Bayesian Parameter Synthesis of Markov Population Models

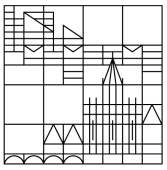
Master Thesis

Submitted by

Nhat-Huy Phung

at the

Universität Konstanz



Modeling of Complex, Self-organising Systems

Department of Computer and Information Science

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

Konstanz, 2021

Contents

Acknowledgements

First and foremost, I would like to express its gratitude and appreciation to my supervisor Prof.Dr. Tatjana Petrov. Without her motivation, advice, supports, and feedback on my implementation and writing, this thesis would be far from being done.

Working in the research group MCSS (modeling of complex, self-organising systems) is an excellent experience. I would like to thank the members of MCSS group, Matej Hajnal, Stefano Tognazzi, and Denis Repin for their nice discussions and feedback on the technical and theoretical aspects of my work.

My deep gratitude goes to my family. Despite of being thousands of kilometers away, they are always the best source of support and encouragement. A special thank is for Lu Dinh, for her endless love and encouragement.

Abstract

Population models are mathematical models to study the dynamics of a population. Markov population processes are Markov chains of continuous or discrete time, in which a state tracks population size of each of the involved species. We study a framework for data-informed parameter synthesis of parametric Markov population processes. Given experimental data for the population at its steady-state, parameter synthesis aims to find a set of parameters satisfying a temporal property of interest. We assume that experimental observations of the population are available at steady-state. We design and compare performance of a Bayesian framework for parameter synthesis in two cases: (i) when the exact likelihood function of the property of interest is computed in a pre-processing step, and (ii) when it has to be approximated by means of Monte-Carlo methods (it is likelihood-free). The frameworks are constructed with different sampling and optimization techniques to approximate the posterior distribution. We evaluate the frameworks using various population models of different sizes, using synthetic data generated from a known true parameter. By measuring the distance between estimated parameters and true parameters, and by visualize synthesized parameter values with their corresponding posterior estimate, we show that both frameworks can derive a set of satisfying parameter values and an estimation that is close to the true parameter. Furthermore, results show that the likelihood-free framework is overall performing better.

Chapter 1

Introduction

1.1 Motivation

Firstly introduced by Kingsman [34], Markov population processes are finite state-space, stochastic models widely used in modeling complex and dynamical systems. In a Markov population model, each state represents the number of individuals, and the transitions among states represent the increase or decrease of a population. In general, Markov population models study the population dynamics of a system of interest. For example, Markov population processes are able to model:

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

Studying the Markov population model has challenges. First, model state space exponentially expands as we capture more attributes and behavior of the system of interest. The explosion of state-space makes model checking of the Markov population model computationally intensive. Second, in a Markov population model, such as Discrete-time Markov Chain, initial and transition probabilities are known a priori. 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 parameters represent unknown features of the system, it gives the following research questions

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

Parameter synthesis is an emerging research direction on probabilistic model checking. Katoen [31] defines the parameter synthesis problem for the parametric discrete-time Markov chain to find a set of parameter values that satisfy a given reachability property. In this thesis, we combine Bayesian parameter inference and parameter synthesis. The result parameters (i) satisfy the property of interest, and (ii) are likely to produce given steady-state data. Contributions of the thesis are

- We are presenting and implementing a data-informed, Bayesian framework on parameter synthesis of parametric Discrete-time Markov chain. The frameworks work in two cases: (i) when the exact likelihood function of the property of interest is available, and (ii) when it has to be approximated utilizing Monte-Carlo methods.
- We compare the performances of proposed frameworks used to approximate posterior distribution on various case studies.
- We evaluate the scalability of the frameworks for different sizes of model state-space.

1.2 Related works

The frameworks presented in this thesis are based on ABC-SMC framework [40] and ABC-(SMC)2 [39] by Molyneux et al. However, the ABC-SMC and ABC-(SMC)2 frameworks synthesize parameters for CTMC and check the CTMC model against CSL properties. In parametric DTMC, since the symbolic rational function of PCTL property is obtainable [9], we based on Del Moral [12] and Daviet [8] to construct an algorithm based on evaluation of symbolic rational function, then benchmark it against a simulation-based approach.

