Bayesian Parameter Synthesis of Markov Population Models.

Master Thesis

Submitted by

Nhat-Huy Phung

at the

Universität Konstanz

Modeling of Complex, Self-organising Systems

Department of Computer and Information Science

1. **Supervised by:** Jun-.Prof. Dr. Tatjana Petrov
2. **Supervised by:** Prof. Dr. Stefan Leue

Konstanz, 2020

Contents

Acknowledgements

To the completement of this thesis, I would like to describe my deep

Abstract

We present frameworks for data-informed parameter synthesis of Markov population processes. Given statistics data of the population at its steady-state, the object is to synthesize a set of parameters so that a temporal property of interest is satisfied. We design Bayesian frameworks for parameter synthesis in both cases: when the closed form of the interested property is obtainable, and when only simulation is possible. The frameworks are constructed with different sampling and optimization techniques to approximate the posterior distribution. Later, we evaluate the frameworks using different population models of different size using synthetic data generated from a known true parameter. By measuring the distance between synthesized parameters and true parameters and visualize sampled parameter values with their corresponding weights, we show that our frameworks are capable of deriving a set of satisfying parameter values, as well as an estimation which is close to the true parameter.

Chapter 1

Introduction

In different areas of research and application, the objects are to study how the number of individuals in a closed environment develop under a certain set of assumptions. For instance

- Number of online nodes in a distributed system.
- Number of surviving individuals in an epidemic model.

Markov population models [25] are finite state-space, stochastic models that is widely used in modeling complex and dynamic systems. In a Markov population model, each state represents the number of individuals. Formally, in a Markov population model whose state space is $S = (s_1, \dots, s_n)$, there is a map $f : S \to \{0, \dots, N\}$ where $N \in \mathbf{N}^*$ is the maximum number of individuals in the system.
In a Markov population models, for example Discrete-time Markov Chain, initial and transition probabilities are known a-priori. In order to encompass unknown attributes of a system, we introduce *parametric Markov population models*. In a parametric Markov population model, each transition is a rational function of parameters. As unknown features of the system are represented by parameters, the following research questions are raised

- Given a set of data collected by observing the system, how can we know about its parameters?
- Which values of parameters instantiate a model that satisfies a certain property of interest?

Parameter synthesis is an emerging research direction on probabilistic model checking. Katoen [24] define the parameter synthesis problem for pDTMC as to find a set of parameter values, which satisfy a given reachability property. In this thesis, we combines Bayesian parameter inference and parameter synthesis, so that the result parameters (1) satisfy the property of interest, and (2) likely to produce given steady-state data. Contributions of the thesis are

- Presenting and implementing a data-informed, Bayesian frameworks on parameter synthesis of parametric Discrete-time Markov Chain with different case studies.
- Comparing the performances of optimization methods used to approximate posterior distribution in both cases: closed-form solutions are available and only simulations are possible.
- Evaluating the scalability of the frameworks with different sizes of model state-space.

- **Chapter 1** introduces motivations and goals of this research.
- **Chapter 2** presents the theoretical background on probabilistic model checking, include discrete stochastic models and their corresponding temporal logics.
- **Chapter 3** presents essential concepts on Bayesian inference, including sampling and optimization algorithms.
- **Chapter 4** reviews the state-of-the-art works of other researchers on the problem of parameter synthesis.
- **Chapter 5** present Bayesian parameter synthesis frameworks.
- **Chapter 6** describes case studies and benchmarks presented frameworks under different setups.
- **Chapter 7** conclusion and possible future works.

Chapter 2

Probabilistic model checking

We use Discrete-time Markov chain as the formalism to model stochastic population process. In this chapter, we present essential concepts on probabilistic model checking, including probabilistic models and properties. We also briefly present a general deterministic model checking algorithm for a specific temporal logic, namely PCTL. Due to the state space explosion, applying deterministic model checking algorithm is possible to be computationally expensive. Therefore, we also present a simulation based model checking, namely *statistical model checking* for bounded and unbounded path property. Since statistical model checking relies only on simulation of stochastic models, it is advantageous for checking models with large space size. We also introduce definitions of parametric model and parameter synthesis problems, as well as the symbolic computing approach to verify parametric models.

2.1 Markov models

2.1.1 Discrete Time Markov chain

Markov models are stochastic models of discrete or continous time which satisfy memoryless property.

Definition 2.1.1 (Discrete-time memoryless property)
Let X be a random variable of geometric distribution. X has memoryless property if and only if

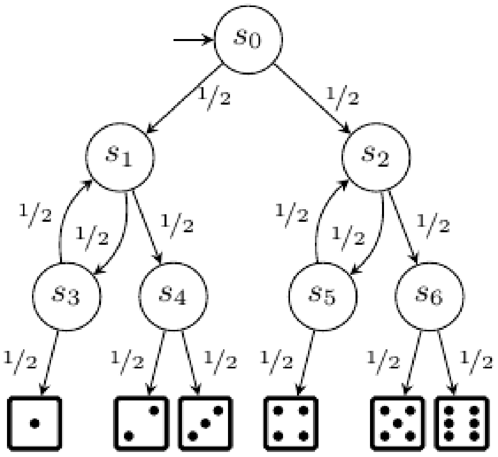
$$Pr(X = t + m | X > m) = Pr(X > m) \forall t, m \in \mathbb{N} k \geq 1$$

Markov model can be non-deterministic *(Markov Decision Process)*. However, in this thesis we consider only Markov models without non-determinism. The following definitions of discrete-time and continuous-time Markov chains follows the definitions presented by Baier [3].

Definition 2.1.2 (Discrete Time Markov Chain)
A Discrete-time Markov chain (DTMC) $\mathcal{M}$ is a tuple $(S, \mathbf{P}, \iota_{init}, AP, L)$, in which

- *S is a countable, non-emty set of states*
- *$\mathbf{P} : S \times S \to [0,1]$ is the transition probability function such that*
$$\forall s \in S : \sum_{s' \in S} \mathbf{P}(s, s') = 1$$
- *$\iota_{init} : S \to [0,1]$ is the initial distribution such that*
$$\sum_{s \in S} \iota_{init}(s) = 1$$
- *AP is a set of atomic propositions.*
- *$L : S \to 2^{AP}$ is the labelling function on states.*

Example 2.2 (Knuth-Yao die)
Knuth-Yao die to simulate a 6-faced die by a fair coin. In this Knuth's die DTMC, there are 6 BSCCs, each of them represents an outcome of a die tossing. Image taken from [24]



Definition 2.2.1 (Strongly Connected Component)
Let $\mathcal{M} = (S, \mathbf{P}, \iota_{init}, AP, L)$ a DTMC. A subset $S' \subset S$ is strongly connected if and only if for every pair $s_1, s_2 \in S'$ there is a path between s_1 and s_2 which consists of only of state in S'. If there exist no $S'' \subseteq S$, such that $S \subset S''$ and S'' is strongly connected, then S' is a Strongly Connected Component, or SCC in short.

Definition 2.2.2 (Bottom Strongly Connected Component)
*Let $\mathcal{M} = (S, \mathbf{P}, \iota_{init}, AP, L)$ a DTMC and $S' \in S$ a Strongly Connected Component. S' is also a Bottom Strongly Connected Component, or BSCC for short, if and only if there exist no state $s \in S$
S' that is reachable from any state in S'. If $|S'| = 1$ then S' is a trivial BSCC. We denote $BSCC(\mathcal{M}) \in S$ is the set of all BSCCs of $\mathcal{M}$.*

Intuitively, BSCCs are arbsobing; once a path in a DTMC reaches a state in a BSCC, it visits all states in the BSCC infinitely often. It is proven by [3] that any run on a DTMC $\mathcal{M}$ ends in $BSCC(\mathcal{M})$ almost surely.

Theorem 1 (Long-run theorem)
Let $\mathcal{M} = (S, \mathbf{P}, \iota_{init}, AP, L)$ a DTMC.

$$Pr(\Diamond BSCC(\mathcal{M})) = 1$$

In this thesis we concern the *steady-state distribution* of a DTMC.

Definition 2.2.3 (Steady-state distribution)
Let $\mathcal{M} = (S, \mathbf{P}, \iota_{init}, AP, L)$ a DTMC and vector v_t be a transient state distribution

$$v_t = (Pr(X_t = s_1), \ldots, Pr(X_t = s_N)), s_0, \ldots, s_N \in S$$

A transient state distribution v of $\mathcal{M}$ is a steady-state distribution of $\mathcal{M}$ if and only if

$$v = vP$$

As a result from long-run theorem, if $BSCC(\mathcal{M}) \neq \emptyset$ then there exists a steady-state distribution $v = (Pr(X = s_1), \ldots, Pr(X = s_{|S|}))$, such that

$$\forall 1 \leq i \leq |S| : P[X = s_i] \neq 0 \Leftrightarrow s_i \in BSCC(\mathcal{M})$$

2.2.1 Continuous-time Markov chain

The discrete-time memoryless property can also be extended into continuous-time memoryless property. In continous-time, memoryless property has the following form

Definition 2.2.4 (Continuous-time memoryless property)
Let X be a continuous random variable of exponentially distribution. X has memoryless property if and only if

$$Pr(X > t + \delta | X > t) = Pr(X > \delta), \forall t, \delta \in \mathbb{R}_{\geq 0}$$

Based on continuous-time memory less property, we introduce the definition of *Continous-time Markov chain* [23].

Definition 2.2.5 (Continuous-time Markov chain)
A Continuous-time Markov chain (CTMC) $\mathcal{C}$ is a tuple $(S, \mathbf{P}, \mathbf{r}, \iota_{init}, AP, L)$

- *S is a countable, non-emty set of states*
- *$\mathbf{P} : S \times S \to [0, 1]$ is the transition probability function such that*

$$\forall s \in S : \sum_{s' \in S} \mathbf{P}(s, s') = 1$$

- *$\mathbf{r} : S \to \mathbb{R}_{>0}$ is the exit rate function such that*

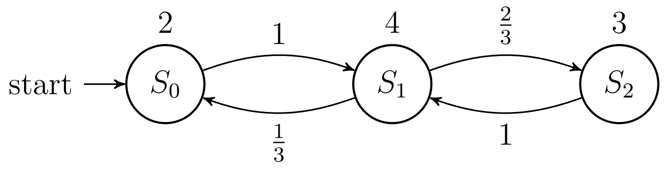
$$\forall s \in S : \sum_{s' \in S} \mathbf{P}(s, s') = 1$$

- *$\iota_{init} : S \to [0, 1]$ is the initial distribution such that*

$$\sum_{s \in S} \iota_{init}(s) = 1$$

- *AP is a set of atomic propositions*
- *$L : S \to 2^{AP}$ is the labelling function on states.*

Example 2.3 (CTMC)
An example of a CTMC with 3 states.



Continous-time Markov chain has a wide range of applications, especially in bioinformatics where chemical reaction network [12] [2]. However, the frameworks in this thesis apply for discrete-time Markov models, thus we do not use continuous-time Markov chain to model systems of interest directly. Instead, we do not use Continuous-time Markov models directly. Instead, we transform CTMCs into DTMCs through uniformization [24]

Definition 2.3.1 (CTMC Uniformization)
Let $\mathcal{C} = (S, \mathbf{P}, \mathbf{r}, \iota_{init}, AP, L)$ be a CTMC. We define the uniformization rate r such that

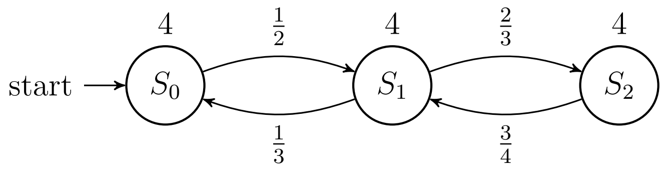
$$\forall s \in S : r \geq \mathbf{r}(s), r \in \mathbb{R}_{>0}$$

The uniformized CTMC $unif(r, \mathcal{C}) = (S, \bar{\mathbf{P}}, \bar{\mathbf{r}}, \iota_{init}, AP, L)$ such that

$$\forall s \in S : \bar{\mathbf{r}}(s)$$

$$\forall s, s' \in S : \bar{\mathbf{P}}(s, s') = \begin{cases} \frac{\mathbf{r}(s)}{r}\mathbf{P}(s, s') & \textit{if } s \neq s' \\ \\ \frac{\mathbf{r}(s)}{r}\mathbf{P}(s, s') + 1 - \frac{\mathbf{r}(s)}{r} & \textit{if } s = s' \end{cases}$$

Example 2.4 (Uniformized CTMC)
We uniformize the CTMC in Example 2.3 by uniformization rate $r = 4$.



It has been shown by Katoen [23] that uniformization preserves the transient probability distributions. Furthermore, in this thesis we concern steady state data and state property, thus uniformizing exit rate does not affect the validity of our constructed frameworks.

2.5 Property specification

2.5.1 Probabilistic Computational Tree Logic

Model checking verifies a formalism of a system *(model)* against a property of interest. We formalize a property by a *temporal logic*, specifically *Probabilistic Computational Tree Logic* (or *PCTL*). Firstly introduced by Clarke et al. [7], PCTL is widely used in model checking of discrete-time stochastic models and supported by most probabilistic model checking tools [10], [26].

Definition 2.5.1 (PCTL)
The syntax of PCTL consists of state formulas and path formulas.

- *State formulas are defined over AP*

$$\Phi ::= \mathit{true} \mid a \mid \Phi \mid \Phi_1 \wedge \Phi_2 \mid \Phi_1 \vee \Phi_2 \mid P_J(\phi)$$

where $a \in AP$, ϕ is a path formula, and $J \subseteq [0,1]$ is an interval.

- *Path formulas*

$$\phi ::= \bigcirc\Phi \mid \Phi_1 \mathsf{U} \Phi_2 \mid \Phi_1 \mathsf{U}^{\leq n} \Phi_2$$

where Φ, Φ_1, Φ_2 are state formulas, and $n \in \mathbb{N}$.

PCTL properties is applicable on discrete-time stochastic models such as DTMC, as the times between state transitions are uniform. In a DTMC, a PCTL state formula is verified at each state, while a PCTL path formula is verified through a trace from an execution path.
The algorithm to model check DTMC against PCTL properties is described in detail in Katoen [3]. Given a DTMC $\mathcal{M}$ and a PCTL property Φ, general algorithm for checking $\mathcal{M} \models \Phi$ has complexity of polynomial to $|\mathcal{M}|$ and linear to $|\Phi|$ [23].

