking is equivalent to two-player stochastic game $\mathrm{G}^{\Phi}_{M^A}(player, board, rule)$. If the player *verifier* has a winning strategy from the initial place s^A_{init} of M^A, then M^A satisfies Φ, i.e., $M^A \vDash_3 \Phi$, and the winning strategy can be served as the evidence for $M^A \vDash_3 \Phi$. If the player *refuter* has a winning strategy from the initial place s^A_{init} of M^A, then M^A does not satisfy Φ, i.e., $M^A \nvDash_3 \Phi$, and the winning strategy can be served as the evidence for $M^A \nvDash_3 \Phi$, i.e., the counterexample. If either of them has the winning strategy, then the result is indefinite, which means the abstract model should be refined.

6.2 Strategy solving in three-valued stochastic game

A two-player game process $G^{\Phi}_{M^A}$ for three-valued PCTL* stochastic model checking can be presented by the game-graph $Gg(N, E, w)$. It can be constructed from initial configuration as the initial node in a BFS (breadth first search) or DFS (depth first search) manner, and owns the same characteristics (Liu et al. 2016): game-graph can be partitioned into some MSCCs (maximal strongly connected components), and every play never leaves a MSCC_m into a MSCC_n with $m < n$.

The game $G^{\Phi}_{M^A}$ for three-valued PCTL* stochastic model checking is a three-valued stochastic game, essentially. We implement a three-valued coloring method for strategy solving, which can alter the coloring process (Liu et al. 2016; Shoham and Grumberg 2007) of traditional PCTL* stochastic model checking game and CTL model checking game. It colors each node in the MSCCs of game-graph $Gg(N, E, w)$, and processes MSCC_i according to i bottom-up. Let MSCC_i be the smallest MSCC at present, i.e., the other MSCC_m with $m < i$ have all been colored, the three-valued coloring rules of a node in MSCC_i is as follows: for a node with a current Φ' of subformula Φ, (1) if player *verifier* can win for current Φ', the corresponding node is colored white; (2) if player *refuter* can win for current Φ', the corresponding node is colored black; (3) if either of the players can win for current Φ', the corresponding node is colored gray.

We color all the nodes of the game-graph $Gg(N, E, w)$ according to the coloring rules. If the initial node is colored white, player *verifier* wins the game, and all the white nodes compose the winning strategy of *verifier*. This means $M^A \vDash_3 \Phi$ according to Theorem 4, and $M \vDash \Phi$ according to Theorem 3. If the initial node is colored black, player *refuter* wins the game, and all the black nodes compose the winning strategy of *refuter*. This means $M^A \nvDash_3 \Phi$ according to Theorem 4, and $M \nvDash \Phi$ according to Theorem 3. If the initial node is colored gray, either of players wins the game. This means the result of three-valued PCTL* stochastic model checking M^A is indefinite, according to Theorem 4; and the result of PCTL* stochastic model checking

M is also indefinite, according to Theorem 3. The gray nodes need to be refined, which is presented in Sect. 6.

Strategy solving algorithm for three-valued PCTL* stochastic model checking game can be summarized into three steps: (1) construct the game-graph $Gg(N, E, w)$ for M^A and Φ; (2) find the MSCCs and sort them bottom-up, (3) three-valued color the nodes of game-graph with the MSCCs' order, according to the three-valued coloring rules; (4) determine the color of the initial node and return the corresponding winning strategy in DFS manner. It is the combination of searching MSCCs with order algorithm and three-valued coloring process, and the complexity of which is PSPACE. It can be proved similarly with stochastic model checking game (Liu et al. 2016; Kwiatkowska et al. 2021; Shoham and Grumberg 2007).

The winning strategy of *verifier* or *refuter* is the evidence whether PCTL* Φ is satisfied or not, respectively. An evidence of *refuter*, i.e., the black nodes of the three-valued colored game-graph, can be served as a kind of counterexample. It should be noted that the ideal counterexample(Abraham et al. 2014) just contains the smallest set of necessary nodes from the winning strategy of *refuter*.

7 Refinement

If the three-valued stochastic model checking returns an indefinite result, i.e., "?", it means the abstract place is so coarse that we can say nothing from it and have to refine the abstract place. Roughly, refining an invalid abstract place is to split it into the smaller abstract places until the three-valued stochastic model checking result is definite on it. We can exploit information of the colored three-valued game-graph in refinement process.

7.1 Identifying and analyzing failure nodes

Definition 10 Failure node. A node in colored three-valued game-graph is the failure node, if it is colored gray, and none of its sons was colored gray when it is colored.

Intuitively speaking, a failure node is responsible for the loss of information in abstraction. Thus, it should be refined. If the initial node is colored gray, a failure node can be found by the identifying failure nodes algorithm. For each node, it is colored gray and is not a failure node, the coloring algorithm is adapted to remember that a son was colored gray when n is colored, denoted as $go(n)$. Identifying

```
FNIdentify ($n$){
    If $n$ satisfies the definition of failure node, return $n$
    Else FNIdentify($go(n)$)
}
```

Fig. 4 Identifying failure nodes algorithm

failure nodes algorithm in Fig. 4 can identify the failure node from $go(n)$, which starts from the initial node.

Theorem 5 *The identifying failure nodes algorithm can terminate.*

Proof If the current node n is not a failure node, the algorithm continues to identify at $go(n)$. According to Definition 10 (failure node), there is a node that was colored gray before n. Thus, each recursive call is applied on the node colored gray earlier. So, the number of recursive calls is not bigger than the run time of the coloring algorithm that is finite. □

Failure nodes identifying (FNIdentify) algorithm is a DFS-like greedy algorithm. It proceeds from node to node of colored three-valued game-graph recursively, until it finds a failure node. Moreover, the failure node fulfils Theorem 6.

Identifying failure nodes algorithm is a recursive algorithm, which needs a recursive stack, so its space complexity is $\mathcal{O}(|S|)$. It takes time complexity $\mathcal{O}(|S|)$ to find the adjacent points of each node in the adjacency matrix. To get the whole matrix, the total time complexity is $\mathcal{O}(|S|^2)$, where $|S|$ is the number of nodes.

Theorem 6 *A failure node returned by Identifying failure nodes algorithm is one of the following:* (1) *the terminal node* $(verifier, s^A, a/P_{\geq p}(a))$ *colored gray, where* $a \in AP$; (2) *the node in form of* $(verifier, s^A, P_{\geq p}(X\Psi), \Omega)$ *colored gray, where* $\sum_{t^A} P(s^A, s^{A\prime}) \cdot \mathrm{ps}^{A\prime} > 1 - \mathrm{p}$, $P(s^A, s^{A\prime})$ *is the upper probability from the place* s^A *to* $s^{A\prime}$, $s^A \to s^{A\prime}$ *and* $ps^{A\prime}$ *is the probability of* $s^{A\prime}$ *satisfying* $\neg\Psi$.

Proof According to coloring algorithm, Φ is $\Phi_0 \wedge \Phi_1$, $\Phi_0 \vee \Phi_1$, $\mathrm{P}_{\geq p}(\Psi_0 \wedge \Psi_1)$, or $\mathrm{P}_{\geq p}(\Psi_0 \vee \Psi_1)$, the node of which is colored gray only if it has at least one son that is colored gray. Thus, these nodes do not satisfy Definition 10 (failure node). For $\mathrm{P}_{\geq p}(\Psi_1 \mathrm{U}\, \Psi_2)$ or $\mathrm{P}_{\geq p}(\Psi_1\, \mathrm{R}\Psi_2)$, the node of which is colored gray depends on the witness of operator $\wedge$, $\vee$ or X, so, it also does not satisfy Definition (failure node). If the node is a terminal node, it has to be $(verifier, s^A, a/\mathrm{P}_{\geq p}(a))$, since $(verifier, s^A, \mathrm{true/false}/\mathrm{P}_{\geq p}(\mathrm{true})/\mathrm{P}_{\geq p}(\mathrm{false}))$ is colored by black or white, definitely. If the node is the form of $(verifier, s^A, \mathrm{P}_{\geq p}(\mathrm{X}\Psi), \Omega)$, the value of $\sum_{t^A} P(s^A, s^{A\prime}) \cdot ps^{A\prime} > 1 - \mathrm{p}$ has to be true, where $P(s^A, s^{A\prime})$ is the upper probability from the place s^A to $s^{A\prime}$, $s^A \xrightarrow{t^A} s^{A\prime}$ and $ps^{A\prime}$ is the probability of $s^{A\prime}$ satisfying $\neg\Psi$. Otherwise, it will be colored by black. □

7.2 Refining failure nodes

For returning the definite result of three-valued stochastic model checking, it is enough to refine the failure node not all the indefinite nodes. The type of failure node in Theorem 6 provides the criteria for the refinement.