The theoretical background of model checking discrete-time Markov chain is presented by Baier et al. [2]. Katoen [31] presents a tutorial to model check parametric discrete-time Markov chain and current methods on parameter synthesis. More in-depth surveys and discoveries on parametric model checking and parameter synthesis are presented by Junges [29] and Hutschenreiter [26].

Markov Chain Monte Carlo sampling algorithms used in this thesis are presented by Metropolis [38], and Hastings [21]. Del Moral [12] designed Sequential Monte Carlo to address the limitations of Markov Chain Monte Carlo. A comparison between different Monte Carlo sampling algorithms, including Markov-chain Monte Carlo and Sequential Monte Carlo is presented in [8]. Silk [45] and Filippi [15] discussed different approaches on the perturbation 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 is implemented using Storm model checker [22]. 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, the Statistical Model Checking step in simulation-based frameworks is implemented using PRISM [36].

1.3 Structure of the thesis

The content in this thesis is organized to 7 chapters:

- **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** proposes Bayesian parameter synthesis frameworks.
- **Chapter 5** describes case studies and benchmarks presented frameworks under different setups.
- **Chapter 6** conclusion and outlook.

Chapter 2

Probabilistic model checking

Discrete-time Markov chain is a formalism often used 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 Probabilistic Computational Tree Logic (or PCTL in short). Due to the state space explosion, applying a deterministic model checking algorithm is often computationally expensive. Therefore, we also present a simulation-based model checking, namely statistical model checking (or SMC in short) for bounded path property. We also introduce definitions of parametric DTMC and the parameter synthesis problem, and the symbolic computing approach to verify parametric models.

2.1 Markov models

2.1.1 Discrete Time Markov chain

Markov models are stochastic models evolving in discrete or continuous time, which satisfy the *memoryless property*. Memoryless property is a property of stochastic processes, which refers to the independence of the future states of the process on its previously visited states. Discrete-time Markov chain is a Markov model of discrete state-space and countable (discrete) time. In a discrete-time Markov chain, times between any state transition are uniform.

Definition 2.1.1 (Discrete Time Markov Chain [2])
A *Discrete-time Markov chain* (or DTMC in short) $\mathcal{M}$ 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 1 (Knuth-Yao die)
Knuth and Yao [35] introduced an algorithm to simulate a fair dice by a fair coin. The algorithm terminates returning one among six possible outcomes of a die tossing (from one to six). We formalize the algorithm by a DTMC with 6 BSCCs. Each of the 6 BSCCs represents a possible outcome of a die tossing. The probabilities of reaching each BSCC among the six is 1/6.

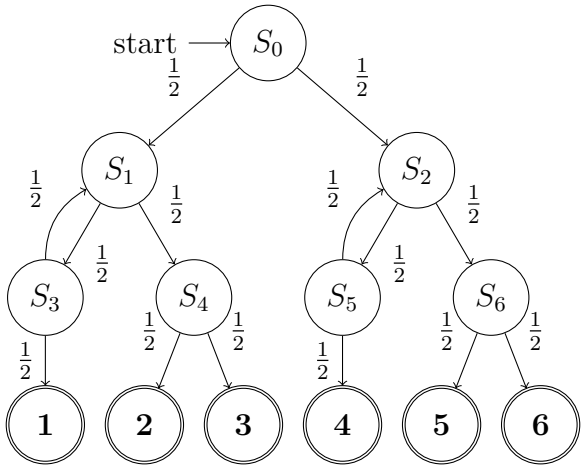


Figure 2.1: DTMC model of Knuth-Yao algorithm. The accepting states 1 to 6 represent a face of a fair die.

Definition 2.1.2 (Strongly Connected Component)
Let $\mathcal{M} = (S, \mathbf{P}, \iota_{init}, AP, L)$ be 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 states in S'. If S' has no superset $S'' \subseteq S$, such that S'' is strongly connected, then S' is a *Strongly Connected Component*, or *SCC* in short.

Definition 2.1.3 (Bottom Strongly Connected Component)
Let $\mathcal{M} = (S, \mathbf{P}, \iota_{init}, AP, L)$ be a DTMC and $S' \in S$ a Strongly Connected Component. S' is also a *Bottom Strongly Connected Component* (or *BSCC* in short), if and only if there exist no state $s \in S \backslash S'$ 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 [2] that any run on a DTMC $\mathcal{M}$ ends in $BSCC(\mathcal{M})$ almost surely.