Theorem 2 (Complexity of checking a DTMC against a PCTL formula.)
For finite DTMC $\mathcal{M}$ and PCTL state-formula Φ,the PCTL model-checking problem can be solved in time

$$\mathcal{O}(poly(size(D) \cdot n_{max} \cdot |\Phi|)$$

where

$$n_{max} = \begin{cases} max(n|(\Psi_1 \mathsf{U}^{\leq n} \Psi_2) \quad occurs\ in \quad \Phi) \\ 1\ if\ \Phi\ contains\ no\ bounded\ until\ property \end{cases}$$

2.5.2 State exlosion problem

The soundness of the model checking relies heavily on how the system is modeled. In fact, the model checking is only as sound and valid as the model.

1. Which formalism is used?
2. How the system is encoded into states and transitions?

For example, we consider a distributed software system, in which a *global state* is a composition of

1. values of all variables, and
2. states of all communication channels.

It is obvious that the number of possible states grows exponentially as more variables and communication channels are added to the system.
State-explosion problem occurs when the size of the system state space grows exponentially as the number of state variables in the system increases [6]. As discussed before, the complexity of model checking a PCTL property against a DTMC model is polynomial to the DTMC's state-space. However, the state-explosion problem renders model checking computationally expensive. One possible way to cope with state-explosion problem and to reduce the computational cost is to use *statistical model checking.*

2.6 Statistical Model checking

Statistical model checking is a simulation-based approach to model check a stochastic model $\mathcal{D}$ against a PCTL property Φ. The essential concept of probabilistic model checking is to simulate N traces from $\mathcal{M}$, verify if each trace satisfies Φ, then estimate probability $P(\mathcal{M} \models \Phi)$ by a statistical, frequentist approach.
In statistical model checking of, different methods are applied to *quantitative* and *qualitative* questions.[1] Given a stochastic model $\mathcal{M}$ and a property Φ, statistical model checking solves the following problems:

1. **Quantitative**: Estimate the probability $p = Pr(\mathcal{M} \models \Phi)$. In other words, it checks $\mathcal{M}$ the property

$$P_{=?}(\Phi)$$

2. **Qualitative**: Decide if $p = Pr(\mathcal{M} \models \Phi)$ is greater or less than a threshold ϵ. In other words, it checks $\mathcal{M}$ the property

$$P_J(\Phi)$$

   where $J \subseteq [0, 1]$ is an interval.

2.6.1 Statistical model checking of quantitative properties.

Given an approximation ϵ and a confidential level α, we estimate $\hat{p}$ as an estimation of p such that

$$Pr(|p - \hat{p}| \leq \epsilon) \geq 1 - \alpha$$

How many simulations must be performed? As verifying a simulation trace against a reachability property Φ is Bernoulli trial (satisfied or not satisfied), the number of simulation N can be estimated using different bounds. Chernoff [5] presents Chernoff inequality to estimate N for Bernoulli trials. Hoeffding [18] later extends Chernoff inequality to general cases.

[1] `https://www-verimag.imag.fr/Statistical-Model-Checking-814.html#nb3`

Let $Sat(N)$ be number of satisfying trace in N sampled traces. By applying Chernoff-Hoeffding inequality we obtain

$$P(|\frac{Sat(N)}{N} - p| > \epsilon) \leq 2\exp\frac{-N\epsilon^2}{4}$$
$$\Leftrightarrow \qquad P[|\frac{Sat(N)}{N} - p| \leq \epsilon] \geq 1 - 2\exp\frac{-N\epsilon^2}{4}$$

Replace $\alpha = 2\exp\frac{-N\epsilon^2}{4}$ and $\hat{p} = \frac{Sat(N)}{N}$, we have

$$P[|\hat{N} - p| \leq \epsilon] \geq 1 - 2\alpha$$
$$\Leftrightarrow \qquad N \geq 4\frac{\log\frac{2}{\alpha}}{\epsilon^2}$$

The estimation algorithm is described in detail in [1].

Algorithm 1 Statistical Model Checking, APMC method.

Input:

- $\mathcal{D}$: a DTMC
- α, ϵ: confidence level and approximation, respectively.
- $\Phi = P_{=?}(\varphi)$: a PCTL property in $\mathcal{D}$

Output: $\hat{p}$: an estimation of p for

$$p = Pr(\mathcal{D} \models \varphi)$$

```
1: procedure SMC-APMC
2:     $N \leftarrow 4\frac{\log\frac{2}{\alpha}}{\epsilon^2}$
3:     $A \leftarrow 0$
4:     $i \leftarrow 1$
5:     while $i \leq N$ do
6:         Simulate a trace $t$ from $\mathcal{D}$ by discrete-event simulation.
7:         if $t \models \phi$ then
8:             $A \leftarrow A + 1$
9:         end if
10:    end whilereturn $\frac{A}{N}$
11: end procedure
```

The Chernoff-Hoeffding gives more close estimation of N, thus it helps reducing the computational cost, while still maintain estimation accuracy. In case Φ has probabilistic bound, we use hypothesis test as the outcome of checking $\mathcal{D}$ against Φ is Boolean.

2.6.2 Statistical model checking of qualitative properties.

Wald [36] introduces Sequential Probability Ratio Test (SPRT) to perform hypothesis test on sequential statistical analysis, in which the sample size N is not know a-priori. SPRT updates a cummulative sum and stop when it has enough information to decide whether to statistically accept null hypothesis or alternative hypothesis.
Younes [37] introduces an application of Wald's SPRT on statistical model checking. Given a DTMC $\mathcal{D}$ and a PCTL property Φ with probabilistic bound $\Phi = P_J(\varphi)$. Without loss of generality, we asume that $\Phi = P_{\geq p}(\varphi), p \in [0, 1]$. SPRT method tests the following null hypothesis and alternative hypothesis with approximation width ϵ:

$$\begin{aligned} H_0 &: \hat{p} \geq p + \epsilon \\ H_0 &: \hat{p} < p - \epsilon \end{aligned}$$

Among N simulated traces from $\mathcal{D}$, the probability to have A satisfying traces given probability of success in a single Bernoulli trial is p:

$$P(A|p) = \binom{N}{A} \cdot p^A (1-p)^{N-A}$$

At each simulation i, SPRT compute the likelihood ratio based on accumulated number of success trials:

$$R_i = \frac{P(A_i|p+\delta)}{P(A|p-\delta)}$$

Let α and β be error of type I and type II, respectively.

$$\begin{aligned} \alpha &= Pr(\mathcal{D} \models \Phi|\text{Accept} \quad H_1) \\ \beta &= Pr(\mathcal{D} \not\models \Phi|\text{Accept} \quad H_0) \end{aligned}$$

We compute bounds for accepting or rejecting null hypothesis

$$p_0 = \frac{\beta}{1-\alpha}$$
$$p_1 = \frac{\alpha}{1-\beta}$$

Wald's sequential probabilistic ratio test allows the algorithm to terminate as soon as simulated traces reach either of the two bounds p_0 or p_1.

Algorithm 2 Statistical Model Checking, SPRT method

Input:

- $\mathcal{D}$: a DTMC
- α, β: probability of type I and type II error respectively.
- ϵ: approximation width.
- $\Phi = P_{\geq p}(\varphi)$: a PCTL property in $\mathcal{D}$

Output: Decide whether $(\mathcal{D} \models \Phi)$

```
1:  procedure SMC-SPRT
2:      $p_0 = \frac{\beta}{1-\alpha}$
3:      $p_1 = \frac{\alpha}{1-\beta}$
4:      $N \leftarrow 0$
5:      $A \leftarrow 0$
6:      while True do
7:          $R_i \leftarrow \frac{P(A_i|p+\delta)}{P(A|p-\delta)}$
8:          if $R_i \geq p_1$ then
9:              Accept $H_1$
10:             Return
11:         else
12:             if $R_i \leq p_0$ then
13:                 Accept $H_0$
14:                 Return
15:             end if
16:         end if
17:     end while
18: end procedure
```

2.7 Parametric model

2.7.1 Parametric Discrete Time Markov chain

In order to generalize the model and encompass the unknown features of the interested system in DTMC, we introduce *parameters* into transition probabilities. In this thesis we assume that parameters' domain is $\mathbb{R}$.

Definition 2.7.1
Rational functions Let $\theta = \{x_1, \ldots, x_n\}$ be a variable; let $\mathbf{Pol}[\mathbf{x}]$ be the set of all polynomial functions over $\mathbf{x}$. A rational function $h(\mathbf{x})$ is defined as following.

$$h(x) := \frac{f(\mathbf{x})}{g(\mathbf{x})}, f, g \in \mathbf{Pol}[\mathbf{x}], g(\mathbf{x}) \neq 0$$

We denote $\mathbb{Q}(\mathbf{x})$ the set of all rational functions over $\mathbf{x}$.

With the set of parameters and rational functions being formally defined, we define parametric Discrete-time Markov chain based the definition on [21].

Definition 2.7.2 (Discrete Time Markov Chain)
A parametric Discrete-time Markov chain (pDTMC) $\mathcal{M}_\theta$ is a tuple $(S, \theta, \mathbf{P}, \iota_{init}, AP, L)$ where

- *S is a countable, non-emty set of states*
- *$\theta \in \mathbb{R}^n, n \in \mathbb{N}$ as the set of parameters.*
- *$\mathbf{P} : S \times S \to \mathbb{Q}(\mathbf{x})$ is the transition probability function such that*

$$\forall s \in S : \sum_{s' \in S} \mathbf{P}(s, s') = 1$$

- *$\iota_{init} : S \to [0, 1]$ is the initial distribution such that*

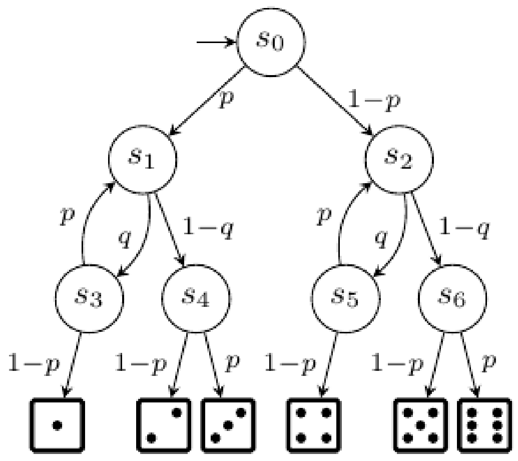
$$\sum_{s \in S} \iota_{init}(s) = 1$$

- *AP is a set of atomic propositions*
- *$L : S \to 2^{AP}$ is the labelling function on states.*

A parametric DTMC instantiates a non-parametric DTMC by an assignment of its variable.

Definition 2.7.3
Parameter value Let $\mathcal{M}_\theta = (S, \theta, \mathbf{P}, \iota_{init}, AP, L)$ be a pDTMC, $\theta = \{\theta_1, \ldots, \theta_n\}$. A value of θ is a map $v : \theta \rightarrow \mathbb{R}^n$. A value v instantiates a non-parametric Discrete-time Markov chain if $f\mathbf{v}(\theta)$ evaluates to a real value for all $f \in \mathbf{P}$.

Example 2.8 (Parametric Knuth-Yao die)
A DTMC modelling Knuth-Yao die to simulate a 6-faced die by two unfair coins with probability of one side p and q. Image taken from [24].



2.8.1 Parameter synthesis of pDTMC

With a pDTMC represents a class of DTMC, we concerns of instantiated DTMC which satisfy a certain property of interest.

Definition 2.8.1
Parameter synthesis (Katoen 2016)[24] Given a parametric DTMC $\mathcal{M}_\theta = (S, \theta, \mathbf{P}, \iota_{init}, AP, L)$ and a reachability property Φ, find a set of parameter values θ such that $\mathcal{M}_\theta \models \Phi$.

Katoen [23] summarizes the following methods on parameter synthesis of pDTMC:

1. *Computing symbolic reachability probabilities*: using states elimination to obtain symbolic rational function of a reachability property [9] [14].

2. *Candidate region generation and checking*: partition the parameter space into *safe* and *unsafe* regions using non-linear interval arithmetic [27].

3. *Parameter lifting* is another parameter space partitioning method; it introduces new transitions into the original pDTMC through *relaxation* and *substitution.* The procedure results in a non-parametric transition system with transition labels are bounds from given intervals. The region is then checked using candidate region generation and checking method.

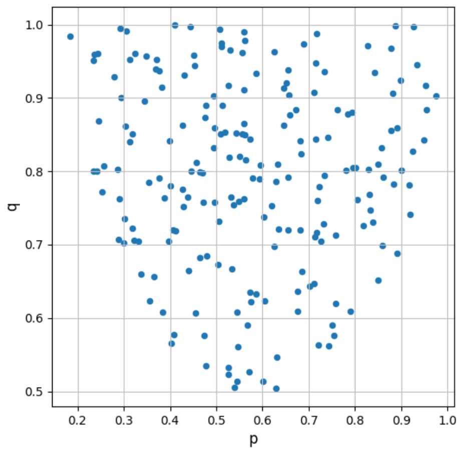
In this thesis we use only symbolic model checking [9].

Example 2.9
Parametric Knuth's die We continue the example with Knuth die model $\mathcal{M}_p$. Assume the

```
P(F "1") = (p^2*q+(-1)*p*q)/(p*q+(-1))
P(F "2") = ((p)^2 * (q+(-1)))/(p*q+(-1))
P(F "3") = (-1 * ((p) * (p+(-1)) * (q+(-1))))/(p*q+(-1))
P(F "4") = (-1 * (p^2*q+(-1)*p*q))/(p*q+(-1)*p+1)
P(F "5") = (p^2*q+(-2)*p*q+q)/(p*q+(-1)*p+1)
P(F "6") = (-1 * ((p+(-1))^2 * (q+(-1))))/(p*q+(-1)*p+1)
```

With the symbolic rational function $f_\Phi(\theta)$ of Φ obtained, we assign a parameter value to θ, then replace all symbolic parameters by their concrete value to check if $\mathcal{M}_\theta \models \Phi$.