(1) The failure node is a terminal node $(verifier, s^A, a/\mathrm{P}_{\geq p}(a))$. The reason for its gray color is the fact that one or some concrete places abstracted by s^A are labeled a,

meanwhile one or some concrete places abstracted by s^A are labeled $\neg a$. In order to avoid the indefinite result returned at s^A, s^A is refined by two abstract places s^{A1} and s^{A2}, where the concrete place set FP_t abstracted by s^{A1} are labeled a and the concrete place set FP_f abstracted by s^{A2} are labeled $\neg a$.

(2) The failure node is the form of $(verifier, s^A, \mathrm{P}_{\geq p}(\mathrm{X}\Psi), \Omega)$, where $\sum P(s^A, s^{A\prime}_{t^A}) \cdot ps^{A\prime} > 1 - \mathrm{p}$, $P(s^A, s^{A\prime})$ is the upper probability from the place s^A to $s^{A\prime}$, $s^A \to s^{A\prime}$ and $ps^{A\prime}$ is the probability of $s^{A\prime}$ satisfying $\neg\Psi$. In order to avoid the indefinite result returned at s^A, s^A is refined by two abstract places s^{A1} and s^{A2}, where the concrete place set FP_t abstracted by s^{A1} refute $\mathrm{P}_{\geq p}(\mathrm{X}\Psi)$ definitely and the concrete place set FP_f abstracted by $s^{A2} = s^A \backslash s^{A1}$.

From the above analysis, it can be seen that the key for refining failure nodes is to split the abstract places in a way that eliminates the failure cause. Actually, a place is a family of valuations for all variables in V. We uses$^A_{\to v}$ to denote the valuation of a place s^A for the variable$v$. Given an abstract place in the failure node, they cannot be distinguished in the abstract model with the visible variables. Formally, $\forall v \in V_{S^A}$, $s^{A1}_{\to v} = s^{A2}_{\to v}$, but s^{A1} and s^{A2} should be distinguished in the refinement model. This is a state separation problem (SSP), and the minimal state separation problem (MSSP) is the NP-hard problem (Clarke et al. 2002). Moreover, the place at a failure node may abstract many concrete places, especially for large concrete model, which makes time complexity of refining is high. For dealing with this, we take the approximate solution of SSP as a tradeoff between the precision and time complexity. We infer the separation set on sample place set selected by sample learning (He et al. 2016, 2010; Clarke et al. 2002), instead of the entire concrete place set abstracted by the failure node, then, we use BPSO (binary particle swarm optimization) algorithm to solve the separation set for refinement.

7.2.1 Formalizing refinement as the MSSP

The place s^A at a failure node is denoted as a vector of length n with $s^A[i] = s^A_{\to v}$, where v is the i th invisible variable in V_N. s^A cannot be separated by any present visible variable, so, we only consider its invisible variables. For simplicity, we use p_j to denote a pair of places in $FP_t \times FP_f$, i.e., $FP_t \times FP_f = \{p_1, p_2, \ldots, p_m\}$, $1 \leq j \leq m$. Assume $p_j =< (s)^{p_j}, (s')^{p_j} >$, p_j can be separated by certain variable in the separation set Λ, i.e., $\exists v_i \in \Lambda$, $(s)^{p_j}[i] \neq (s')^{p_j}[i]$, where $(s)^{p_j} =< (s)^{p_j}[1], (s)^{p_j}[2], \ldots, (s)^{p_j}[n] >$ and $(s')^{p_j} =< (s')^{p_j}[1], (s')^{p_j}[2], \ldots, (s')^{p_j}[n] >$. If $v_i \in \Lambda$, we define the decision variable $x_i = 1$, else $x_i = 0$, which is equivalent to $\sum_{i=1}^{n}((s)^{p_j}[i] \oplus (s')^{p_j}[i]) \bullet x_i$, where $\oplus$ is the exclusive or operator, x_i is the decision variable of v_i. Let $A = \{a_{ij}\}_{m\times n}$ be a coefficient matrix where $a_{ij} = (s)^{p_j}[i] \oplus (s')^{p_j}[i]$, $1 \leq i \leq n, 1 \leq j \leq m$. $a_{ij} = 1$ iff the state pair p_j is separated by the variable v_i. The refinement is the MSSP with n invisible variables and m state pairs: $min \sum_{i=1}^{n} x_i$, where $\sum_{i=1}^{n} a_{ij}x_i \geq 1, j = 1, \ldots, m; x_i = \{0, 1\}, i = 1, \ldots, n$.

```
$\Lambda := \emptyset$
SAMPLE $:= \emptyset$
N_SAMPLE $:= \emptyset$
While $(FP_t \times FP_f \neq \emptyset)$
  For $j$=1 to MAXSAM do
    Pick the next sample $< s, s' >$ from $FP_t \times FP_f$
    If $< s, s' >$ cannot be covered by $\Lambda$ then
      N_SAMPLE $:=$ N_SAMPLE $\cup < s, s' >$
    End if
  End for
SAMPLE:= SAMPLE $\cup$ N_SAMPLE;
Call solver to compute the separation set $\Lambda$ based on SAMPLE;
Update $FP_t \times FP_f$
End while
```

Fig. 5 Sample learning algorithm

7.2.2 Sample learning

We adopt sample learning (Clarke et al. 2002) algorithm to get the smaller place set S_{FP_t} and S_{FP_f} from S_{FP_t} and S_{FP_f}, respectively, where $S_{FP_t} \subseteq FP_t$ and $S_{FP_f} \subseteq FP_f$. Then, the minimal separation set is computed on S_{FP_t} and S_{FP_f}, which equals to be computed on FP_t and FP_f. In the process of selecting the sample, places that contain more information will be selected, instead of random selecting places.

The iterative sample learning algorithm is shown in Fig. 5. MAXSAM denotes the maximal number of samples picked in the iteration. At each iteration, MAXSAM samples are selected from $FP_t \times FP_f$ which cannot be separated by the present separation set. Then separation set is computed, and $FP_t \times FP_f$ is updated, until $FP_t \times FP_f = \emptyset$.

According to the formula $\sum_{i=1}^{n} (s)^{p_j}[i] \oplus (s')^{p_j}[i]) \bullet x_i$, if $\sum_{i=1}^{n} a_{ij}x_i \geq 1$, i.e., $A_j x_i \geq 1$, the place pair p_j can be separated. The effective approach of checking the validity of samples is based on $\sum_{i=1}^{n} a_{ij}x_i \geq 1$. This is easy to be performed and can always be accomplished in a constant time. Note that the order of selecting the sample is also important. In each iteration process, if there are not place pairs separated by the existing separation set, it will be selected, as can be separated by less variables in a greater probability as a sample. The sample learning algorithm is a recursive process, and its time complexity is $\mathcal{O}\left(\left|FP_t \times FP_f\right| * MAXSAM\right)$.

7.2.3 BPSO algorithm

PSO (Particle swarm optimization) algorithm (Kennedy and Eberhart 1995; Wang et al. 2020) is a kind of EA (evolutionary algorithm), which originally was put forward by Eberhart and Kennedy in 1995. PSO algorithm is modified in binary form to obtain the invisible variables which should be visible in the refined model.

(1) Objective function

The MSSP problem that we are solving is a constraint optimization problem, which can be described as: $min \sum_{i=1}^{n} x_i$, where $\sum_{i=1}^{n} a_{ij}x_i \geq 1,\ j = 1, \dots, m;\ x_i = \{0, 1\},\ i = 1, \dots, n.$

The particle in PSO algorithm has a strong ability to search generally at the first time, then it has to convergence. We adopt a non-stationary multi-stage assignment penalty to gain more accurate results and improve the convergence. The penalty factor is modified dynamically, according to the constraint function. Then, the optimization problem turns into:

$$\text{Minimize} \sum_{i=1}^{n} x_i + h(k) \sum_{j=1}^{m} f\left(\sum_{i=1}^{n} a_{ij}x_i \geq 1 \right) \tag{1}$$

where $h(k)$ is a penalty factor, the value of which is $\sup(\sqrt{k})$, k is the current iteration, $f(\bullet)$ is a penalty function which has a strong influence on the performance of the algorithm. A simple and efficient penalty function is employed as follows: $f(x) = \begin{cases} 0, & \textit{if } x \textit{ is true} \\ BIGVALUE, & \textit{otherwise} \end{cases}$.