Theorem 2 (Long-run theorem)
Let $\mathcal{M} = (S, \mathbf{P}, \iota_{init}, AP, L)$ be a DTMC. Let $BSCC(\mathcal{M})$ be the set of all BSCCs of $\mathcal{M}$. Then, the probability of reaching a BSCC and visit all of its states infinitely often is equal to 1. Formally

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

In this thesis, we use the experiment data as the *steady-state distribution* of a DTMC.

Theorem 3 (Steady-state distribution)
Let $\mathcal{M} = (S, \mathbf{P}, \iota_{init}, AP, L)$ be a DTMC and a vector $\nu_t \in [0,1]^{|S|}$ be a transient state distribution at time $t \in \mathbb{N}$ defined by

$$\nu_t = (Pr(X_t = s_1), \ldots, Pr(X_t = s_{|S|})) \quad s_1, \ldots, s_{|S|} \in S$$

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

$$t \to \infty : \nu_t = \nu_t P$$

As a result from long-run theorem, we have the following lemma regarding the existence of the steady state distribution in DTMC with BSCC(s)

Lemma 4
Let $\mathcal{M} = (S, \mathbf{P}, \iota_{init}, AP, L)$ be a DTMC and $BSCC(\mathcal{M})$ be the set of all BSCCs of

$\mathcal{M}$. If $BSCC(\mathcal{M}) \neq \emptyset$ then there exists a steady-state distribution $\nu = (Pr(X = s_1), \ldots, Pr(X = s_{|S|}))$, such that

$$\forall i : 1 \leq i \leq |S| : P(X = s_i) \neq 0$$

if and only if

$$s_i \in BSCC(\mathcal{M})$$

2.1.2 Continuous-time Markov chain

The discrete-time memoryless property can also be extended into continuous-time memoryless property. Continuous-time Markov chain is often used to model a system in which times between transitions vary and are real numbers.

Definition 2.1.4 (Continuous-time Markov chain [32])
A Continuous-time Markov chain (or CTMC in short) $\mathcal{C}$ is a tuple $(S, \mathbf{P}, \mathbf{R}, \iota_{init}, AP, L)$, where

- *S is a countable, non-emty set of states*
- *$\mathbf{R} : S \times S \to \mathbb{R}_{>0}$ is the rate matrix.*
- *$\iota_{init} : S \to [0, 1]$ 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 5 (CTMC)
We give an example of a CTMC $\mathcal{C} = (S, \mathbf{R}, \iota_{init}, AP, L)$, where

- *State set $S = \{S_0, S_1, S_2\}$*
- *Initial state $\iota_{init} = \{S_0\}$*
- *Transition rate matrix*

$$\mathbf{R} = \begin{bmatrix} 0 & 2 & 0 \\ \frac{4}{3} & 0 & \frac{8}{3} \\ 0 & 3 & 0 \end{bmatrix}$$

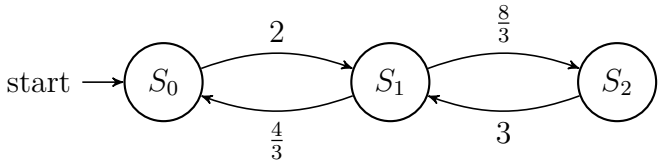


Figure 2.2: An example of a CTMC with 3 states.

Definition 2.1.5 (Exit rate [32])
Let $\mathcal{C} = (S, \mathbf{R}, \iota_{init}, AP, L)$ be a CTMC. We define the *exit rate* $\mathbf{E}(s), s \in S$ as the row sum of the rate matrix $\mathbf{R}$

$$\mathbf{E}(s) = \sum_{s' \in S} \mathbf{R}(s, s')$$

CTMC is a solid formalism to model systems concerning real-time properties. In this thesis, we transform CTMCs into DTMCs through a process of *uniformization* [32]. Uniformization discretizes a CTMC by transforming it in to a CTMC which has uniform exit rates among states, so that we can approximate the uniformized CTMC using a DTMC by removing the exit rates, and modelling times between transitions by a Poisson process with rate equal to the chosen uniformization constant. Katoen [32] shows that the uniformization of CTMC preserves the transient probabilities. Baier [3] proved that the time-bounded until formulas remain unaffected as we transform a CTMC to its equivalent uniformized form. Through uniformization, we can check the time-bounded reachability property. Furthermore, transforming a CTMC to an equivalent DTMC allows us to obtain a symbolic form of the PCTL formula, thus enabling a more efficient evaluation of properties.