Example 2.10
Given a pDTMC of Knuth die $\mathcal{M}_{(p,q)}, (p,q) \in [0,1] \times [0,1]$ and a reachability property $\Phi = P_{\geq 0.2}($`F "one"`*$)$, synthesize parameter (p,q) so that $\mathcal{M}_{(p,q)} \models \Phi$. A simple Monte-Carlo search on parameter space gives the following satisfying points:*



2.10.1 Summary

In this chapter we introduce the essential theoretical concepts on discrete-time models, probabilistic temporal logic, and parameter synthesis problem. However, the parameter synthesis methods presented exhaustively search on parameter space for satisfying parameter values. In the next chapter, we present Bayesian inference methods for later construction of data-informed parameter synthesis frameworks.

Chapter 3

Bayesian inference

We present essential concepts in Bayesian parameter inference and several methods to estimate posterior distribution. The methods range from posterior conjugations, in which tractability is guaranteed as we know the analytic form of both likelihood and prior distribution. Afterwards, we discuss different sampling algorithm to approximate the posterior distribution when no conjugations are available. We also present a likelihood-free method to exploit in the case that the analytical form of the likelihood is not achievable or is too complex to evaluate. The sampling algorithms presented in this chapter are the building block for the Bayesian frameworks that we present in this thesis.

3.1 Bayesian inference

3.1.1 Bayesian formula

Let D_{obs} be observed data. In statistical inference, we assume that the observed data has a probability distribution of unknown parameter θ, that is $D_{obs} \sim P(D_{obs}|\theta)$. There are two main approaches in statistical inference

1. Frequentist approach.

2. Bayesian approach.

In frequentist approach, the estimation of θ based on long-run property, that is, given a large enough sample size, expected value of parameter estimation $\hat{\theta}$ is equal to θ. Therefore, frequentist approach requires to gather a

large amount of data to deliver a close estimation $\hat{\theta}$. The main advantage of Bayesian approach over frequentist approach is that it require less data to obtain an estimation $\hat{\theta}$.
In Bayesian approach, we use the information gained from previously observed data *(beliefs)* to enhance the accuracy of the estimation of $\hat{\theta}$. The beliefs obtained from prior knowledge of model parameter θ is represented by *prior distribution* $\pi(\theta)$. We have the *likelihood* $P(D_{obs}|\theta)$ as the probability distribution over observed data, given parameter θ.

Definition 3.1.1
Bayes theorem

$$\pi(\theta|D_{obs}) = \frac{P(D_{obs}|\theta)\pi(\theta)}{\int_\theta P(D_{obs}|\theta)\pi(\theta)d\theta}$$

where

- $\int_\theta P(D_{obs}|\theta)\pi(\theta)d\theta$ *is the marginal distribution.*
- $\pi(\theta|D_{obs})$ *is the posterior distribution*

The essential part of Bayesian inference in statistic is to compute or estimate the posterior distribution. From the analytical form or the samples from the posterior distribution, we estimate the model parameter θ.

3.1.2 Bayesian parameter estimation

With posterior distribution $\pi(\theta|D)$ we estimate the parameter $\hat{\theta}$ using Bayesian posterior mean.

Definition 3.1.2
Bayesian posterior mean

$$\hat{\theta} = \mathbf{E}[\theta] = \int_\theta \theta\pi(\theta|D)d\theta$$

In case we have samples from posterior distribution, for example a trace T of parameter values $\theta_1, \ldots, \theta_|T|$ from Metropolis-Hastings algorithm, the discrete form of posterior mean is used:

$$\hat{\theta} = \mathbf{E}[\theta] = \sum_\theta \theta\pi(\theta|D)$$

Definition 3.1.3 (Bayesian Credible Set)
Set C is a $(1\alpha)100\%$ *credible set for the parameter θ if the posterior probability for θ to belong to C equals* (1α)*.*

$$P(\theta \in C|D) = \int_C \pi(\theta|D)d\theta = 1 - \alpha$$

In this thesis, we use by default 0.95 credible set, which corresponds to $\alpha = 0.05$

Definition 3.1.4 (Highest Posterior Density credible set)
Highest Posterior Density $(1 - \alpha)100\%$ *credible set (HPD for short) is the interval with minimum length over all Bayesian* $(1 - \alpha)100\%$ *Credible Set.*

In this research, the HPD is calculated using algorithm from *PyMC3* library [32]. For simplicity, we assume that in all cases which we concern, HPD is computed using the algorithm for unimodal distribution.

Algorithm 3 Compute Highest Posterior Density Interval

```
    Input: $S$ is samples from a distribution.
    Input: $0 \leq \alpha \leq 1$
    Output: HPD interval
1: procedure COMPUTE HPD($S$)
2:     Compute interval width $w = |S| * \alpha$
3:     Find modal (peak) of sample points.
4:     Return minimal interval of size $|S| - w$ which contains the modal.
5: end procedure
```

3.1.3 Selection of prior distribution

Theoretically, prior can be of any distribution family. However, a selection of prior distribution that is too different than the actual distribution of parameter can leads to a false propagation of beliefs and degrade inference results. It is suggested by [31] that in case of no prior knowledge exists to help the selection of prior distribution, Uniform distribution is preferable since it is less likely to propagate false beliefs to the inference.
A systematic inference to select prior distribution family and prior distribution parameter (hyperparameters) is possible with *Hierarchical Bayes Models* [1].

3.1.4 Estimation of posterior distribution

Posterior conjugation

Conjugated posteriors are special cases of Bayesian inference, in which the prior and posterior distribution belongs to the same family of distribution. When posterior conjugation is applicable, only the parameters of probability distribution function need to be re-estimated. Applying conjugated posterior when it is possible gives advantages:

- Tractability: we have analytical form of posterior distribution with only changes in its parameters.
- Computationally effective: updating model parameter is of linear time to the dimension of parameter.

We consider two conjugated posteriors as examples: Binomial-Beta and Dirichlet-Multinomial.

Lemma 3 (Binomial-Beta Conjugation)
Binomial distribution is conjugated to beta distribution.

Proof. The observed data $D = (x_1, \dots, x_n)$ is sampled from $Binomial(k, \theta)$ function

$$P(D|\theta) = \prod_{i=1}^{n} \binom{k}{x_i} \theta^{x_i} (1-\theta)^{k-x_i}$$

The parameter θ is of $Beta(\alpha, \beta)$ distribution

$$\pi(\theta) = \theta^{\alpha-1}(1-\theta)^{\beta-1}$$

We obtained:

$$\begin{aligned}
\pi(\theta|D) &\sim P(D|\theta)\pi(\theta) \\
&\sim \theta^{\sum_{i=1}^{n} x_i}(1-\theta)^{nk-\sum_{i=1}^{n} x_i}\theta^{\alpha-1}(1-\theta)^{\beta-1} \\
&= \theta^{\alpha-1+\sum_{i=1}^{n} x_i}(1-\theta)^{\beta-1+nk-\sum_{i=1}^{n} x_i}
\end{aligned}$$

Thus, the posterior is $Beta(\alpha + \sum_{i=1}^{n} x_i, \beta + nk - \sum_{i=1}^{n} x_i)$ □

Generalize this conjugation, we also have Multinomial-Dirichlet conjugation.

Lemma 4 (Multinomial-Dirichlet Conjugation)
Multinomial distribution is conjugated to Dirichlet distribution.

Proof. The observed data $D = (x_1, \dots, x_n)$ is sampled from $Multinomial(n; \theta_1, \dots, \theta_n)$ function

$$P(x_1, \dots, x_n | N, \theta_0, \dots, \theta_n) = \frac{n!}{x_1! \dots x_n!} \prod_{i=1}^{n} \theta_i^{x_i}$$

The parameter $(\theta_1, \dots, \theta_n)$ is $Dirichlet(\alpha_1, \dots, \alpha_n)$

$$\pi(\theta_1, \dots, \theta_n) = \frac{1}{\mathbf{B}(\alpha_1, \dots, \alpha_n)} \prod_{i=1}^{n} \theta_i^{\alpha_i - 1}$$

We obtain

$$\begin{aligned} \pi(\theta_1, \dots, \theta_n | D) &\sim P(D|\theta)\pi(\theta) \\ &\sim \prod_{i=1}^{n} \theta_i^{x_i} \prod_{i=1}^{n} \theta_i^{\alpha_i - 1} \\ &\sim \prod_{i=1}^{n} \theta_i^{\alpha_i - 1 + \sum_{i=1}^{n} x_i} \end{aligned}$$

Thus, the posterior is $Dirichlet(\alpha_1 + x_1, \dots, \alpha_n + x_n)$ □

More detailed description in these cases can be found in [35] and [4]. We summarize the necessary results in the following table:

Likelihood	Prior	Posterior parameters
$Binomial(n, k)$	$Beta(\alpha, \beta)$	$\alpha' = \alpha + \sum_{i=1}^{n} x_i$ $\beta' = \beta + nk - \sum_{i=1}^{n} x_i$
$Multinomial(n; \theta_1, \dots, \theta_n)$	$Dirichlet(\alpha_1, \dots, \alpha_n)$	$\alpha'_i = \alpha_i + x_i, 1 \leq i \leq n$

However, posterior conjugation is applicable to a subset of prior and likelihood functions. In Bayesian inference, it is usual that the posterior distribution has no analytical form or its analytical form is difficult to directly sample from. In these cases, we can several different sampling and optimization methods to approximate the posterior distribution. In the following section we discuss different approaches for posterior distribution approximation:

- Markov chain Monte-Carlo.
- Sequential Monte-Carlo.
- Approximate Bayesian Computation.

Markov chain Monte-Carlo

In case the posterior distribution has no analytical form or its analytical form is difficult to sample from directly, we use *Metropolis-Hastings* algorithm (*MH* in short).
invented by Metropolis [28] and later generalized by Hastings [16] Metropolis-Hastings algorithm is a *Monte Carlo Markov Chain* algorithm. In its essential, Metropolis-Hastings algorithm draws sample from an unknown distribution. Using the MH algorithm, we can estimate the parameter by posterior mean, without knowing the analytical form of posterior distribution itself.

Algorithm 4 Metropolis-Hastings Algorithm

Input:

- Model $\mathcal{M}_\theta$
- D_{obs}: observation data
- Likelihood function $P(D|\theta)$
- $\pi(\theta)$: prior distribution
- Transition kernel $Q(\theta^t|\theta^{t-1})$
- N number of particles.

Output:

- $(\theta_1, \ldots, \theta_N)$ sample of N particles
- $(w_1, \ldots, w_N)$ corresponding likelihoods.

```
1: procedure METROPOLIS-HASTINGS
2:     Draw $\theta_0$ from $\pi(\theta)$
3:     $i \leftarrow 0$
4:     while maxIteration not reached do
5:         $L \leftarrow P(D|\theta)$
6:         Draw a point $\theta'$ from the proposal distribution.
7:         $L' \leftarrow P(D|\theta')$
8:         if $\ln(L') - \ln(L) > 0$ then
9:             Add $\theta'$ to $Trace$
10:            $\theta = \theta'$
11:        else
12:            Draw a random number $x$ from $Uniform(0, 1)$
13:            if $x \leq \xi$, ($\xi$ very small, e.g $10^{-8}$) then
14:                Add $\theta'$ to $Trace$ (avoiding local maxima)
15:                $\theta = \theta'$
16:            end if
17:        end if
18:    end while
19: end procedure
```

The likelihood function can be implemented as log-likelihood to avoid underflow error. Advantages of Metropolis-Hastings are

- Parameter transition only needs the computation of likelihood function. Therefore, Monte Carlo Markov Chain can be used in general Bayesian inference, in which we are not guaranteed to have an analytical form of posterior.
- Computationally efficient; as marginal distribution is cancelled out, and likelihood can be replaced by log-likelihood, Metropolis-Hastings simplifies the computation of Bayes formula and avoid infinitesimal values.
- Simple to implement.

Disadvantages of Metropolis-Hastings are

1. Particle in Metropolis-Hastings algorithm moves in a linear Markov chain; it is highly probable to be stuck in a local maximum or minimum.
2. Not parallelizable; since there is only one linear chain, and current step depends on previous step, Metropolis-Hastings algorithm does not scale up to multi-processors.

The next algorithm, *Sequential Monte-Carlo*, address the issues of Metropolis-Hastings.

Sequential Monte-Carlo

Sequential Monte-Carlo method is firstly proposed by [11]. Instead of having one particle moving in its parameter space, Sequential Monte-Carlo estimates by using N particles moving independently. Sequential Monte-Carlo uses starting by a *pertubation kernel.*

Therefore Sequential Monte-Carlo method has a significant advantage of easily parallelizable. Comparision between [8] good for multimodal

Algorithm 5 Sequential Monte-Carlo Algorithm

Input:

- Model $\mathcal{M}_\theta$
- D_{obs}: observation data
- $\pi(\theta)$: prior distribution
- $P(D|\theta)$: Likelihood function.
- Transition kernel $Q(\theta^t|\theta^{t-1})$
-
- Transition kernel $Q(\theta^t|\theta^{t-1})$
- N number of particles.

Output:

- $(\theta_1, \ldots, \theta_N)$ sample of N particles
- $(w_1, \ldots, w_N)$ corresponding likelihoods.

```
procedure SEQUENTIAL-MONTE-CARLO
    $i \leftarrow 1$
    while $i \leq N$ do                                        ▷ SMC initialization
        Draw $\theta$ from $\pi(\theta)$
        $\theta_i \leftarrow \theta$
        $w_i \leftarrow P(D_{obs}|\theta_i)$
        $i \leftarrow i + 1$
    end while
    $t \leftarrow 1$
    while $t \leq M$ do
        $i \leftarrow 1$                                       ▷ SMC correction step
        while $i \leq N$ do
            $w'_i \leftarrow \frac{w_i}{\sum_{i=1}^{N} w_i}$
        end while
        Sample with replacement $(\theta'_1, \ldots, \theta'_N)$          ▷ SMC selection step
          from $(\theta_1, \ldots, \theta_N)$ with probabilities $(w'_1, \ldots, w'_N)$
        $(\theta_1, \ldots, \theta_N) \leftarrow (\theta'_1, \ldots, \theta'_N)$
        $i \leftarrow 1$
        while $i \leq N$ do                                    ▷ SMC pertubation step
            Draw $\hat{\theta}_i^t$ from $F_t(\theta^t|\theta_1^{t-1}, \ldots, \theta_N^{t-1}), 1 \leq t \leq M$
            $(\theta_1^*, \ldots, \theta_{N_{MH}}^*), (w_1^*, \ldots, w_{N_{MH}}^*) \leftarrow Metropolis - Hastings(\hat{\theta}_i^t)$
            $\theta_i \leftarrow \theta_{N_{MH}}^*$
            $w_i \leftarrow w_{N_{MH}}^*$
        end while
    end while
```

Selection of kernel function for SMC is mentioned in [33].
Advantages of Sequential Monte-Carlo algorithm:

- Approximate multimodal distributions: since Sequential Monte-Carlo consists of N particles moving independently and later selected with replacement, it is less likely to fall into a local maximum or minimum.
- Parallelizable: Sequential Monte-Carlo has trivially data-parallelism, in contrast to Metropolis-Hastings where no parallelization is possible.