(2) Probabilistic initialization of particles

We use a n-bit vector $\vec{xx}$ as a particle, i.e., $\vec{xx} = \left(v_1, v_2, \dots, v_n\right)$, where n is the number of invisible variables. If the value of the i th bit is 1, the variable v_i is selected into the separation set. The number of populations is Pop-size. We generate particles randomly in order to get wide guidance particles. The initialization process of particles is shown in Fig. 6.

(3) Velocity and position evolution

The update formulae of velocity and position in the classical PSO algorithm is not suitable for discrete constraint optimization problem, we need to reconstruct the formulae. We adopt the method proposed in Nguyen et al. (2021) with some modifications to update the velocity and position of a particle, which is named as BPSO (binary particle swarm optimization algorithm).

Two vectors for each particle are introduced as $\overrightarrow{V}_i^0$ and $\overrightarrow{V}_i^1.\overrightarrow{V}_i^0$ is the probability of the bits of the i th particle to change to zero, while $\overrightarrow{V}_i^1$ is the probability that bits of the i th particle to change to one. Since the inertia term is used in the process of update equation, these velocities are not complementing. Then, we give the definition of the probability of change in the j th bit of the i th particle:

Produce the size of the separation set randomly, i.e., an integer e $0 \leq e \leq n$.
Select e variables randomly into the separation set, and the probability to be selected is proportional to the number of place pairs it separated.

Fig. 6 Initializing a particle

$$V^c_{ij} = \begin{cases} V^1_{ij} \; if \; xx_{ij} = 0 \\ V^0_{ij} \; if \; xx_{ij} = 1 \end{cases} \tag{2}$$

Assume that the j th bit of the i th best particle is one. The velocity of particle is calculated in this way. The j th bit of the i th particle is guided to its best position. The velocity of change to one ($\overrightarrow{V}^1_{ij}$) for that particle increase and the velocity of change to zero ($\overrightarrow{V}^0_{ij}$) is decreased. We get the following rules: (1)
$If\ p^j_{ibest} = 1\ then\ d^1_{ij,1} = c_1 r_1\ and\ d^0_{ij,1} = -c_1 r_1,$ (2)
$If\ p^j_{ibest} = 0\ then\ d^0_{ij,1} = c_1 r_1\ and\ d^1_{ij,1} = -c_1 r_1,$ (3)
$If\ p^j_{gbest} = 1\ then\ d^1_{ij,2} = c_2 r_2\ and\ d^0_{ij,2} = -c_2 r_2,$ (4)
$If\ p^j_{gbest} = 0\ then\ d^0_{ij,2} = c_2 r_2\ and\ d^1_{ij,2} = -c_2 r_2,$ where d^1_{ij}, d^0_{ij} are two temporary values, r_1 and r_2. are two random variable in the range of (0,1) which are updated at each iteration, c_1 and c_2 are two fixed constants which are determined by user. Then: $V^1_{ij}(t+1) = wV^1_{ij}(t) + d^1_{ij,1} + d^1_{ij,2}$, $V^0_{ij}(t+1) = wV^0_{ij}(t) + d^0_{ij,1} + d^0_{ij,2}$, where w is the inertia term.

Due to the velocities of the particles must be restricted within the range between 0 and 1 (Nguyen et al. 2017) the normalization function used here is a sigmoid function: $vel'_{ij} = sig(vel_{ij}) = \frac{1}{1+e^{-vel_{ij}}}$. So, the new position of the particle is obtained with the below equation:

$$xx_{ij}(t+1) = \begin{cases} \overline{xx}_{ij}(t), & if\ r_{ij} < Vel'_{ij} \\ xx_{ij}(t), & otherwise \end{cases} \tag{3}$$

where $\overline{xx}_{ij}$ is the 2's complement of xx_{ij}. i.e., if $xx_{ij} = 0$, then $\overline{xx}_{ij} = 1$; if $xx_{ij} = 1$, then $\overline{xx}_{ij} = 0.r_{ij}$ is a uniform random number between 0 and 1. The value is proportional to the number of place pairs it separated. The BPSO algorithm shown in Fig. 7.

The time complexity of BPSO algorithm is linear with the number of iterations m and dimension n, its time complexity is $\mathcal{O}(m * n)$.

7.3 Incremental refinement

We refine the abstract model via splitting its places belong to failure node. There is no reason to split places for which the three-valued stochastic model checking results are definite. Although, the refinement process is decided locally, it has a global effect, since the refinement leads to the change of whole abstract model.

```
Generate an initial population according to algorithm in Figure 6;
While not (terminal condition) do
    Computing the fitness of each particle according to the object functi on equality (1) in section 6.2;
    Update the local best position for each particle according to classical PSO algorithm;
    Update the global best position for all particles according to classical PSO algorithm;
    Substitute the velocity and position according to equality (2) (3) in section 6.2;
End while.
```

Fig. 7 BPSO algorithm

The results of game-based three-valued stochastic model checking provide the valuable information for avoiding unnecessary refinement. At the end of the ith iteration of three-valued abstraction-refinement, the nodes that were colored gray are remembered. During the construction of a new refined three-valued game-graph in the $(i+1)$th iteration, we prune the game-graph according to the remembered information. As a result of this, only the places of gray nodes are refined. The iterative abstraction-refinement process give rise to an incremental three-valued abstraction-based stochastic model checking framework. Moreover, the termination of the abstraction-refinement is given by the following theorem.

Theorem 7 *For a finite concrete LNPPN model, the iterative three-valued abstraction-refinement process can guarantee to terminate with a definite result.*

Proof The refinement is done by means of splitting the indefinite places, which leads to a refined abstract LGIPPN model. So, every place s_r^A in the refined model has some corresponding super-place s_s^A in the less refined model, in the sense that the set of concrete places that s_r^A abstracts is a subset of those abstracted by s_s^A. Moreover, there is at least one abstract place is splited in the refinement, which ensures that at least one of the refined places abstracts strictly fewer concrete places than its super-place. Thus, the number of concrete places is the bound of iterations of abstraction-refinement. Therefore, the abstraction-refinement process is guaranteed to terminate if the place space is finite, and when there are not the indefinite places which leads to the indefinite result. □

The termination of the three-valued abstraction-refinement process means the three-valued abstraction-based PCTL* stochastic model checking returns a definite result.

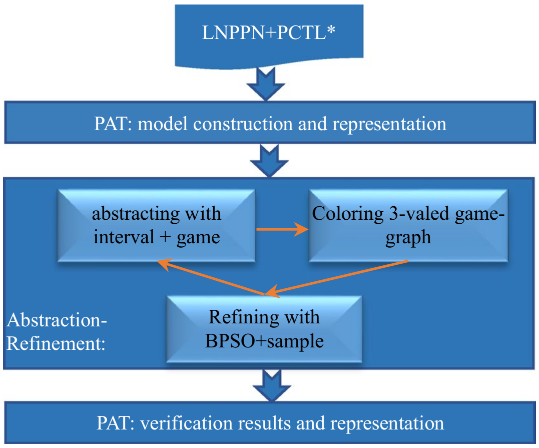
LNPPN+PCTL*
PAT: model construction and representation
abstracting with interval + game
Coloring 3-valed game-graph
Refining with BPSO+sample
Abstraction-Refinement:
PAT: verification results and representation

Fig. 8 Prototype tool TVAR

8 Case study

8.1 Experiments

Based on our open-source model checker PAT(Liu et al. 2011; Liu 2019), we implement three-valued abstraction-refinement framework in a prototype tool TVAR (three-valued abstraction-refinement), as shown in Fig. 8. It is developed in Java language with explicit-state data structures (sparse matrices, bit-sets, etc.). A symbolic (MTBDD-based) implementation of TVAR with implicit-state data structures (MTBDD), is being developed to offer more scalability on models with regularity. It will be released as the open-source software at PAT website (http://pat.comp.nus.edu.sg).

In order to demonstrate the feasibility and efficiency of three-valued abstraction-refinement framework in this paper, we compare it with the current approximate abstraction techniques of stochastic model checking (game-based abstraction and probabilistic CEGAR). The cases are the popular protocol algorithms in software-driven autonomous systems, and belong to the PRISM Benchmark Suite. All experiments are run on a PC with CPU Intel Core i7-3632QM (2.20 GHz) and RAM 8.00 GB.