Definition 2.1.6 (CTMC uniformization [32])
Let $\mathcal{C} = (S, \mathbf{R}, \iota_{init}, AP, L)$ be a CTMC with exit rate $E(s), s \in S$. We define the *uniformization rate* r such that

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

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

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

Example 6 (Uniformized CTMC)
After uniformizing the CTMC in Figure **??** *by uniformization rate* $r = 4$*, we obtain the followig DTMC*

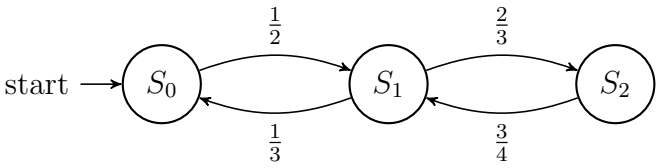


Figure 2.3: Uniformized DTMC

2.2 Property specification

2.2.1 Probabilistic Computational Tree Logic

Model checking verifies 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 Hansson et al. [20], PCTL is widely used in model checking of discrete-time stochastic models and supported by probabilistic model checking tools Storm [11], and PRISM [36].

Definition 2.2.1 (PCTL [2])
The syntax of PCTL consists of state formulas and path formulas.

- State formulas are defined over AP

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

  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 $n \in \mathbb{N}$.

Example 7
Given the DTMC as in Figure 2.1. The probability that the simulation eventually ends with the outcome "one dot" is equal to $\frac{1}{6}$*:*

$$P_{=\frac{1}{6}}(\Diamond 1)$$

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 [2]. Given a DTMC $\mathcal{M}$ and a PCTL property Φ, general algorithm for checking $\mathcal{M} \models \Phi$ has time complexity polynomial to $|\mathcal{M}|$ and linear to $|\Phi|$.

Theorem 8 (Complexity of checking a DTMC against a PCTL formula [30])
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

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

2.2.2 State explosion problem

The *state-explosion problem* occurs when the size of a model state space grows exponentially as the number of state variables in the system increases [6]. As a finite-state abstraction of a system, each state in a model consists of state variables, which encode a property of the system it models. However, due to the combinatorial explosion, the state-space grows exponentially in size. For example, consider a distributed software system, in which a *global state* is a composition of

1. N 8-bit integer variables, and
2. M communication channels of capacity Q.

The total number of global states is $2^{8N}Q^M$. The size of state-space grows exponentially as we add new variables and communication channels. 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 the state-explosion problem and to reduce the computational cost is to use *statistical model checking*.

2.3 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 a number of traces from $\mathcal{M}$, monitor if each trace satisfies Φ, then estimate probability $P(\mathcal{M} \models \Phi)$ by a statistical inference [1].
Given a stochastic model $\mathcal{M}$ and a property Φ, statistical model checking solves the *qualitative* and *quantitative* problem[1]:

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.

	Quantitative	Qualitative
Input	Model $\mathcal{M}$ Property $P_{=?}(\Phi)$	Model $\mathcal{M}$ Property $P_J(\Phi), J \subseteq [0, 1]$
Output	$p = P(\mathcal{M} \models \Phi)$	$P(\mathcal{M} \models \Phi) \in J$?

Table 2.1: Input and output of quantitative and qualitative statistical model checking

2.3.1 Statistical model checking of quantitative properties.

Given an *approximation parameter* ϵ and a *confidence parameter* 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

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

inequality to estimate N for Bernoulli trials. Hoeffding [24] later extends Chernoff inequality to general cases.
Let $Sat(N)$ be number of satisfying traces in N sampled traces. By applying Chernoff-Hoeffding inequality we obtain

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

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

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

For example, given approximation parameter $\epsilon = 0.01$ and confidence parameter $\alpha = 0.05$, we obtain $N \geq 147556$. However, if we want a more precise estimation, for example $\epsilon = 0.005$, then the total number of simulation runs $N \geq 590221$. In other words, the total number of simulation runs is proportional to the quadratic reciprocal of the approximation parameter ϵ. The estimation algorithm *Approximate Probabilistic Model Checking* (APMC in short) is described in detail in [23].