However, Sequential Monte-Carlo also has disadvantages:

- Selection of pertubation and transition kernel is not trivial.
- More difficult to implement.

Approximate Bayesian Computation

The methods mentioned before is used with an assumption that the likelihood $P(D_{obs}|\theta)$ has an analytical form; the analytical can be evaluated without introducing computational burden. However for situations in which the likelihood has no analytical form, or the analytical form is expensive to be evaluated, we use a class of *likelihood-free* methods. Likelihood-free methods in Bayesian inference estimates the likelihood $P(D_{obs}|\theta)$, estimate it or replace it by other measures. Approximate Bayesian Computation is a widely used likelihood-free method for approximating posterior distribution. Instead of estimating the likelihood $P(D|\theta)$ directly, we sample a observable data set $\hat{D}$ and define a distance measure $\delta(D, \hat{D})$. Approximate Bayesian Computation accepts a set of tuples $(\hat{\theta}, \hat{D})$, each satisfies that $\delta(D_{obs}, D_{sim}) < \epsilon, \epsilon \in \mathbf{R}_{\leq 0}$.

Algorithm 6 Approximate Bayesian Computation

Input:

- Model $\mathcal{M}_\theta$
- D_{obs}: observation data
- $\pi(\theta)$: prior distribution
- $\delta(D_{sim}, D_{obs})$: distance function between two set of data simulated by $\mathcal{M}_\theta$
- *epsilon*: distance threshold
- N number of particles.

Output:

- $(\theta_1, \ldots, \theta_N)$: N sampled particles.
- $(w_1, \ldots, w_N)$: corresponding weights of sampled particles.

```
1: procedure Approximate-Bayesian-Computation
2:     Select a proposal distribution $\pi(\theta)$
3:     $i \leftarrow 1$
4:     while $i \leq N$ do
5:         Draw a random particle $\theta$ from $\pi(\theta)$
6:         Simulate data $D_{sim}$ from $\mathcal{M}_\theta$
7:         if $d = \delta(D_{sim}, D_{obs}) < \epsilon$ then
8:             $\theta_i \leftarrow \theta$
9:             $w_i = d$
10:        end if
11:    end while
12:    Return $(\theta_1, \ldots, \theta_N)$, $(w_1, \ldots, w_N)$
13: end procedure
```

Advantages of Approximate Bayesian Computation:

- Likelihood-free: applicable when the likelihood has no analytical form or there is no likelihood.
- Easy to implement.

Disavantages of Approximate Bayesian Computation:

- Which distances are proper to measure two data set? In the other word, is measuring distance a proper method to approximate likelihood?
- How to select a threshold so that the likelihood is closely approximated?

3.2 Summary

We introduces basic concepts on Bayesian parameter inference and posterior estimation. Since the posterior distributions normally have no analytical form, the presented sampling and optimization methods which are essentials to posterior estimation. In the following chapter we propose a data-driven approach for parameter synthesis combining Approximate Bayesian computation, Sequential Monte Carlo, and Statistical Model Checking.

Chapter 4

Related works

The frameworks presented in this thesis are based on ABC-SMC framework [30] and ABC-(SMC)2 [29] by Molyneux et al. However, the ABC-SMC and ABC-(SMC)2 frameworks synthesize parameters for CTMC and check the CTMC model agains CSL property. In parametric DTMC, since the symbolic rational function of PCTL property is obtainable, we based on Del Moral [11] and Daviet [8] to construct an algorithm based on evaluation of symbolic rational function, then benchmark it againts the approach based on only simulation.

The theoretical background of model checking DTMC is presented by Baier et al [3]. An introduction to parametric DTMC and current methods on parameter synthesis is presented by the tutorial by Katoen [24]. More in-depth surveys and discoveries on parametric model checking and parameter synthesis is presented by Junges [22] and [19]. Polgreen [31], and Haesaert [13] present Bayesian approach on data-informed parameter synthesis and model verification. Jha [20] pioneered the first algorithm for Bayesian approach on Statistical Model Checking, namelyBayesian Sequential Hypothesis Testing.

Monte-Carlo sampling algorithms used in this thesis are presented by Metropolis [28], Del Moral [11], and Toni [34]. A comparision between different Monte-Carlo sampling algorithms, including Markov-chain Monte-Carlo, Sequential Monte-Carlo, Hamiltonian Monte-Carlo, and Hamiltonian Sequential Monte-Carlo is presented in [8]. Silk [33] discussed different approaches on the pertubation kernel selection of Sequential Monte-Carlo and Sequential Monte-Carlo with Approximate Bayesian Computation algorithms.

The model checking step in the frameworks presented by this thesis are implemented using Storm model checker [17]. Storm provides well documented and easy to use APIs to embed model checking to software projects programmatically. However, Storm does not support Statistical Model Checking. Thus, Statistical Model Checking step in simulation-based frameworks is implemented using PRISM [26].

Chapter 5

Bayesian frameworks for parameter synthesis.

We present frameworks for data-informed parameter synthesis of pDTMC. The frameworks are designed to synthesize a set of parameter values so that for each value, the instantiated model satisfies the interested property. Formally, given a pDTMC model $\mathcal{M}_\theta$, a PCTL property Φ, and observed data D_{obs}, the frameworks synthesize a set of N parameters $(\theta_1, \ldots, \theta_N)$ such that

$$\forall i \in [1, N] : \mathcal{M}_{\theta_i} \models \Phi$$

There are two frameworks designed towards two different use cases:

1. Symbolic rational functions are available for both BSCC reachability properties and Φ.

2. Only simulation and statistical model checking are possible.

5.1 Generic framework

Algorithm 7 Generic framework for Bayesian, Monte-Carlo parameter synthesis

Input:

- $\mathcal{M}_\theta$: parametric Discrete-Time Markov chain of parameter θ
- Φ: bounded reachability property of interest.
- D_{obs}: observed data.
- N: number of particles.

Output:

- $(\theta_1, \ldots, \theta_{N_{MH}})$: N_{MH} sampled particles.
- $(w_1, \ldots, w_{N_{MH}})$: corresponding weights of sampled particles.

```
1: procedure Generic-Bayesian-Monte-Carlo
2:     $i \leftarrow 1$
3:     while $i \leq N$ do
4:         Sample $\theta$ by Sequential Monte-Carlo sampling algorithm.
5:         Verify instantiated model $\mathcal{M}_\theta$ against $\Phi$
6:         if $\mathcal{M}_\theta \models \Phi$ then
7:         end if
8:     end while
9: end procedure
```

5.1.1 Selection of pertubation kernel

An analysis on the seletion of pertubation kernel is presented by [33]. For simplicity and avoiding the propagation of false beliefs, we select uniform, component-wise pertubation kernel. In the Metropolis-Hastings step when the analytical form of likelihood is available, we implement Single Component Metropolis-Hastings. The transition kernel $Q(\theta^t|\theta^{t-1})$ is selected component-wise, identical to the pertubation kernel, however the minimum and maximum values are extracted from local Metropolis-Hasting trace.

5.2 Symbolic computation based frameworks

As we have analytical form for both target property and likelihood function, the framework is designed identical to the original Sequential Monte-Carlo algorithm [11]. The only difference is that our framework only accept parameter values that instantiate satisfying concrete DTMC models. As in Sequential Monte-Carlo sampler, the first step is to define Metropolis-Hastings step for each independent particle θ.

Algorithm 8 Markov chain Monte-Carlo with rational functions

Input:

- $\mathcal{M}_\theta$: parametric Discrete-Time Markov chain of parameter θ
- Φ: bounded reachability property of interest.
- $\pi(\theta)$: prior distribution on θ.
- N_{MH}: length of particle trace.
- $Q(\theta^t|\theta^{t-1})$: transition kernel.
- D_{obs}: observed data.
- $P(D_{obs}|\theta)$: likelihood function.

Output:

- $(\theta_1, \ldots, \theta_{N_{MH}})$: N_{MH} sampled particles.
- $(w_1, \ldots, w_{N_{MH}})$: corresponding weights of sampled particles.

```
1: procedure RF-MCMC
2:     $sat \leftarrow False$
3:     while $sat = False$ do
4:         Draw $\theta_{cand}$ from $\pi(\theta)$
5:         Evaluate $val \leftarrow RF_\Phi(\theta)$
6:         if $val$ satisfies the boundary of $\Phi$ then
7:             $sat \leftarrow True$
8:         end if
9:     end while
10:    $\theta_1 \leftarrow \theta_{cand}$
11:    $w_1 \leftarrow \ln(P(D_{obs}|\theta_{cand}))$
```

Afterwards, we implement the main Sequential Monte-Carlo algorithm for the case where rational functions are obtainable.

```
12:     $i \leftarrow 2$
13:     **while** $i \leq N_{MH}$ **do**
14:         $sat \leftarrow False$
15:         **while** $sat = False$ **do**
16:             Draw $\theta_{cand}$ from $Q(\theta'|\theta_{i-1})$
17:             Evaluate $val \leftarrow RF_{\Phi}(\theta)$
18:             **if** $val$ satisfies the boundary of $\Phi$ **then**
19:                 $sat \leftarrow True$
20:             **end if**
21:         **end while**
22:         **if** $\ln(P(D_{obs}|\theta_{cand})) - \ln(P(D_{obs}|\theta_{i-1})) > 0$ **then**
23:             $\theta_i \leftarrow \theta_{cand}$
24:             $w_i \leftarrow \ln(P(D_{obs}|\theta_{cand}))$
25:             $i \leftarrow i + 1$
26:         **else**
27:             Draw a random number $u$ from $Uniform(0, 1)$
28:             **if** $u \leq \xi$, ($\xi$ small, e.g $10^{-2}$) **then**
29:                 $\theta_i \leftarrow \theta_{cand}$
30:                 $w_i \leftarrow \ln(P(D_{obs}|\theta_{cand}))$
31:                 $i \leftarrow i + 1$
32:             **end if**
33:         **end if**
34:     **end while**
35:     Return $(\theta_1, \ldots, \theta_{N_{MH}})$, $(w_1, \ldots, w_{N_{MH}})$
36: **end procedure**
```

Algorithm 9 Sequential Monte-Carlo with rational functions

Input:

- $\mathcal{M}_\theta$: parametric Discrete-Time Markov chain of parameter θ
- Φ: bounded reachability property of interest.
- $\pi(\theta)$: prior distribution on θ.
- N: number of particles in the Sequential Monte-Carlo trace.
- M: number of pertubation kernels
- $F_t(\theta^t|\theta_1^{t-1}, \dots, \theta_N^{t-1}), 1 \leq t \leq M$: pertubation kernels
- N_{MH}: number of particles in each Metropolis-Hastings step.
- $Q_t(\theta^t|\theta^{t-1}), 1 \leq t \leq N_{MH}$: transition kernel for Metropolis-Hastings step.
- D_{obs}: observed data for Bayesian inference or its summary statistic S_{obs}
- $P(D_{obs}|\theta)$: likelihood function.

Output:

- $(\theta_1, \dots, \theta_N)$: N sampled particles.
- $(w_1, \dots, w_N)$: corresponding weights of sampled particles.

```
1: procedure RF-SMC
2:     $i \leftarrow 1$
3:     while $i \leq N$ do                              ▷ SMC initialization
4:         Draw $\theta$ from $\pi(\theta)$
5:         $\theta_i \leftarrow \theta$
6:         $w_i \leftarrow P(D_{obs}|\theta_i)$
7:         $i \leftarrow i + 1$
8:     end while
```

```
 9:     $t \leftarrow 1$
10:     while $t \leq M$ do
11:         $i \leftarrow 1$                                          ▷ SMC correction step
12:         while $i \leq N$ do
13:             $w'_i \leftarrow \frac{w_i}{\sum_{i=1}^{N} w_i}$
14:         end while
15:         Sample with replacement $(\theta'_1, \ldots, \theta'_N)$          ▷ SMC selection step
16:             from $(\theta_1, \ldots, \theta_N)$ with probabilities $(w'_1, \ldots, w'_N)$
17:         $(\theta_1, \ldots, \theta_N) \leftarrow (\theta'_1, \ldots, \theta'_N)$
18:         $i \leftarrow 1$
19:         while $i \leq N$ do                               ▷ SMC pertubation step
20:             Draw $\hat{\theta}_i^t$ from $F_t(\theta^t|\theta_1^{t-1}, \ldots, \theta_N^{t-1}), 1 \leq t \leq M$
21:             $(\theta_1^*, \ldots, \theta_{N_{MH}}^*), (w_1^*, \ldots, w_{N_{MH}}^*) \leftarrow RF - MCMC(\hat{\theta}_i^t)$
22:             $\theta_i \leftarrow \theta_{N_{MH}}^*$
23:             $w_i \leftarrow w_{N_{MH}}^*$
24:         end while
25:     end while
26:     Return $(\theta_1, \ldots, \theta_N)$, $(w_1, \ldots, w_N)$
27: end procedure
```

5.3 Simulation based frameworks.

Without the availability of analytical form of observational and interested properties, we face the following obstacles:

- **Absence of likelihood functions:** As the rational functions for properties are not available, we do not have the analytical form of likelihood. The abscence of likelihood suggests to exploit *likelihood-free methods*. In this framework we use *Approximate Bayesian Computation* in combination with *Sequential Monte-Carlo method*.

- **Absence of rational function for verification of bounded reachability property:** the satisfaction of an instantiated model to a bounded path property cannot be computed. In the case that the number of states is too large, we use *Statistical Model Checking*.