The first case is the IPv4 Zeroconf protocol, which is a dynamic IP addresses configuration protocol for some software-driven autonomous systems connecting the network(Ejaz et al. 2019). It is a distributed "plug-and-play" manner for IP address configuration, which is automatically executed when an autonomous system is connected to the network. The process of IP addresses configuration is as follows: (1) When an autonomous system is being connected to the network, it firstly chooses a random IP address from the 65,024 available addresses (from 169.254.1.0 to 169.254.254.255) which are allocated by the Internet Assigned Number Authority for the purpose of such link-local networks. (2) Then the autonomous system will send messages to the other autonomous systems that have been connected to the network, and ask whether any of them are currently using the chosen IP address. (3) If no reply is received by the system, even after such messages were resent three more times, it starts to use the chosen IP address, and sends two more messages to claim that the IP address is occupied. (4) If the chosen IP address is being used, which will be replied by the corresponding system, the configuration process will return to 1) and the new system will choose another random IP address. Note that it sends the message repeatedly to avoid the message loss in the network. This IP address configuration process is both probabilistic and timed. Probability is used to characterize the random in initial selection of an IP address, or the message loss. Whereas, timing aspect is used to define the time periods that elapse between repeated retransmissions of the same message. In this case, the model to be verified is Zeroconf network configuration protocol for configured hosts K and IP addresses J, the property to be verified is minimum probability that the host configures successfully.

The second case is the IEEE 802.11 WLAN (Wireless Local Area Networks) protocol for network composed with the autonomous systems, which is used for a limited geographical area, e.g., stations, homes, offices, campuses(Chi and Chen 2019).

The international standard IEEE 802.11 is developed for the interoperability of heterogeneous communication devices of autonomous systems in WLAN. It is part of the IEEE 802 technical standards, and specifies the set of MAC (Media Access Control) and PHY (Physical Layer) protocols for implementing wireless devices communication of autonomous systems. It is not the same as wired devices, the stations of a WLAN cannot employ medium access control schemes, e.g., CSMA/CD (Carrier Sense Multiple Access with Collision Detection), to prevent simultaneous transmission on the channel, as they are unable to listen to their own transmission. IEEE 802.11 standard describes a CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) mechanism, which uses a randomized exponential backoff rule to minimize the likelihood of transmission collision. In this case, the model to be verified is IEEE 802.11 WLAN protocols with a backoff counter maximum of *bc* and a maximum packet send time of 500 μs, and the property to be analyzed is the minimum probability that a station's backoff counter reaches *bc*.

The third case is the IEEE 802.3 CSMA/CD (Carrier Sense, Multiple Access with Collision Detection) protocol for autonomous systems network, which is also part of the IEEE 802 technical standards (Dey et al. 2020). It is designed for networks with a single channel and specifies the behaviors of stations with the aim of minimizing simultaneous use of the channel (data collision). The basic structure of the protocol is as follows: (1) When a station tries to send data, it will listen to the medium; (2) If the medium was free (no one transmitting), the station starts to send its data; (3) If the medium is busy, the station waits a random amount of time and then repeats this process. In this case, the model to be verified is IEEE 802.3 CSMA/CD protocols with a backoff counter maximum of *bc* and a maximum packet send time of 500 μs, the property to be analyzed is minimum probability that a station's backoff counter reaches *bc*.

The experimental results are shown in Tables 2 and 3. The "actual value" columns in Tables 2 and 3 are the exact probability value obtained by monolithic stochastic model checking tool PRISM. The model statistics are shown in Tables 4 and 5. We will compare them further and analyse the reasons for them in the next section.

8.2 Analysis

The experimental results are very encouraging. In 14 cases of Tables 2 and 3, three-valued abstraction of this paper and game-based abstraction successfully generate tighter results than Probabilistic CEGAR. The three-valued abstraction framework gets the tightest results of them. The upper bounds produced by our framework is in coincidence with the actual value, and yields the exact values in some cases (for instance, IPv4 Zeroconf protocol with parameters J = 32, K = 4; J = 32, K = 5; and J = 64, K = 4). The reasons for this are that in three-valued abstraction and game-based abstraction, the additional nondeterministic behaviors of player 1 place are kept, and the transitions are equipped with interval in three-valued abstraction. In order to present the significance of our framework on the lower bound and upper bound, we make the statistical tests, as shown in Figs. 9 and 10. It can be seen that

Table 2 Minimum probability that the new host eventually succeeds in selecting a fresh IP address

Zeroconf protocol (parameter)		Actual value	Probabilistic CEGAR (LNPPN)		Game-base abstraction (LGPPN)		Three-valued abstraction (LGIPPN)		Case number
J	*K*		Lower bound	Upper bound	Lower bound	Upper bound	Lower bound	Upper bound	
32	4	0.99997866	0.98645312	0.99999998	0.99993761	0.99997866	0.99997866	0.99997866	1
	5	0.99997575	0.98478524	0.99999999	0.99991922	0.99997575	0.99991423	0.99997575	2
	6	0.99997248	0.97459618	0.99999986	0.99989916	0.99999973	0.99989869	0.99999982	3
	7	0.99997097	0.95879365	0.99999985	0.99987737	0.99999969	0.99987754	0.99999973	4
	8	0.99996896	0.95087699	0.99999978	0.99985421	0.99999948	0.99985437	0.99999968	5
64	4	0.99999023	0.92001763	0.99999989	0.99999023	0.99999023	0.99999023	0.99999023	6
	5	0.99998894	0.90899373	0.99999990	0.99996311	0.99999989	0.99995813	0.99999995	7
	6	0.99998776	0.90987821	0.99999999	0.99995499	0.99999994	0.99995440	0.99999990	8
	7	0.99998751	0.89889379	0.99999991	0.99994658	0.99999986	0.99994650	0.99999989	9
	8	0.99998722	0.87761651	0.99999992	0.99993777	0.99999984	0.99993780	0.99999989	10

Table 3 Minimum probability that a station's backoff counter reaches *bc*

Case (parameter)		Actual Value	Probabilistic CEGAR (LNPPN)		Game-base abstraction (LGPPN)		Three-valued abstraction (LGIPPN)		Case number
			Lower bound	Upper bound	Lower bound	Upper bound	Lower bound	Upper bound	
WLAN	4	0.843821732	0.76424334	0.94214258	0.80353624	0.87743524	0.79454563	0.89463738	11
(*bc*)	5	0.678546653	0.54323642	0.73244352	0.66734532	0.70156453	0.58983968	0.73594450	12
CSMA	6	0.869791274	0.63342225	0.89942347	0.74525672	0.893735433	0.68334906	0.89634574	13
(*bc*)	7	0.583332547	0.24362437	0.84354273	0.56346763	0.59435457	0.48890561	0.676769807	14

Table 4 Model statistics (places, transitions)

Zeroconf protocol (parameter)		Concrete model (LNPPN)		Probabilistic CEGAR (LNPPN)		Game-base abstraction (LGPPN)		Three-valued abstraction (LGIPPN)		Case number
J	*K*	places	transitions	places	transitions	places	transitions	places	transitions	
32	4	76,760	50,639	2326	1619	2338	1601	1998	1386	1
	5	197,698	139,201	2518	1702	2466	1681	2126	1467	2
	6	578,780	432,979	2698	1826	2604	1771	2311	1598	3
	7	835,536	615,023	2714	1979	2670	1813	2375	1626	4
	8	1,686,672	1,254,487	2968	2002	2738	1857	2472	1689	5
64	4	148,468	98,091	2598	1890	2536	1799	2098	1408	6
	5	383,498	270,281	2694	1992	2683	1898	2238	1499	7
	6	1,122,011	839,826	2821	2174	2765	1932	2387	1601	8
	7	1,616,327	1,189,798	3008	2221	2803	1946	2535	1678	9
	8	3,278,518	2,439,613	3019	2312	2820	1979	2643	1731	10

Table 5 Model statistics (places, transitions)