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 to be evaluated $\mathcal{D}$

Output: $\hat{p}$: an estimation of p, $(p = Pr(\mathcal{D} \models \varphi))$

```
1: procedure SMC-APMC
2:     N ← 4 ln(2/α) / ε^2
3:     A ← 0
4:     i ← 1
5:     while i ≤ N do
6:         Simulate a trace t from D by discrete-event simulation.
7:         if t ⊨ φ then
8:             A ← A + 1
       return A/N
```

The Chernoff-Hoeffding gives a lower bound of total simulation runs N to estimate a satisfaction property with a specific error margin.

2.3.2 Statistical model checking of qualitative properties.

Wald [49] introduces the Sequential Probability Ratio Test (SPRT in short) to perform hypothesis tests on sequential statistical analysis, in which the sample size N is not known *a priori*. SPRT updates a cumulative sum and stops when it has enough information to accept the null hypothesis or alternative hypothesis statistically.
Younes [50] 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 $\Phi = P_{\geq p}(\varphi), p \in [0, 1]$. The SPRT method tests the following null hypothesis and alternative hypothesis with approximation width ϵ:

$$H_0 : \hat{p} \geq p + \epsilon$$
$$H_1 : \hat{p} < p - \epsilon$$

Let A be the set of satisfying traces among total N simulated traces from $\mathcal{D}$. Given probability of success in a single Bernoulli trial is p, we have

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

As a new trace t is collected from a simulation run on $\mathcal{D}$, SPRT updates the set of satisfying traces A. It then computes the likelihood ratio based on A:

$$R = \frac{P(A|p+\epsilon)}{P(A|p-\epsilon)}$$

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

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

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: $(\mathcal{D} \models \Phi)$ or $(\mathcal{D} \not\models \Phi)$

procedure SMC-SPRT
  $p_0 = \frac{\beta}{1-\alpha}$
  $p_1 = \frac{\alpha}{1-\beta}$
  $N \leftarrow 0$
  $A \leftarrow 0$
  while `True` **do**
    Simulate a trace T from $\mathcal{D}$
    if $T \models \Phi$ **then**
      Append T to A
    $R \leftarrow \frac{P(A|p+\delta)}{P(A|p-\delta)}$
    if $R \geq p_1$ **then**
      Accept H_1, return $(\mathcal{D} \not\models \Phi)$
    else
      if $R \leq p_0$ **then**
        Accept H_0, return $(\mathcal{D} \models \Phi)$

2.4 Parametric model checking

2.4.1 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 2.4.1 (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 [28].

Definition 2.4.2 (Parametric discrete-time Markov chain)
A *parametric discrete-time Markov chain* $\mathcal{M}_\theta$ 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 2.4.3
Parameter value Let $\mathcal{M}_\theta = (S, \theta, \mathbf{P}, \iota_{init}, AP, L)$ be a parametric DTMC, $\theta = \{\theta_1, \ldots, \theta_n\}$. A value of θ is a map $v : \theta \to \mathbb{R}^n$. 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 9 (Parametric Knuth-Yao die [35])
An algorithm by Knuth [35] to simulate a 6-faced die by two possibly unfair coins with probabilities of showing head are p and q.

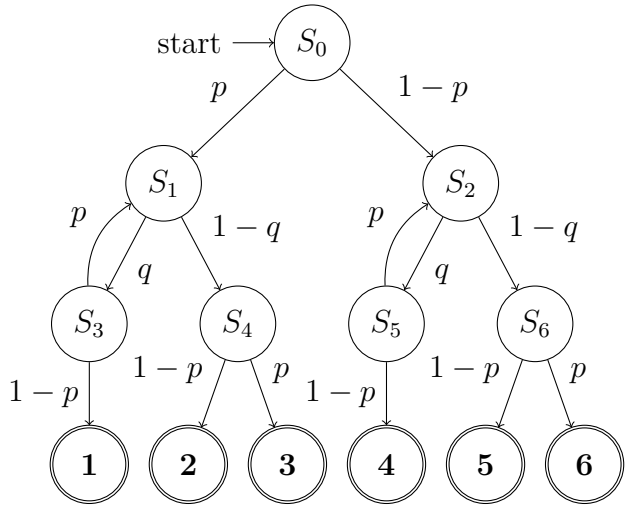


Figure 2.4: Knuth Die by two possibly unfair coins.