For this case we present Statistical Model Checking, Approximate Bayesian Computation - Sequential Monte-Carlo method *SMC-ABC-SMC* framework.

Algorithm 10 Sequential Monte-Carlo with Approximate Bayesian Computation and Statiscal Model Checking

Input:

- $\mathcal{M}_\theta$: parametric Discrete-Time Markov chain of parameter θ
- Φ: bounded reachability property of interest.
- $\pi(\theta)$: prior distribution on θ.
- N: number of particles in the Sequential Monte-Carlo trace.
- M: number of pertubation kernels
- $F_t(\theta^t|\theta_1^{t-1}, \ldots, \theta_N^{t-1}), 1 \leq t \leq M$
- N_{MH}: number of particles in each Metropolis-Hastings step.
- $Q_t(\theta^t|\theta^{t-1}), 1 \leq t \leq N_{MH}$: transition kernel for Metropolis-Hastings step.
- D_{obs}: observed data for Bayesian inference or its summary statistic S_{obs}
- ϵ: threshold for Approximate Bayesian Computation.
- δ, α: indifference and α-level for Statistical Model Checking using SPRT method.

Output:

- $(\theta_1, \ldots, \theta_N)$: N sampled particles.
- $(w_1, \ldots, w_N)$: corresponding weights of sampled particles.

```
1: procedure SMC-ABC-SMC
2:     $i \leftarrow 1$
3:     while $i \leq N$ do                                    ▷ SMC initialization
4:         Draw $\theta$ from $\pi(\theta)$
5:         $\theta_i \leftarrow \theta$
6:         $w_i \leftarrow 1$
7:     end while
```

```
    $t \leftarrow 1$
    while $t \leq M$ do
        $i \leftarrow 1$                                          ▷ SMC correction step
        while $i \leq N$ do
            $w'_i \leftarrow \frac{w_i}{\sum_{i=1}^{N} w_i}$
        end while
        Sample with replacement $(\theta'_1, \ldots, \theta'_N)$          ▷ SMC selection step
          from $(\theta_1, \ldots, \theta_N)$ with probabilities $(w'_1, \ldots, w'_N)$
        $(\theta_1, \ldots, \theta_N) \leftarrow (\theta'_1, \ldots, \theta'_N)$
        $i \leftarrow 1$
        while $i \leq N$ do                                       ▷ SMC pertubation step
            $rejected \leftarrow True$
            while $rejected == True$ do
                $sat \leftarrow False$
                while $sat = False$ do
                    Draw $\hat{\theta}_i^t$ from $F_t(\theta^t | \theta_1^{t-1}, \ldots, \theta_N^{t-1}), 1 \leq t \leq M$
                    Do SPRT SMC on $\mathcal{M}_{\hat{\theta}^t}$ and $\Phi$
                    if $\mathcal{M}_{\hat{\theta}^t} \models \Phi$ then
                        $sat \leftarrow True$
                    end if
                end while
                $D_{sim} \leftarrow Simulate(\mathcal{M}_{\hat{\theta}^t})$
                $d = Distance(D_{sim}, D_{obs})$
                if $d < \epsilon$ then
                    $rejected \leftarrow False$
                    $\theta_i \leftarrow \hat{\theta}^t$
                    $w_i \leftarrow d$
                end if
            end while
        end while
    end while
    Return $(\theta_1, \ldots, \theta_N)$, $(w_1, \ldots, w_N)$
end procedure
```

5.4 Summary

In this chapter we presents two Sequential Monte-Carlo based frameworks, which based on rational functions and simulations. In the following chapter, we benchmark the frameworks using different parametric DTMC to evaluate their performances.

Chapter 6

Case studies

To evaluate the effectiveness of the presented frameworks, we use three case studies. In each case study, we benchmark the frameworks using the following steps:

1. Describe the system of concern.
2. Construct pDTMC models and formalize a reachability property.
3. Select model true parameters and generate synthetic data.
4. Apply the frameworks for both rational functions and simulation-based setups.
5. Visualize the parameter synthesis and inference result.
6. Measure runtime among different model sizes.
7. Discussion on results.

Three case studies include firstly a simple and standard case study *IPv4 ZeroConfiguration Protocol*. The second case study comes from the experiments of the Department of Biology at the University of Konstanz on the defensive behaviour of bee colonies[15]. Third case study is an epidemics model; it is introduced in order to show the expansion of the model state-space as the system has more states to be encoded.
All experiments are conducted in the following system:

- Intel Xeon W-2135 processor, 64GB RAM, OpenSUSE 15.2
- Python 3.8.8, StormPy 1.6.3, Storm stable, PRISM 4.6

6.1 ZeroConfiguration Protocol

6.1.1 System description

Zero-configuration protocol (*zeroconf* for short) is a protocol used in IPv4 network to allocate newly attached device an unique IP address without any intervention from network operators.

Algorithm 11 IPv4 Zeroconf procedure.

```
Input:
  • N: number of probes.
Output:
  • An unused address
procedure Zeroconf
    Select an address ip randomly
    $i = 1$
    while do$i \leq N$
        Broadcast message asking if ip is already in use.
        if Received reply that ip is in use then
            Select an address ip randomly
            Continue loop.
        end if
        if timeout then
            if $i = N$ then
                Return ip
            end if
            $i \leftarrow i + 1$
            Continue loop.
        end if
    end while
end procedure
```

6.1.2 Model and properties

We introduce two real parameters $(p, q) \in [0, 1] \times [0, 1]$.

- p: probability of a message is loss (no reply and timed out).
- q: received a reply that ip is in use.

By replace non-determinisms (timeout and address occupied) by probability distribution, we construct the a pDTMC as a formalism of the Zeroconf protocol of N probes.

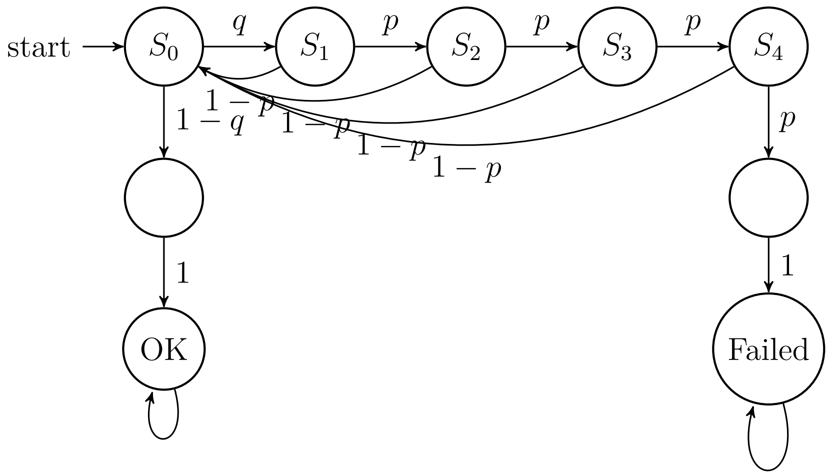


Figure 6.1: IPv4 ZeroConf model, 4 probes

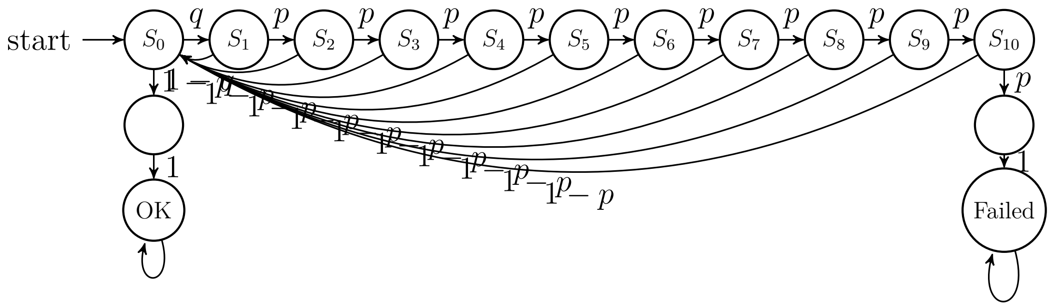


Figure 6.2: IPv4 ZeroConf model, 10 probes

We want to verify the following property:

Eventually, an IP is successfully allocated with at most N probes with probability of at least 75 p

In PCTL formula

$$P_{\geq 0.75}(\texttt{true} \;\; \mathsf{U}^{\leq N} \;\; \texttt{"OK"})$$

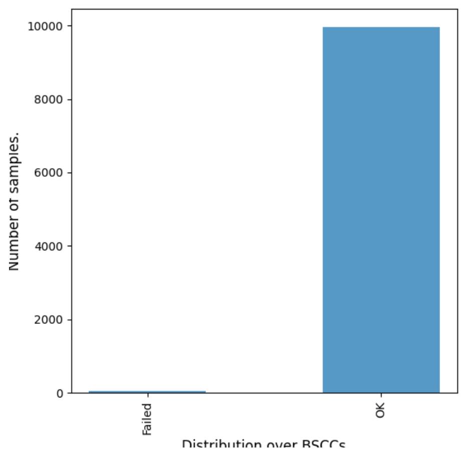
6.1.3 Evaluation

True parameters and synthetic data

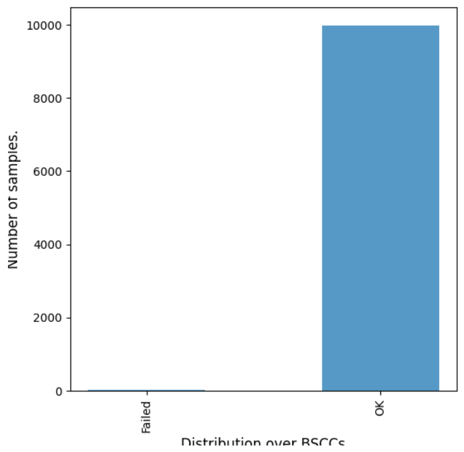
We use the following true parameter:

1. Zeroconf 4 probes: $(p, q) = (0.10554747, 0.44965874)$
2. Zeroconf 10 probes: $(p, q) = (0.19777902, 0.62182433)$

Observation data D_{obs} is from simulating the instantiated DTMC of true parameters 10000 times.



(a) True parameters , 10000 chain simulations.



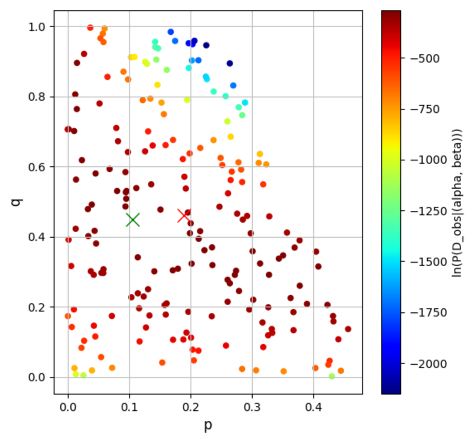
(b) True parameters $(p, q) = (0.19777902, 0.62182433)$, 10000 chain simulations.

Parameter synthesis results

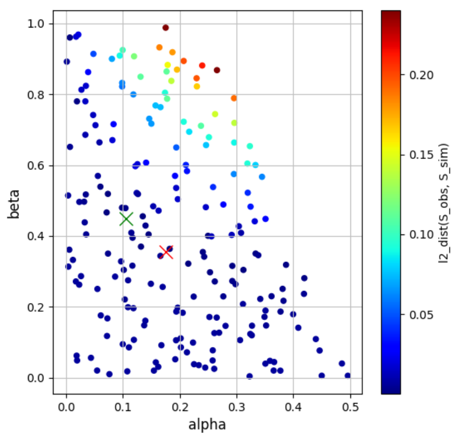
Parameter synthesis for Zeroconf pDTMC of 4 probes.

Zeroconf, 4 probes	Rational function SMC	Statistical model checking ABC-SMC
True parameter	(0.10554747, 0.44965874)	
Number of states	9	
Number of BSCCs	2	
Target property	$P_{\geq 0.75}[!(i > 2)\mathsf{U}^{\leq 4}(\texttt{"OK"})]$	
Synthetic data	$(41, 9959)$	
Inferred parameter point	$(0.18895572, 0.46055455)$	$(0.17646959, 0.35532204)$
L2 distance to true parameter	0.08411690302113349	0.11802271422141228
Run time (hh:mm:ss)	0:06:05.137928	0:54:52.112848

Table 6.1: SIR(5,1,0) parameter estimation results.



(a) Sampled particles using Rational Functions SMC

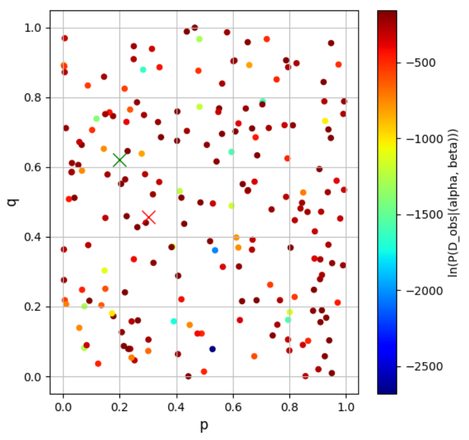


(b) Sampled particles using Statiscal Model Checking ABC-SMC

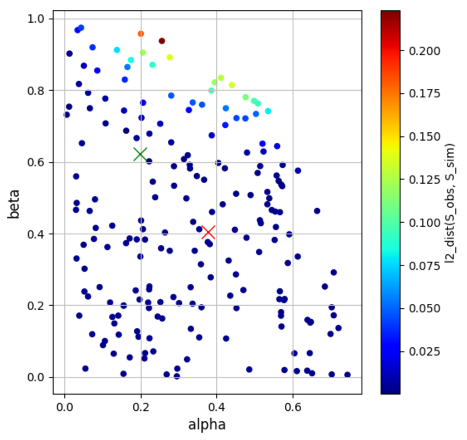
Parameter synthesis for Zeroconf pDTMC of 10 probes.