Case (parameter)		Concrete model (LNPPN)		Probabilistic CEGAR (LNPPN)		Game-base abstraction (LGPPN)		Three-valued abstraction (LGIPPN)		Case number
		places	transitions	places	transitions	places	transitions	places	transitions	
WLAN	4	1,048,291	708,969	5168	2296	5093	589	4768	67	11
(*bc*)	5	4,061,831	3,718,256	11,835	8098	11,592	4898	10,852	231	12
CSMA	6	2,099,546	1,121,659	5,408	1978	5347	694	5006	109	13
(*bc*)	7	6,680,611	4,986,817	10,294	4392	10,212	3,663	9561	348	14

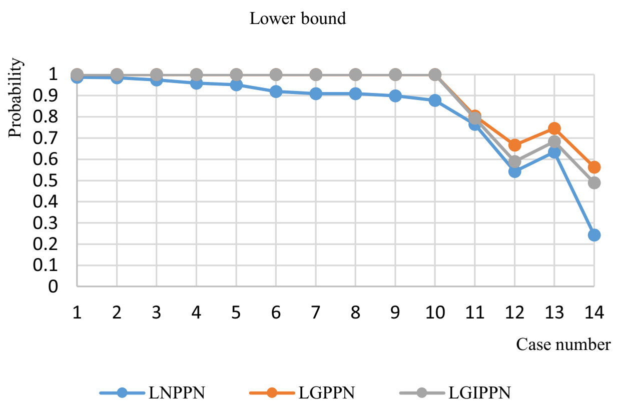

Fig. 9 Comparison of lower bound

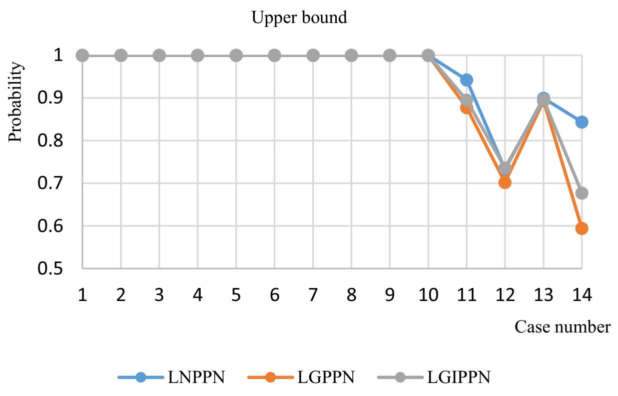

Fig. 10 Comparison of upper bound

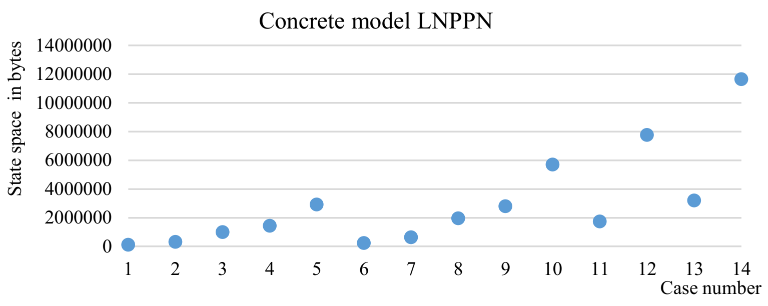

Fig. 11 State space of concrete model

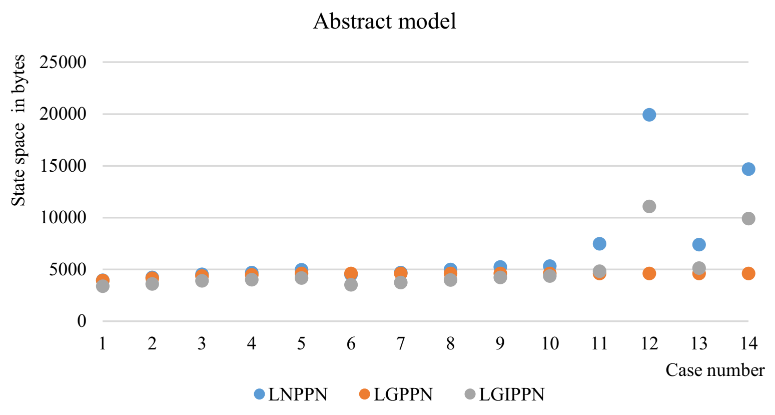

Fig. 12 State space of abstract model

LNPPN, LGPPN and LGIPPN get very similar results in solving their upper bound probability for IPv4 Zeroconf protocol. The results of LGPPN and LGIPPN are better than LNPPN in solving the lower bound probability. For WLAN and CSMA protocols, the compactness of the solutions by LGIPPN, LGPPN and LNPPN decreases in turn. The reasons for which are the nondeterminism modeled by LGPPN can be closer to the real probability value than LNPPN as an abstract model, and LGIPPN introduces a third kind of nondeterministic interval value, which preserves more information of transition probability and can simultaneously approximate the upper bound and lower bound.

The state space of abstract model LGIPPN in our framework is significantly reduced, compared with LNPPN and LGPPN. The state space of concrete model for 14 cases is shown in Fig. 11, and the state space of abstract model is compared in Fig. 12. The abscissa represents 14 cases, and the ordinate represents the size of

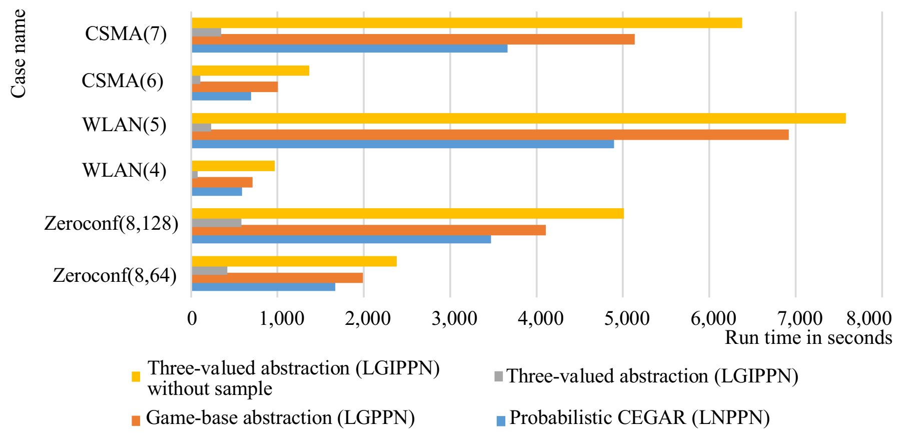

Fig. 13 Run time

state space in bytes. Figure 12 shows that the abstract state space of our framework is smaller than probabilistic CEGAR or game-based abstraction, and the state space of probabilistic CEGAR and game-based abstraction are almost the same. This is because: (1) the places and transitions of abstract model is constructed by nearly the same method in all abstraction techniques, but in three-valued abstraction, the transitions are equipped by interval; (2) the dependent relation of variables is very strong, or the domain of some a variable is too big. It hides all logical relation which is defined on these variables. When some invisible variables are made visible in the refinement, the corresponding predicates variables are added, which may also lead to search more place space. It can be concluded that different transition methods have great influence on the state space.

We also compare the time efficiency of our framework with others. As shown in Fig. 13, we compare our framework with probabilistic CEGAR (LNPPN), game-base abstraction (LGPPN), and our framework without sample learning + BPSO. The run time (in seconds) contains the time spent on all stages, including abstract model construction, model checking abstract model and refinement. The run time of three-valued abstraction in this paper is distinctly less than probabilistic CEGAR and game-based abstraction, because the refinement in it is accelerated by sample learning and BPSO algorithm. When our framework integrates the sample learning + BPSO, it takes obvious less time, which makes the advantage of time consumption more apparent. BPSO has a trade-off between the solution precision and cost, which gives a suboptimal but sufficiently good set separation and makes refinement iterations less. It plays an important role to reduce the running time of LGIPPN. Moreover, the bigger of concrete model, the more preponderance of three-valued abstraction emerges. But, if the three-valued abstraction do not use the sample learning and BPSO algorithm for refining, it will be the worst among the abstraction techniques. Because refining the transitions with interval and places of player 1 in iteration is very time-consuming. The run time of probabilistic CEGAR is less than game-based abstraction, because refining the places of player 1 in iteration is very time-consuming.