Zeroconf, 4 probes	Rational function SMC	Statistical model checking ABC-SMC
True parameter	$(0.19777902, 0.62182433)$	
Number of states	14	
Number of BSCCs	2	
Target property	$P_{\geq 0.75}[!(i > 2)\mathsf{U}^{\leq 10}(\texttt{"OK"})]$	
Synthetic data	$(22, 9978)$	
Inferred parameter point	$(0.30180755, 0.45709018)$	$(0.37877446, 0.40586992)$
L2 distance to true parameter	0.19483140118558434	0.2817723545992011
Run time (hh:mm:ss)	0:09:29.760715	0:37:56.507119

Table 6.2: SIR(5,1,0) parameter estimation results.



(a) Sampled particles using Rational Functions SMC



(b) Sampled particles using Statiscal Model Checking ABC-SMC

6.1.4 Discussion

For both model, rational function evaluation is faster than simulation. The simplicity of graphical model gives simple rational function, so evaluating it is much faster than simulation, which requires a lot of system calls to random generators. Both rational function evaluation and simulation scheme are able to deliver a satisfying parameter estimation point. However, in Zeroconf model of 10 probes, simulation has an advantage as the model transitions are simple topologically, while rational function suffers from numerical error.

6.2 Bees colony

6.2.1 System description

We study the collective behavior of a bee colony. Each bee in a colony possibly stings after observing a threat in the surrounding environment, and warn other bees by releasing a special substance, pheromone. By sensing the pheromone released in the environment, other bees in the colony may also sting. Assume that each bee in a colony decides its next action (to sting or not to sting) based only on the current state of the environment, and the number of bees who sting or not sting can be modeled as a Markov process. To reduce the complexity of the model, we make another assumption that the states of the bees colony are observed after uniform time duration, hence the model is of discrete-time. However, since stinging leads to the termination of an individual bee, it reduces the total defense capability as well.

6.2.2 Model and properties

With parametric Discrete-time Markov chain as the model, we study how the actions of a single bee change with regarding to the colony size of and pheromone amount.There are 3 assumptions on the system:

1. Each bee release an unit amount of pheromone immediately after stinging.

2. A bee dies after stinging and releasing pheromone. In the other words, no bee can sting more than once.

3. Stinging behaviour only depends on the concentration of pheromone in the environment.

Under these assumption, a bee colony can be viewed as a set of agents (bees) interact with each other in a closed environment with the appearance of a factor *pheromone*. Afterward, the agent has probability to commit an action, namely *sting*. The agent is eliminated from environment after stinging. Assume that we have a colony of n bees initially. As aforementioned, an individual bee is terminated after it stings. Thus, at the end of experiment, the number of bees is $n' \in \{0, 1, \ldots, n\}$. We model the bee colony with a DTMC $\mathcal{M} = (S, \mathbf{P}, S_{init}, AP, L)$, such that

- $|S_{init}| = 1$
- There exists $n + 1$ tSCCs which encode the population at the end of the experiment.

Semantics of *semi-synchronous* Markov population models for bees colony are developed by [15].

- *Synchronous model*:

In *semisynchronous model*, we assume that the behaviour is initally synchronous. From all succeeding states from initial states, the updates are of asynchronous semantics.

Example 6.3
Semisynchronous model of 3 bees. We model a colony of 3 bees using semisynchronous semantics. In the following DTMC model, each individual bee is encode by an integer represents its state

- *0: bee never stings.*
- *1: bee stings and dies.*
- *2: bee does not sting in 2 consecutive observations.*

Let p, q_1, q_2 represent the probabilities that a bee stings without any stimulation and a bee stings at 1 and 2 attemps, respectively. We then construct the following pDTMC

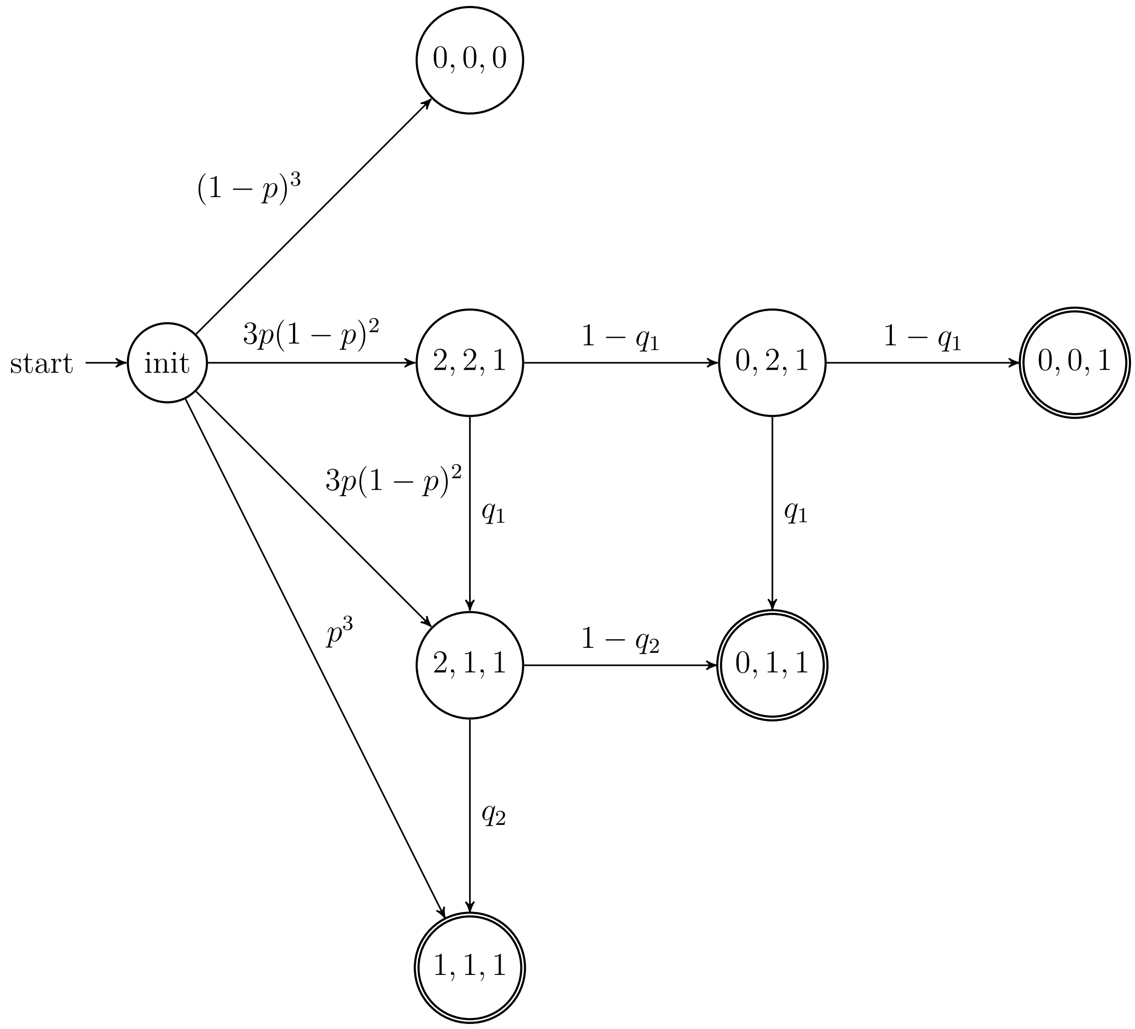


Figure 6.6: Semisynchronous model of 3 bees, 3 parameters p, q_1, q_2)

We verify the following property:

With probability of at least 25 percents, at least a half of bees population survives.

In PCTL formula

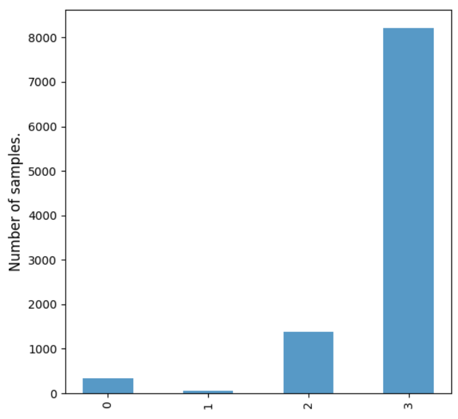
$$P_{\geq 0.25}(\texttt{true} \ \ \mathsf{U} \ \ \texttt{"|Survived| > 0.5N"})$$

	Semisynchronous model, 3 bees	Semisynchronous model, 5 bees
Number of BSCCs	4	6
True parameter	(0.66562362, 0.83040077, 0.83977757)	(0.2783698, 0.30599383, 0.4897924, 0.73725233, 0.76658066)
Synthetic data	(344, 54, 1390, 8212)	(1940, 11, 216, 2682, 4200, 951)

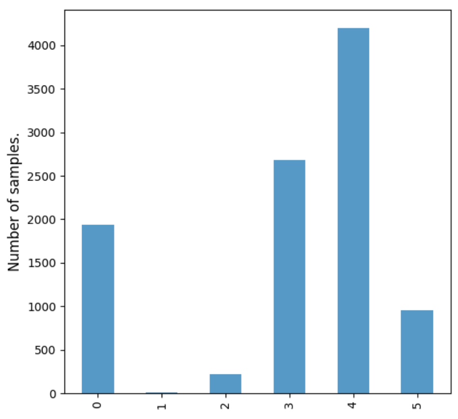
6.3.1 Evaluation

True parameters and synthetic data

We use the following true parameter: Observation data D_{obs} is from simulating the instantiated DTMC of true parameters 10000 times.



(a) True parameters , 10000 chain simulations.



(b) True parameters, 10000 chain simulations.

Parameter synthesis results

Parameter synthesis for semisynchronous model of 3 bees.

Semisynchronous model, 3 bees	SMC with rational function	ABC-SMC with statistical model checking
True parameter	(0.66562362, 0.83040077, 0.83977757)	
Estimated parameter	(0.67138814, 0.57502566, 0.52550228)	(0.81165139, 0.62107331, 0.5441299)
L2 distance	0.40499214733462613	0.3905759806675119
Time elapsed	0:05:55.157159	1:08:47.830380

Parameter synthesis for semisynchronous model of 5 bees.

Semisynchronous model, 5 bees	SMC with rational function	ABC-SMC with statistical model checking
True parameter	(0.2783698, 0.30599383, 0.4897924, 0.73725233, 0.76658066)	
Estimated parameter	(0.57656519, 0.58972387, 0.49033413, 0.55439667, 0.52443301)	(0.36121979, 0.31600669, 0.5456908, 0.64396223, 0.59120587)
L2 distance	0.511366013681474	0.22259463218787823
Time elapsed	0:29:31.224466	5:52:05.296708

6.3.2 Discussion

The case study with bee colony we observed that both simulation-based framework and rational function based framework deliver estimations which are close to the true parameter. However, the simulation-based framework is much more expensive computationally. In the experiment with 5 bees, it is shown that the simulation-based framework obtain better estimation to the true parameter. However, the result is not conclusive since the algorithm is randomized, and more experiments must be conducted to have enough data to conclude the effectiveness of two frameworks.

6.4 SIR model

6.4.1 System

SIR model is a population model, which is widely used in modeling epidemics. In a SIR model, each individual is of one among three types:

- *Susceptible (S)*
- *Infected (S)*
- *Recovered (S)*

SIR system is a stochastic system modeled by reactions between S, I and R. In this thesis we use only 2 reactions.

$$R_1 : S + I \xrightarrow{\alpha} 2I$$
$$R_2 : I \xrightarrow{\beta} R$$

Algorithm 12 Generate SIR CTMC from reactions.

Input:

- (S_0, I_0, R_0): initial population.
- Reactions of rate α, β

$$R_1 : S + I \xrightarrow{\alpha} 2I$$
$$R_2 : I \xrightarrow{\beta} R$$

Output:

- CTMC $\mathcal{C}$

```
1:  procedure Explore($s$,$i$,$r$)
2:      if $i = 0$ then
3:          Mark $(s, i, r)$ as a BSCC.
4:          Return
5:      end if
6:      if $(s > 0) \wedge (i > 0)$ then
7:          if State $(s - 1, i + 1, r)$ already visited then
8:              Return
9:          end if
10:         Add $(s - 1, i + 1, r)$ to state space
11:         Explore($s - 1$, $i + 1$,$r$)
12:     end if
13:     if $(i > 0)$ then
14:         if State $(s, i - 1, r + 1)$ already visited then
15:             Return
16:         end if
17:         Add $(s, i - 1, r + 1)$ to state space
18:         Explore($s$, $i - 1$,$r + 1$)
19:     end if
20: end procedure
21: procedure Sir-Explore-Statespace($s_0, i_0, r_0$)
22:     Sir-Explore-Statespace($s_0, i_0, r_0$)
23: end procedure
```

6.4.2 Model and properties

Theorem 5
Acyclicity A CTMC $\mathcal{C}$ constructed by Algorithm 12 using reactions R_1, R_2 is acyclic.

Proof: For any arbitrary transition in $\mathcal{C}$

1. $|S|$ is monotonically decreasing, as there exists no reaction which produces S.

2. $|R|$ is monotonically increasing, as there exists no reaction which consumes R.

3. If $P((s,i,r),(s',i',r')) \neq 0$, then $i \neq i'$. That is because all reactions change i.

As $|S| + |I| + |R| = S_0 + I_0 + R_0$ and S_0, R_0, I_0 are constants, if there exists a path fragment

$$(s^t, i^t, r^t) \rightarrow \quad \dots \quad \rightarrow (s^{t+k}, i^{t+k}, r^{t+k})$$

such that $(s^t, i^t, r^t) = (s^{t+k}, i^{t+k}, r^{t+k})$ then $k = 0$, because all reactions change i (if $P((s,i,r),(s',i',r')) \neq 0$, then $i \neq i'$). $\square$

Corollary 5.1
A CTMC constructed by Algorithm 12 using reactions R_1, R_2 has BSCCs and the BSCCs are trivial.

Example 6.5
Example of an SIR CTMC model with initial population $(S_0, I_0, R_0) = (3, 1, 0)$

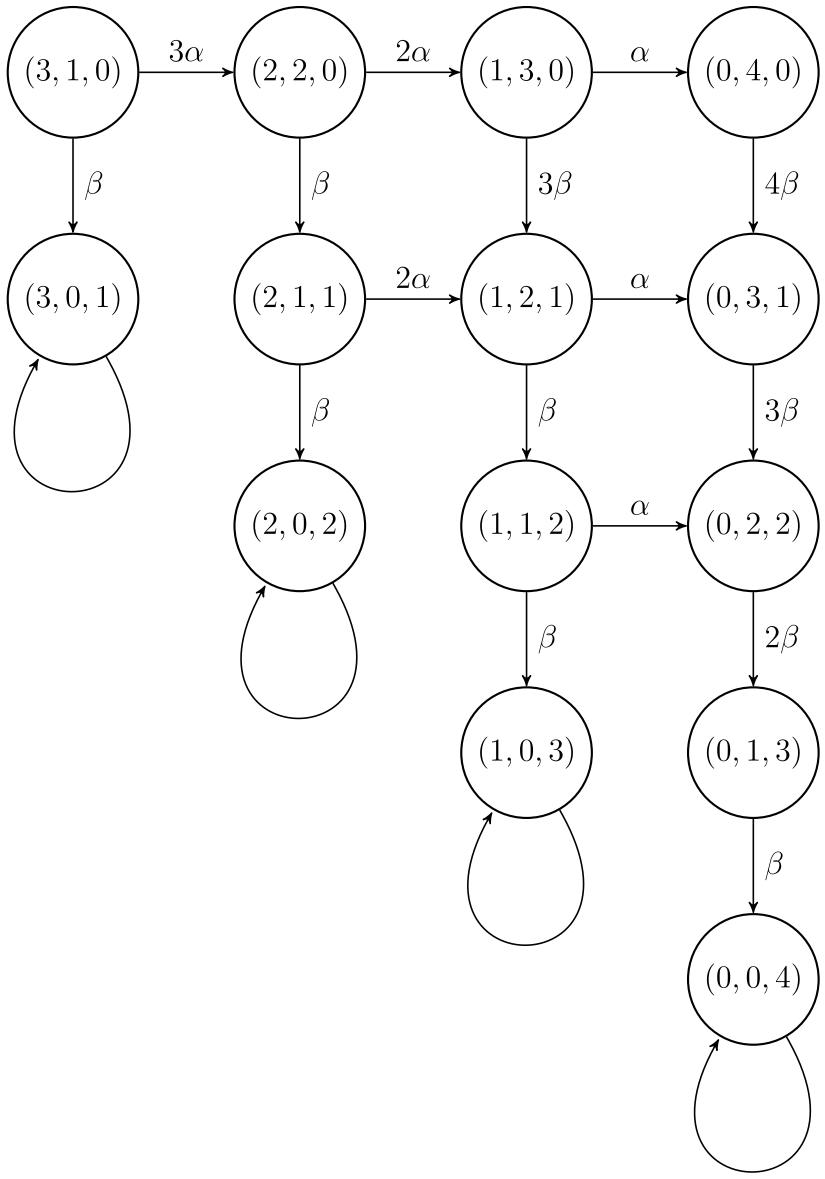
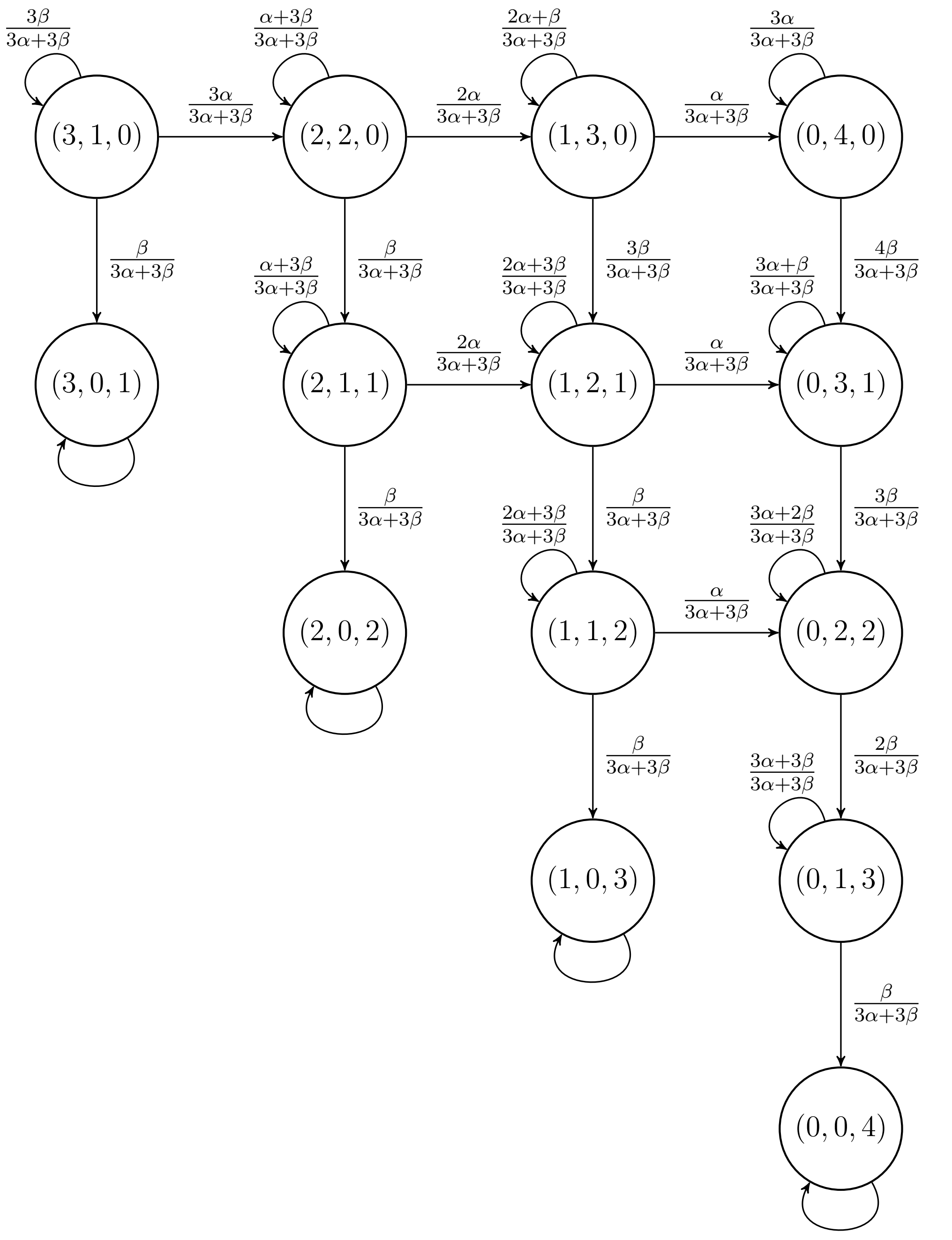


Figure 6.8: $SIR(3, 1, 0)$ CTMC model with parameters(α, β)

Example 6.6
Uniformize the chain with uniformization rate $(3\alpha + 4\beta)$*, we derive the following uniformized DTMC:*



$\frac{3\beta}{3\alpha+3\beta}$
$\frac{\alpha+3\beta}{3\alpha+3\beta}$
$\frac{2\alpha+\beta}{3\alpha+3\beta}$
$\frac{3\alpha}{3\alpha+3\beta}$
$(3,1,0)$
$\frac{3\alpha}{3\alpha+3\beta}$
$(2,2,0)$
$\frac{2\alpha}{3\alpha+3\beta}$
$(1,3,0)$
$\frac{\alpha}{3\alpha+3\beta}$
$(0,4,0)$
$\frac{\beta}{3\alpha+3\beta}$
$\frac{\alpha+3\beta}{3\alpha+3\beta}$
$\frac{\beta}{3\alpha+3\beta}$
$\frac{2\alpha+3\beta}{3\alpha+3\beta}$
$\frac{3\beta}{3\alpha+3\beta}$
$\frac{3\alpha+\beta}{3\alpha+3\beta}$
$\frac{4\beta}{3\alpha+3\beta}$
$(3,0,1)$
$(2,1,1)$
$\frac{2\alpha}{3\alpha+3\beta}$
$(1,2,1)$
$\frac{\alpha}{3\alpha+3\beta}$
$(0,3,1)$
$\frac{\beta}{3\alpha+3\beta}$
$\frac{2\alpha+3\beta}{3\alpha+3\beta}$
$\frac{\beta}{3\alpha+3\beta}$
$\frac{3\alpha+2\beta}{3\alpha+3\beta}$
$\frac{3\beta}{3\alpha+3\beta}$
$(2,0,2)$
$(1,1,2)$
$\frac{\alpha}{3\alpha+3\beta}$
$(0,2,2)$
$\frac{\beta}{3\alpha+3\beta}$
$\frac{3\alpha+3\beta}{3\alpha+3\beta}$
$\frac{2\beta}{3\alpha+3\beta}$
$(1,0,3)$
$(0,1,3)$
$\frac{\beta}{3\alpha+3\beta}$
$(0,0,4)$

Figure 6.9: $SIR(3,1,0)$ Uniformized DTMC model with parameters(α,β) and uniformization rate $(3\alpha+4\beta)$

We check the following property: "With probability of at least 25 percents, the number of infected individuals does not exceed half of the population until the system is stabilized." In PCTL formula:

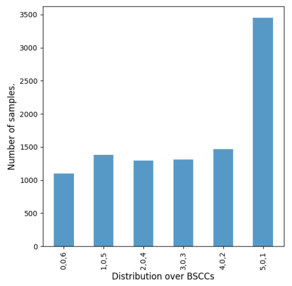
$$P_{\geq 0.25}(!(i > (S_0 + i_0 + R_0)/2) \quad \mathsf{U} \quad (i = 0))$$

6.6.1 Evaluation

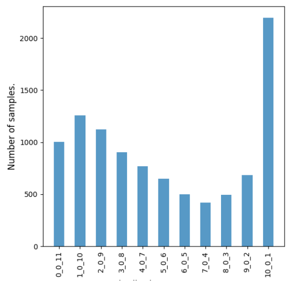
True parameters and synthetic data

Parameter synthesis result

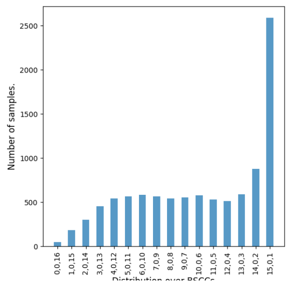
We evaluate simulation-based and rational function based frameworks on different size of initial population.



(a) SIR(5,1,0) synthetic data



(b) SIR(5,1,0) synthetic data



(c) SIR(5,1,0) synthetic data

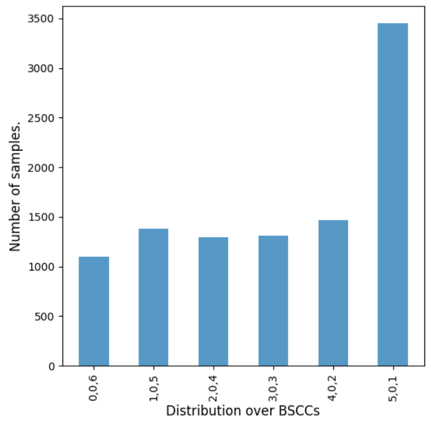
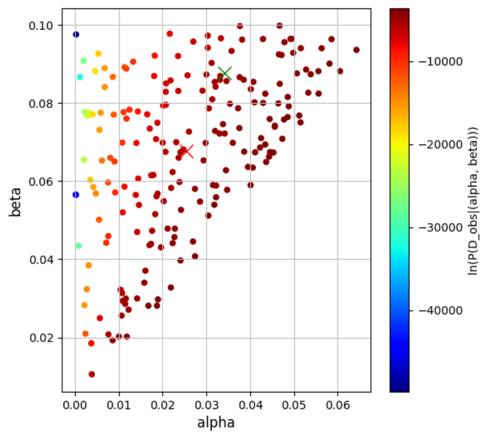


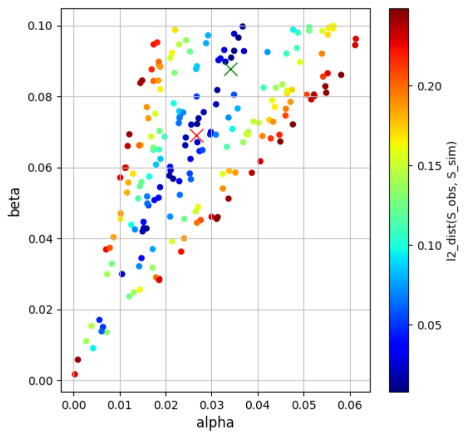
Figure 6.10: Synthetic data D_{obs} using selected true parameter.

SIR(5,1,0)	Rational function SMC	Statistical model checking ABC-SMC
True parameter	(0.03405521, 0.08773454)	
Number of states	27	
Number of BSCCs	6	
Target property	$P_{\geq 0.25}[!(i > 3)U^{<6}(i = 0)]$	
Synthetic data	(1098, 1377, 1296, 1312, 1466, 3451)	
Inferred parameter point	(0.02547391, 0.0676129)	(0.02307652, 0.06481155)
L2 distance to true parameter	0.021875082353604854	0.020119851055552338
Run time (hh:mm:ss)	0:19:34.397596	3:51:36.893424

Table 6.3: SIR(5,1,0) parameter estimation results.



(a) Sampled particles using Rational Functions SMC



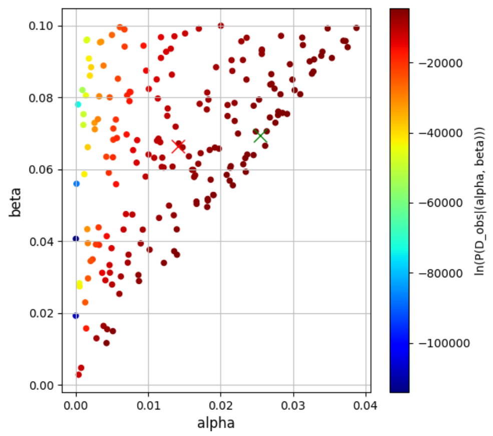
(b) Sampled particles using Statiscal Model Checking ABC-SMC

Figure 6.11: SIR(5,1,0) parameter synthesis results.

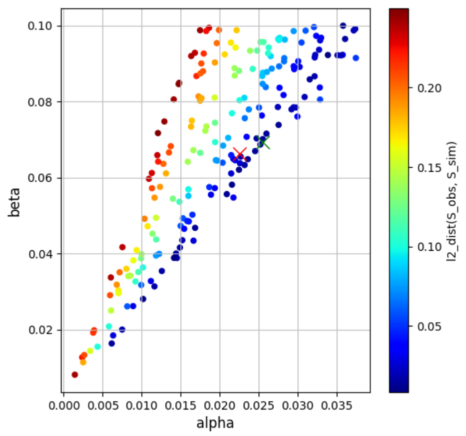
In the experiment with $SIR(5, 1, 0)$ model we can see that rational-function based method and simulation based method deliver results without significantly difference.