9 Conclusion

In this paper, we focus on approximate model abstraction techniques for dealing with state space explosion problem in stochastic model checking. We propose the first three-valued abstraction-refinement framework for stochastic model checking, which is also the first abstraction framework for preserving full PCTL* properties. We present a good balance between the state space and preserved properties of abstract model. The key components of the framework are a new abstract model which orthogonally integrates interval probability of transition and game for nondeterminism, and the refinement process which combines sample learning and BPSO algorithm. Some popular protocols in software-drived autonomous systems are used to demonstrate the efficiency of the framework, which are the best among the current approximate model abstraction techniques. This framework takes LNPPN as the concrete model, but it is applicable to other formal system models as well, such as probabilistic automaton and MDP. However, this framework can just abstract the autonomous system with discrete time stochastic behaviors, and just can be applied at the level of system model. In the future work, we will (1) generalize the three-valued abstraction framework to software-driven autonomous system in continuous time style or hybrid system, (2) generate models from the runtime autonomous system by L* or reinforcement learning algorithms(Zhang et al. 2020), and (3) develop the symbolic (MTBDD-based) implementation of this abstraction framework to offer improved scalability on models exhibiting regularity.

Acknowledgements The work was supported by Singapore-UK Cyber Security of EPSRC under Grant Nos. EP/N020170/1, MOE Humanities and Social Sciences Foundation of China under Grant Nos. 20YJCZH102. We would like to extend our deepest respects to Professor Edmund M. Clarke at Carnegie Mellon University, USA, who received ACM Turing Award for his pioneering work of model checking and passed away on December 22, 2020. He inspired us a lot through his books and papers, especially the direct discussion about abstraction for stochastic model checking. Thanks to Professor Marta Kwiatkowska at University of Oxford, UK, for discussion with her about the game semantics for three-valued stochastic model checking.

References

Abraham, E., Becker, B., Dehnert, C., Jansen, N., Katoen, J.P., Wimmer, R.: Counterexample generation for discrete-time Markov models: an introductory survey. In: Proceedings of the 14th International School on Formal Methods for the Design of Computer, Communication and Software Systems: Executable Software Models (SFM-14:ESM), Springer, vol. 8483 of LNCS, pp. 65–121 (2014)

Albanese, M., Chellappa, R., Moscato, V., Picariello, A., et al.: A Constrained probabilistic petri net framework for human activity detection in video. IEEE Trans. Multimedia **10**(8), 1429–1443 (2008)

Alfaro, L., Roy, P.: Magnifying-lens abstraction for Markov decision processes. In: Proceedings of the 19th International Conference on Computer Aided Verification (CAV'07). Springer, vol. 4590 of LNCS, pp. 325–338 (2007)

Baier, C., Katoen, J.P.: Principles of Model Checking. MIT Press, Cambridge (2008)

Baier, C., Hermanns, H.: Weak bisimulation for fully probabilistic processes. In: Proceedings of the 9th International Conference on Computer Aided Verification. Springer-Verlag, Berlin, Heidelberg, pp. 119–130 (1997)

Baier, C., Groser, M., Ciesinski, F.: Partial order reduction for probabilistic systems. In: Proceedings of the 1st International Conference on Quantitative Evaluation of Systems. IEEE Computer Society Press, Washington, pp. 230–239 (2004)

Baier, C., Katoen, J.P., Hermanns, H., Wolf, V.: Comparative branching-time semantics for Markov chains. Inf. Comput. **200**(2), 149–214 (2005a)

Baier, C., D'Argenio, P., Groesser, M.: Partial order reduction for probabilistic branching time. Electron. Notes Theor. Comput. Sci. **153**(2), 97–116 (2005b)

Belardinelli, F., Lomuscio, A., Malvone, V.: An abstraction-based method for verifying strategic properties in multi-agent systems with imperfect information. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, no. 01, pp. 6030–6037 (2019)

Bernemann, R., Cabrera, B., Heckel, R., König, B.: Uncertainty reasoning for probabilistic petri nets via Bayesian networks, pp. 1–26 (2020) available: https://arxiv.org/abs/2009.14817

Buchholz, P.: Exact and ordinary lumpability infinite Markov chains. J. Appl. Probab. **31**(1), 59–75 (1994)

Chadha, R., Viswanathan, M.: A counterexample guided abstraction-refinement framework for Markov decision processes. ACM Trans. Comput. Logic **12**(1), 1–49 (2010)

Chi, T., Chen, M.: A frequency hopping method for spatial RFID/WiFi/Bluetooth scheduling in agricultural IoT. Wirel. Netw. **25**, 805–817 (2019)

Christian, D., Katoen, J.P., Parker, D.: SMT-based bisimulation minimization of Markov models. In: Proceedings of the 14th International Conference on Verification, Model Checking, and Abstract Interpretation. Springer-Verlag, Berlin, Heidelberg, pp. 28–47 (2013)

Christopher P.: Probabilistic symmetry reduction [Ph.D. Thesis]. University of Glasgow, Scotland (2012)

Ciesinski, F.: High-Level modelling and efficient analysis of randomized protocols [Ph.D. Thesis]. Dresden University of Technology, Dresden (2011)

Clarke, E.M., Grumberg, O., Long, D.E.: Model checking and abstraction. ACM Trans. Program. Lang. Syst. **16**(5), 1512–1542 (1994a)

Clarke, E.M., Grumberg, O., Long, D.E.: Model checking and abstraction. ACM Trans. Program. Lang. Syst. **16**(5), 1512–1542 (1994b)

Clarke, E.M., Jha, S., Enders, R., Filkorn, T.: Exploiting symmetry in temporal logic model checking. Form. Methods Syst. Des. **9**(1–2), 77–104 (1996)

Clarke, E., Gupta, A., Kukula, J., Strichman, O.: SAT based abstraction-refinement using ILP and machine learning techniques. In: Proceedings of Conference on Computer-Aided Verification, Copenhagen, Denmark (2002)

Clarke, E.M., Grumberg, O., Jha, S., Lu, Y., Veith, H.: Counterexample-guided abstraction refinement. J. ACM **50**(5), 752–794 (2003)

Clarke, E.M., Emerson, E.A., Sifakis, J.: Model checking: algorithmic verification and debugging. Commun. ACM **52**(11), 74–84 (2009)

Clarke, E.M., Henzinger, T.A., Veith, H., Bloem, R., et al.: Handbook of Model Checking. Springer, Heidelberg (2018)

D'Argenio, P.R., Niebert, P.: Partial order reduction on concurrent probabilistic programs. In: Proceedings of the 1st International Conference on Quantitative Evaluation of Systems. IEEE Computer Society Press, Washington, pp. 240–249 (2004)

Dams, D., Grumberg, O.: Abstraction and abstraction refinement. In: Clarke, E.M., Henzinger, T.A., Veith, H., Bloem, R. (eds.) Handbook of Model Checking. Springer, Heidelberg (2018)

Dehnert C.: The probabilistic model checker storm: symbolic methods for probabilistic model checking. PhD Thesis at RWTH Aachen University (2018)

Derisavi, S.: A symbolic algorithm for optimal Markov chain lumping. In: Proceedings of the 13th International Conference on Tools and Algorithms for the Construction and Analysis of Systems. Springer, Berlin, Heidelberg, pp. 139–154 (2007)

Dey, D., Dansana, J., Behura, A.: A survey of datalink layer protocol for IoT. In: Smys, S., Senjyu, T., Lafata, P. (eds) Second International Conference on Computer Networks and Communication Technologies, pp. 459–466 (2020)

Didier, F., Henzinger, T., Mateescu, M., Wolf, V.: Sabre: a tool for stochastic analysis of biochemical reaction networks. In: Proceedings of the 7th International Conference on the Quantitative Evaluation of Systems (QEST'10), pp. 193–194. IEEE CS Press (2010)

Donaldson, A., Miller, A.: Symmetry reduction for probabilistic model checking using generic representatives. In: Proceedings of the 4th International Conference on Automated Technology for Verification and Analysis. Springer-Verlag, Berlin, Heidelberg, pp. 9–23 (2006)

Donaldson, A., Miller, A., Parker, D.: Language-level symmetry reduction for probabilistic model checking. In: Proceedings of the 6th International Conference on Quantitative Evaluation of Systems. IEEE Computer Science Press, Washington, pp. 289–298 (2009)

Ebert, C., Weyrich, M.: Validation of autonomous systems. IEEE Softw. **36**(5), 15–23 (2019)

Ejaz, S., Iqbal, Z., Azmat Shah, P., Bukhari, B.H., Ali, A., Aadil, F.: Traffic load balancing using software defined networking (SDN) controller as virtualized network function. IEEE Access **7**, 46646–46658 (2019)