SIR(10,1,0)	Rational function SMC	Statistical model checking ABC-SMC
True parameter	(0.02549012, 0.0692981)	
Number of states	77	
Number of BSCCs	11	
Target property	$P_{\geq 0.25}[!(i > 5)U^{<11}(i = 0)]$	
Synthetic data	(1002, 1258, 1123, 902, 770, 651, 497, 420, 496, 685, 2196)	
Inferred parameter point	(0.0140951 0.06632789)	(0.02255187, 0.06641589)
L2 distance to true parameter	0.011775762120972396	0.004115880723356161
Run time (hh:mm:ss)	2:45:02.154117	9:27:41.345284

Table 6.4: SIR(10,1,0) parameter estimation results.



(a) Sampled particles using Rational Functions SMC



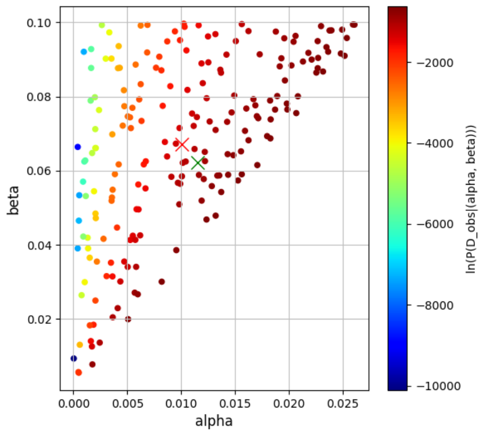
(b) Sampled particles using Statiscal Model Checking ABC-SMC

Figure 6.12: SIR(10,1,0) parameter synthesis results.

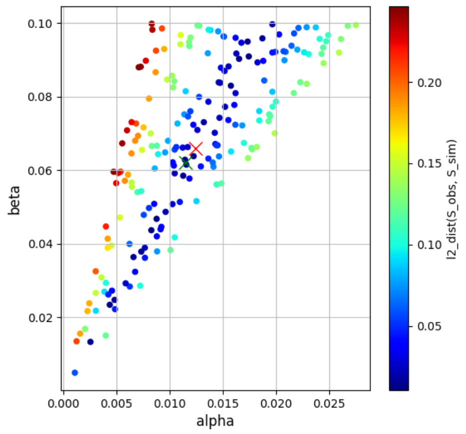
In the experiment with $SIR(10, 1, 0)$ model we can see that simulation based method deliver closer estimation to true parameter.

SIR(15,1,0)	Rational function SMC	Statistical model checking ABC-SMC
True parameter	(0.01724649, 0.06778604)	
Number of states	27	
Number of BSCCs	6	
Target property	$P_{\geq 0.25}[!(i > 7)U^{<16}(i = 0)]$	
Synthetic data	(50,181,302,455,539,567,582,566,541,553,574,528,512,586,875,2589	
Inferred parameter point	(0.02307652, 0.06481155)	(0.01758384, 0.06535699)
L2 distance to true parameter	0.006544985909916083	0.005519695496673707
Run time (hh:mm:ss)	1:07:36.442146	3:05:22.61795

Table 6.5: SIR(15,1,0) parameter estimation results.



(a) Sampled particles using Rational Functions SMC



(b) Sampled particles using Statiscal Model Checking ABC-SMC

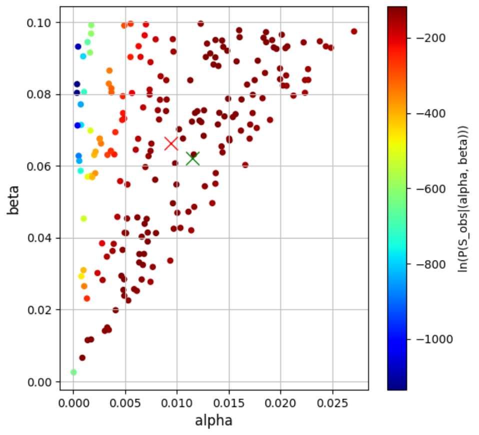
Figure 6.13: SIR(15,1,0) parameter synthesis results.

Parameter synthesis with uncertainty

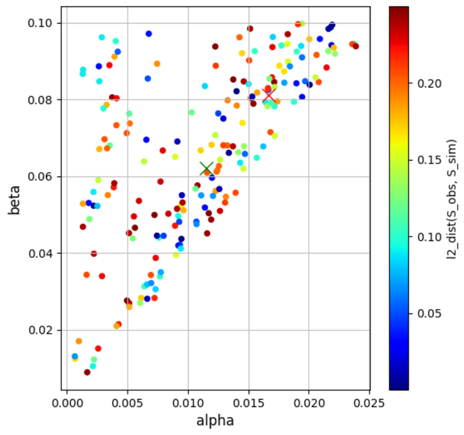
We introduce an uncertainty by merging BSCCs into 2 bins and use lower number of simulation to generate synthetic data.

SIR(15,1,0), BSCC merged	Rational function SMC	Statistical model checking ABC-SMC
True parameter	(0.01724649, 0.06778604)	
Number of states	27	
Number of BSCCs	6	
Target property	$P_{\geq 0.25}[!(i > 7)U^{<16}(i = 0)]$	
Synthetic data	(421, 834, 1126, 1362, 1851, 4406)	
Inferred parameter point	(0.02307652, 0.06481155)	(0.01758384, 0.06535699)
L2 distance to true parameter	0.0065449859099916083	0.0055196954966737 07
Run time (hh:mm:ss)	1:07:36.442146	3:05:22.61795

Table 6.6: SIR(5,1,0) parameter estimation results.



(a) Sampled particles using Rational Functions SMC



(b) Sampled particles using Statiscal Model Checking ABC-SMC

6.6.2 Discussion

Unlike experiments with unmerged BSCCs, the experiment with merged BSCCs and simulation based framework shows no general pattern of distance among satisfying parameter values.

Chapter 7

Conclusion

We presented frameworks to perform data-informed parameter synthesis. The frameworks are tested against different case studies and show that they are able to deliver both satisfying parameter value set and an estimated parameter value that is close to the original value which is used to synthesize test data. Therefore, these frameworks are applicable when we need not only an estimation on the unknown attributes of a system, but also proactively verify the system against a desired property.
There are many possible extensions to the presented frameworks, including but not limited to:

- *Statistical Model Checking*: we can use Absolute-Error Massart Bounds (proposed by Molyneux [29], but currently not supported by PRISM) on Statistical Model Checking to achieve a better bound on the number of simulation.

- *Bayesian Model Checking*: Zuliani [38] presents a novel method that improves Statistical Model Checking by using Bayesian approach.

- *Sampling algorithms*: different sampling algorithms can be used to estimate posterior distribution. For example, PyMC3 library [32] uses No-U-Turn Sampling algorithm by default to estimate posterior distribution.

- *Implementation improvement*: currently StormPy prohibits our implementation to be parallelized, since StormPy's core classes are not serializable. Porting to C++ language possibly has several benefits by achieving higher performance of C++ and exploiting the data-parallelism of Sequential Monte-Carlo algorithm.

Bibliography

[1] Greg M Allenby, Peter E Rossi, and RE McCulloch. “Hierarchical Bayes Models: A Practitioners Guide. Grover R, Vriens M, eds”. In: *SSRN Electron J* (2005).

[2] David F Anderson and Thomas G Kurtz. “Continuous time Markov chain models for chemical reaction networks”. In: *Design and analysis of biomolecular circuits*. Springer, 2011, pp. 3–42.

[3] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT press, 2008.

[4] Michael Baron. *Probability and statistics for computer scientists*. CRC Press, 2019.

[5] Herman Chernoff. “A career in statistics”. In: *Past, Present, and Future of Statistical Science* 29 (2014).

[6] Edmund M Clarke et al. “Model checking and the state explosion problem”. In: *LASER Summer School on Software Engineering*. Springer. 2011, pp. 1–30.

[7] Edmund M. Clarke, E Allen Emerson, and A Prasad Sistla. “Automatic verification of finite-state concurrent systems using temporal logic specifications”. In: *ACM Transactions on Programming Languages and Systems (TOPLAS)* 8.2 (1986), pp. 244–263.

[8] Remi Daviet. “Inference with Hamiltonian Sequential Monte Carlo Simulators”. In: *arXiv preprint arXiv:1812.07978* (2018).

[9] Conrado Daws. “Symbolic and parametric model checking of discrete-time Markov chains”. In: *International Colloquium on Theoretical Aspects of Computing*. Springer. 2004, pp. 280–294.

[10] Christian Dehnert et al. "A storm is coming: A modern probabilistic model checker". In: *International Conference on Computer Aided Verification*. Springer. 2017, pp. 592–600.

[11] Pierre Del Moral, Arnaud Doucet, and Ajay Jasra. "Sequential monte carlo samplers". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 68.3 (2006), pp. 411–436.

[12] Martin Feinberg. "Chemical oscillations, multiple equilibria, and reaction network structure". In: *Dynamics and modelling of reactive systems*. Elsevier, 1980, pp. 59–130.

[13] Sofie Haesaert, Alessandro Abate, and Paul MJ Van den Hof. "Data-driven and model-based verification: A bayesian identification approach". In: *2015 54th IEEE Conference on Decision and Control (CDC)*. IEEE. 2015, pp. 6830–6835.

[14] Ernst Moritz Hahn, Holger Hermanns, and Lijun Zhang. "Probabilistic reachability for parametric Markov models". In: *International Journal on Software Tools for Technology Transfer* 13.1 (2011), pp. 3–19.

[15] Matej Hajnal et al. "Data-Informed Parameter Synthesis for Population Markov Chains". In: *International Workshop on Hybrid Systems Biology*. Springer. 2019, pp. 147–164.

[16] W Keith Hastings. "Monte Carlo sampling methods using Markov chains and their applications". In: (1970).

[17] Christian Hensel et al. "The probabilistic model checker Storm". In: *arXiv preprint arXiv:2002.07080* (2020).

[18] W Hoeffding. "Probability inequalities for sums of bounded random variables. American Statistical Association Journal, 13–30. Katz, S.(1987). Estimation of probabilities from sparse data for the language model component of a speech recognizer". In: *IEEE Transactions on Acoustic, Speech and Signal Processing* 35 (1963), pp. 400–401.

[19] Lisa Hutschenreiter, Christel Baier, and Joachim Klein. "Parametric Markov chains: PCTL complexity and fraction-free Gaussian elimination". In: *arXiv preprint arXiv:1709.02093* (2017).

[20] Sumit K Jha et al. "A bayesian approach to model checking biological systems". In: *International conference on computational methods in systems biology*. Springer. 2009, pp. 218–234.

[21] Sebastian Junges et al. "Parameter synthesis for Markov models". In: *arXiv preprint arXiv:1903.07993* (2019).

[22] Sebastian Junges et al. *Parameter synthesis in Markov models.* Tech. rep. Fachgruppe Informatik, 2020.

[23] Joost-Pieter Katoen. "Model Checking Meets Probability: A Gentle Introduction." In: *Engineering dependable software systems* 34 (2013), pp. 177–205.

[24] Joost-Pieter Katoen. "The probabilistic model checking landscape". In: *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*. 2016, pp. 31–45.

[25] John FC Kingman. "Markov population processes". In: *Journal of Applied Probability* (1969), pp. 1–18.

[26] Marta Kwiatkowska, Gethin Norman, and David Parker. "PRISM 4.0: Verification of probabilistic real-time systems". In: *International conference on computer aided verification*. Springer. 2011, pp. 585–591.

[27] Marta Kwiatkowska, Gethin Norman, and David Parker. "Symmetry reduction for probabilistic model checking". In: *International Conference on Computer Aided Verification*. Springer. 2006, pp. 234–248.

[28] Nicholas Metropolis et al. "Equation of state calculations by fast computing machines". In: *The journal of chemical physics* 21.6 (1953), pp. 1087–1092.

[29] Gareth W Molyneux and Alessandro Abate. "ABC(SMC)^2: Simultaneous Inference and Model Checking of Chemical Reaction Networks". In: *International Conference on Computational Methods in Systems Biology*. Springer. 2020, pp. 255–279.

[30] Gareth W Molyneux, Viraj B Wijesuriya, and Alessandro Abate. "Bayesian verification of chemical reaction networks". In: *International Symposium on Formal Methods*. Springer. 2019, pp. 461–479.

[31] Elizabeth Polgreen et al. "Data-efficient Bayesian verification of parametric Markov chains". In: *International Conference on Quantitative Evaluation of Systems*. Springer. 2016, pp. 35–51.

[32] John Salvatier, Thomas V Wieckiâ, and Christopher Fonnesbeck. "PyMC3: Python probabilistic programming framework". In: *ascl* (2016), ascl–1610.

[33] Daniel Silk, Saran Filippi, and Michael PH Stumpf. “Optimizing threshold-schedules for approximate Bayesian computation sequential Monte Carlo samplers: applications to molecular systems”. In: *arXiv preprint arXiv:1210.3296* (2012).

[34] Tina Toni et al. “Approximate Bayesian computation scheme for parameter inference and model selection in dynamical systems”. In: *Journal of the Royal Society Interface* 6.31 (2009), pp. 187–202.

[35] Stephen Tu. “The dirichlet-multinomial and dirichlet-categorical models for bayesian inference”. In: *Computer Science Division, UC Berkeley* (2014).

[36] Abraham Wald. “Sequential tests of statistical hypotheses”. In: *The annals of mathematical statistics* 16.2 (1945), pp. 117–186.

[37] Håkan LS Younes and Reid G Simmons. “Probabilistic verification of discrete event systems using acceptance sampling”. In: *International Conference on Computer Aided Verification*. Springer. 2002, pp. 223–235.

[38] Paolo Zuliani, André Platzer, and Edmund M Clarke. “Bayesian statistical model checking with application to Stateflow/Simulink verification”. In: *Formal Methods in System Design* 43.2 (2013), pp. 338–367.