Emerson, E.A., Wahl, T.: On combining symmetry reduction and symbolic representation for efficient model checking. In: Proceedings of the 12th IFIP WG Advanced Research Working Conference on Correct Hardware Design and Verification Methods. Springer-Verlag, Berlin, Heidelberg, pp. 216–230 (2003)

Emerson, E.A., Sistla, A.: Symmetry and model checking. Form. Methods Syst. Des. **9**(1–2), 105–131 (1996)

Emerson, E.A., Wahl, T.: Efficient reduction techniques for systems with many components. Electron. Notes Theor. Comput. Sci. **130**, 379–399 (2005a)

Emerson, E.A., Wahl, T.: Dynamic symmetry reduction. In: Proceedings of the 11th International Conference on Tools and Algorithms for the Construction and Analysis of Systems. Springer-Verlag, Berlin, Heidelberg, pp. 382–396 (2005b)

Evangelidis A.: Verified control and estimation for cloud computing. Ph.D. thesis, School of Computer Science, University of Birmingham (2020)

Fecher, H., Leucker, M., Wolf, V.: Don't know in probabilistic systems. In: Proceedings of the 13th International Conference on Model Checking Software. Springer-Verlag, Berlin, Heidelberg, pp. 71–88 (2006)

Fernandez-Diaz, A., Baier, C., Benac-Earle, C., Fredlund, L.A.: Static partial order reduction for probabilistic concurrent systems. In: Proceedings of the 9th International Conference on Quantitative Evaluation of Systems. IEEE Computer Science Press, Washington, pp. 104–113 (2012)

Ferrer, F.L.M., Hashemi, V., Hermanns, H., Turrini, A.: Deciding probabilistic automata weak bisimulation: theory and practice. Form. Asp. Comput. **28**, 109–143 (2016)

Filieri, A., Ghezzi, C., Tamburrelli, G.: Run-time efficient probabilistic model checking. In: Proceedings of the 33rd ACM/IEEE International Conference on Software Engineering. Honolulu, HI, USA, pp. 341–350 (2011)

Fremont, D.J., Chiu, J., Margineantu, D.D., Osipychev, D., Seshia, S.A.: Formal analysis and redesign of a neural network-based aircraft taxiing system with VerifAI. In: 32nd International Conference on Computer Aided Verification (CAV), July (2020)

Gerth, R., Kuiper, R., Peled, D., Penczek, W.: A partial order approach to branching time logic model checking. In: Proceedings of the 3rd Israel Symposium on the Theory of Computing Systems. IEEE Computer Society Press, Washington, pp. 130–139 (1995)

Hahn, E.M., Hermanns, H., Wachter, B., Zhang, L.J.: PASS: abstraction refinement for infinite probabilistic models. In: Proceedings of the 16th International Conference on Tools and Algorithms for the Construction and Analysis of Systems. Springer-Verlag, Berlin, Heidelberg, pp. 353–357 (2010)

Hansen, H., Wang, X.: Compositional analysis for weak stubborn sets. In: Proceedings of the International Conference on Application of Concurrency to System Design. IEEE Computer Science Press, Washington, pp. 36–43 (2011)

Hansen, H., Kwiatkowska, M., Qu, H.: Partial order reduction for model checking Markov decision processes under unconditional fairness. In: Proceedings of the 8th International Conference on Quantitative Evaluation of SysTems. IEEE Computer Science Press, Washington, pp. 203–212 (2011)

Hark, M., Kaminski, B.L., Giesl, J., Katoen, J.P.: Aiming low is harder: induction for lower bounds in probabilistic program verification. In: Proceedings of the ACM Programming Language, POPL, Article 37, vol. 4, pp. 1–28 (2020)

Hartmanns, A., Junges, S., Katoen, J.P., Quatmann, T.: Multi-cost bounded reachability in MDPs. In: Proceedings of the of TACAS, vol 10805 of LNCS (2018)

Hashemi, V., Hermanns, H., Turrini, A.: On the efficiency of deciding probabilistic automata weak bisimulation. Electron. Commun. EASST (2013). https://doi.org/10.14279/tuj.eceasst.66.895

He, F., Song, X., Hung, W.N.N., et al.: Integrating evolutionary computation with abstraction refinement for model checking. IEEE Trans. Comput. **59**(1), 116–126 (2010)

He, F., Gao, X., Wang, M., Wang, B.Y., Zhang, L.J.: Learning weighted assumptions for compositional verification of Markov decision processes. ACM Trans. Softw. Eng. Methodol. **25**(3), 39 (2016)

Hermanns, H., Katoen, J.: Automated compositional Markov chain generation for a plain-old telephone system. Sci. Comput. Program. **36**(1), 97–127 (2000)

Hermanns, H., Turrini, A.: Deciding probabilistic automata weak bisimulation in polynomial time. In: Proceedings of the 32nd International Conference on Foundations of Software Technology and Theoretical Computer Science. Saarbrücken/Wadern: Dagstuhl Publishing, pp. 435–447 (2012)

Hermanns, H., Wachter, B., Zhang, L.: Probabilistic CEGAR. In: Proceedings of 2008 the 20th International Conference on Computer Aided Verification. Springer-Verlag, Berlin, Heidelberg, pp. 162–175 (2007)

Huang, M., Fu, H., Katoen, J.P.: Deciding probabilistic simulation between probabilistic pushdown automata and finite-state systems. Inf. Comput. **268**, 104431 (2019)

Huynh, T., Tian, L.: On some equivalence relations for probabilistic processes. Fundam. Inform. **17**(3), 211–234 (1992)

Jonsson, B., Larsen, K.G.: Specification and refinement of probabilistic processes. In: Proceedings of the 6th Annual IEEE Symposium on Logic in Computer Science. IEEE Computer Society Press, Washington, pp. 266–277 (1991)

Kamaleson, N.: Model reduction techniques for probabilistic verification of Markov chains. Ph.D. thesis, University of Birmingham (2018)

Katoen, J.P., Sher, F.: Modal stochastic games: abstraction-refinement of probabilistic automata. In: Models, Algorithms, Logics and Tools (Essays Dedicated to Kim Guldstrand Larsen on the Occasion of His 60th Birthday). LNCS, Springer, vol. 10460, pp. 426–448 (2017)

Katoen, J.P., Klink, D., Leucker, M., Wolf, V.: Three-valued abstraction for probabilistic systems. J. Logic Algebraic Program. **81**(4), 356–389 (2012)

Kattenbelt, M., Kwiatkowska, M., Norman, G., Parker, D.: A game-based abstraction refinement framework for Markov decision processes. Form. Methods Syst. Des. **36**(3), 246–280 (2010)

Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings IEEE International Conference on Neural Networks, pp. 1942–1948 (1995)

Kwiatkowska, M.: Safety verification for deep neural networks with provable guarantees. In: Proceedings of the 30th International Conference on Concurrency Theory, pp. 1–5 (2019)

Kwiatkowska, M., Norman, G., Parker, D.: Symmetry reduction for probabilistic model checking. In: Proceedings of the 18th International Conference on Computer Aided Verification. Springer-Verlag, Berlin, Heidelberg, pp. 234–248 (2006a)

Kwiatkowska, M., Norman, G., Parker, D.: Game-based abstraction for Markov decision processes. In: Proceedings of the 3rd International Conference on Quantitative Evaluation of Systems. IEEE Computer Science Press, Washington, pp. 157–166 (2006b)

Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: verification of probabilistic real-time systems. In: Proceedings of the 23rd International Conference on Computer Aided Verification. Springer-Verlag, Berlin, Heidelberg, pp. 585–591 (2011)

Kwiatkowska, M., Norman, G., Parker, D.: Symbolic verification and strategy synthesis for linearly-priced probabilistic timed automata. In: Aceto, L., Bacci, G., Bacci, G., Ingólfsdóttir, A., Legay, A., Mardare, R. (eds.) Models, Algorithms, Logics and Tools, vol. 10460, pp. 289–309. Springer, Cham (2017)

Kwiatkowska, M., Norman, G., Parker, D., Santos, G.: PRISM-games 3.0: stochastic game verification with concurrency, equilibria and time. In: Proceedings of the 32nd International Conference on Computer Aided Verification (CAV'20), Springer, vol. 12225 of LNCS, pp. 475–487 (2020)

Kwiatkowska, M., Norman, G., Parker, D., Santos, G.: Automatic verification of concurrent stochastic systems. Form. Methods Syst. Des. (2021). https://doi.org/10.1007/s10703-020-00356-y

Kwiatkowska, M., Norman, G., Parker, D.: Probabilistic model checking and autonomy. Annu. Rev. Control Robot. Auton. Syst. **5**, 1–26 (2022)

Lacerda, B., Faruq, F., Parker, D., Hawes, N.: Probabilistic planning with formal performance guarantees for mobile service robots. Int. J. Robot. Res. **38**(9), 1098–1123 (2019)

Larsen, K., Skou, A.: Bisimulation through probabilistic testing. Inf. Comput. **94**(1), 1–28 (1991)

Liu, Y.: Secure deep learning engineering: a road towards quality assurance of intelligent systems. In: The 21st International Conference on Formal Engineering Methods, November 5th–9th (2019)

Liu, Y., Sun, J., Dong, J.S.: PAT 3: an extensible architecture for building multi-domain model checkers. In: The 22nd annual International Symposium on Software Reliability Engineering (ISSRE 2011), Hiroshima, Japan, pp. 190–199, Nov 29–Dec 2 (2011)

Liu, Y., Li, X.D., Ma, Y.: Model abstraction for stochastic model checking. Ruan Jian Xue Bao/J. Softw. **26**(8), 1853–1870 (2015)

Liu, Y., Li, X.D., Ma, Y.: A game-based approach for PCTL* stochastic model checking with evidence. J. Comput. Sci. Technol. **31**(1), 198–216 (2016)

Luisa, V.L., Loreti, M., Nenzi, L., Hillston, J., Marion, G.: Three-valued spatio-temporal logic: a further analysis on spatio-temporal properties of stochastic systems. In: Proceedings 14th International Conference on Quantitative Evaluation of Systems, pp. 317–332 (2017)

Ma, Y., Cao, Z., Liu, Y.: A Probabilistic assume-guarantee reasoning framework based on genetic algorithm. IEEE Access **7**, 83839–83851 (2019a)

Ma, Y., Cao, Z., Liu, Y.: A PSO-based CEGAR framework for stochastic model checking. Int. J. Softw. Eng. Knowl. Eng. **29**(10), 1465–1495 (2019b)

Miller, A., Donaldson, A., Calder, M.: Symmetry in temporal logic model checking. ACM Comput. Surv. **38**(3), 8 (2006)

Milner, R.: An algebraic definition of simulation between programs. In: Proceedings of the 2nd International Joint Conference on Artificial Intelligence. William Kaufmann Inc., London, pp. 481–489 (1971)

Milner, R.: A Calculus of Communicating Systems. Springer-Verlag, Berlin, Heidelberg (1980)

Nguyen, B.H., Xue, B., Andreae, P.: A novel binary particle swarm optimization algorithm and its applications on knapsack and feature selection problems. In: Leu, G., Singh, H., Elsayed, S. (eds.) Intelligent and Evolutionary Systems. Proceedings in Adaptation, Learning and Optimization, vol. 8. Springer, Cham (2017)

Nguyen, B.H., Xue, B., Andreae, P., Zhang, M.: A new binary particle swarm optimization approach: momentum and dynamic balance between exploration and exploitation. IEEE Trans. Cybern. **51**(2), 589–603 (2021)

Norris, I.P.C., Dill, D.L.: Better verification through symmetry. Form. Methods Syst. Des. **9**(1–2), 41–75 (1996)

Oxford, M., Parker, D., Ryan, M.: Quantitative verification of certificate transparency gossip protocols. In: Proceedings of the IEEE Conference on Communications and Network Security, France, June 29–July 1, pp. 1–9 (2020)

Paige, R., Tarjan, R.: Three partition refinement algorithms. SIAM J. Comput. **16**(6), 973–989 (1987)

Paoli, F., Prabaldi, M.: Proof theory of paraconsistent weak Kleene logic. Stud. Logica **4**(108), 779–802 (2020)

Park D.: Concurrency and automata on infinite sequences. In: Proceedings of the 5th GI-Conference on Theoretical Computer Science. Springer-Verlag, Berlin, Heidelberg, pp. 167–183 (1981)

Peled, D.: All from one, one for all: on model checking using representatives. In: Proceedings of the 5th International Conference on Computer Aided Verification. Springer-Verlag, Berlin, Heidelberg, pp. 409–423 (1993)

Peled, D.: Partial order reduction: linear and branching temporal logics and process algebras. In: Proceedings of the DIMACS Workshop on Partial Order Methods in Verification. AMS Press, New York, pp. 79–88 (1996)

Peled, D., Pratt, V., Holzmann, G.: Partial order methods in verification. In: DIMACS Series in Discrete Mathematics and Theoretical Computer Science. (1997)

Petri, C.A.: Introduction to general net theory. In: Brauer, W. (ed.) Lecture Notes in Computer Science 84, pp. 1–19. Springer-Verlag, Berlin, Heidelberg (1979)

Pfeffer, A., Wu, C., Fry, G., Lu, K., et al.: Software adaptation for an unmanned undersea vehicle. IEEE Softw. **36**(2), 91–96 (2019)

Philippou, A., Lee, I., Sokolsky, O.: Weak bisimulation for probabilistic systems. In: Proceedings of the 11th International Conference on Concurrency Theory. Springer-Verlag, Berlin, Heidelberg, pp. 334–349 (2000)

Segala, R., Lynch, N.: Probabilistic simulations for probabilistic processes. Nord. J. Comput. **2**(2), 250–273 (1995)

Shivakumar, S., Torfah, H., Desai, A., Seshia, S.A.: SOTER on ROS: a run-time assurance framework on the robot operating system. In: 20th International Conference on Runtime Verification (RV), October (2020)

Shoham, S., Grumberg, O.: Game-based framework for CTL counterexamples and 3-valued abstraction-refinement. ACM Trans. Comput. Logic (TOCL) **9**(1), 1 (2007)

Valmari, A.: A stubborn attack on state explosion. Form. Methods Syst. Des. **1**(4), 297–322 (1992)

Wachter, B., Zhang, L.J.: Best probabilistic transformers. In: Proceedings of the 11th International Conference on Verification, Model Checking, and Abstract Interpretation. Springer-Verlag, Berlin, Heidelberg, pp. 362–379 (2010)

Wahl, T., Blanc, N., Emerson, E.A.: SVISS: symbolic verification of symmetric systems. In: Proceedings of the 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems. Springer-Verlag, Berlin, Heidelberg, pp. 459–462 (2008)

Wang, J., Jiang, C., Zhang, H., Ren, Chen K C., Hanzo, L.: Thirty years of machine learning: the road to pareto-optimal wireless networks. IEEE Commun. Surv. Tutor. **22**(3), 1472–1514 (2020)

Winterer, L., Junges, S., Wimmer, R., Jansen, N., Topcu, U., Katoen, J.P., Becker, B.: Motion planning under partial observability using game-based abstraction. In: IEEE 56th Annual Conference on Decision and Control (CDC), pp. 2201–2208, IEEE (2017)

Winterer, L., Junges, S., Wimmer, R., Jansen, N., Topcu, U., Katoen, J.P., Becker, B.: Strategy synthesis for POMDPs in robot planning via game-based abstractions. IEEE Trans. Autom. Control **66**(3), 1040–1054 (2020)

Younes, H.: Ymer: a statistical model checker. In: Proceedings of the 17th International Conference on Computer Aided Verification (CAV'05), Springer, vol. 3576 of LNCS, pp. 429–433 (2005)

Zhang, L.J.: Decision algorithms for probabilistic simulations [Ph.D. Thesis]. Saarland University, Saarbrücken (2008)

Zhang, L.J., David, N.J.: A space-efficient simulation algorithm on probabilistic automata. Inf. Comput. **249**, 138–159 (2016)

Zhang, L.J., Yang, P., Song, L., et al.: Probabilistic bisimulation for realistic schedulers. Acta Inform. **55**, 461–488 (2018)

Zhang, X., Zhou, Y., Han, T., Chen, T.: Training deep code comment generation models via data augmentation. In: Internetware, pp. 185–188 (2020)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Authors and Affiliations

Yang Liu[1,3] · **Yan Ma**[2,3] · **Yongsheng Yang**[1]

Yang Liu
yl.nus.sg@gmail.com

[1] Institute of Logistics Science and Engineering, Shanghai Maritime University, Shanghai 201306, China

[2] School of Accounting, Nanjing University of Finance and Economics, Nanjing 210023, China

[3] School of Computing, National University of Singapore, Singapore 117417, Singapore