2.4.2 Parameter synthesis of parametric DTMC

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

Definition 2.4.4
Parameter synthesis (Katoen 2016)[31] 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 [30] summarizes the following methods on parameter synthesis of parametric DTMC:

1. *Computing symbolic reachability probabilities*: using states elimination to obtain symbolic rational function of a reachability property [9] [18].

2. *Candidate region generation and checking*: partition the parameter space into *safe* and *unsafe* regions using non-linear interval arithmetic (Kwiatkowska [37]) or SMT solvers such as Z3 [10].

3. *Parameter lifting* is another parameter space partitioning method; it introduces new transitions into the original parametric DTMC 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 reachability probabilities [9].

Example 10 (Parametric Knuth-Yao die [35])
Using the example of parametric Knuth-Yao die as in Figure 2.4, we obtain the following symbolic rational functions for each possible outcome of rolling the die.

```
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 11
Given a parametric DTMC 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 $(p,q) \in [0,1]\times[0,1]$ so that $\mathcal{M}_{(p,q)} \models \Phi$. We perform a Monte Carlo search on parameter space. Specifically, we sample $p*$ and $q*$ independently using uniform distribution $Uniform(0,1)$, then discard the pairs $(p,q*)$ that instantiate $\mathcal{M}_{(p*,q*)} \not\models \Phi$. The accepted pairs of $(p*,q*)$ are then visualized as follow:*

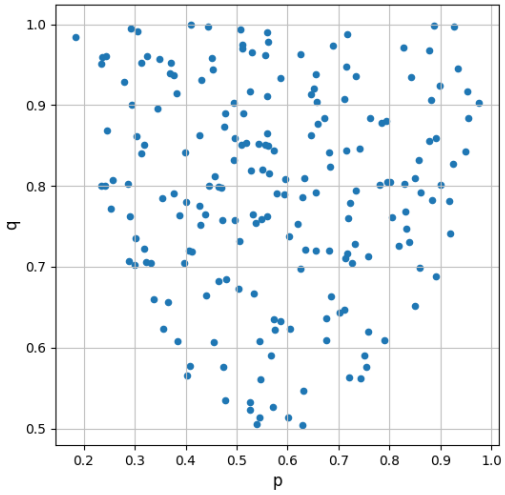


Figure 2.5: Samples of $(p*,q*)$ that instantiate $\mathcal{M}_{(p*,q*)} \models \Phi$.

2.4.3 Summary

This chapter introduced the essential theoretical concepts of discrete-time models, probabilistic temporal logic, and parameter synthesis. 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. Afterward, we discuss different sampling algorithms to approximate the posterior distribution when no conjugations are available. We also present a likelihood-free method to exploit if 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: the frequentist approach and the Bayesian approach. In the frequentist approach, the estimation of θ based on long-run property, that is, given a large enough sample size, the expected value of parameter estimation $\hat{\theta}$ is equal to θ. Therefore, the frequentist approach requires gathering a large amount of data to deliver a close estimation $\hat{\theta}$. The main advantage of the Bayesian approach over the frequentist approach is that it requires fewer data to obtain an estimation $\hat{\theta}$.
In the 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

- $\pi(\theta)$ is the *prior distribution*.
- $P(D_{obs}|\theta)$ is the *likelihood*.
- $\int_\theta P(D_{obs}|\theta)\pi(\theta)d\theta$ 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 $\pi(\theta|D_{obs})$ we estimate the parameter $\hat{\theta}$ using Bayesian posterior mean.

Definition 3.1.2 (Bayesian posterior mean)

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

In case we have samples from posterior distribution, for example a trace of N parameter values $(\theta_1, \ldots, \theta_N)$ sampled from the posterior distribution $\pi(\theta|D_{obs})$, the discrete form of posterior mean is used:

$$\hat{\theta} \approx \mathbf{E}[\theta] \approx \sum_\theta \theta\pi(\theta|D_{obs})$$

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).

$$P(\theta \in C|D_{obs}) = \int_C \pi(\theta|D_{obs})d\theta = 1 - \alpha$$

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 [44]. 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.
- $0 \leq \alpha \leq 1$: confidence level.

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.

3.1.3 Selection of prior distribution

Theoretically, in Bayesian inference, prior distribution can be of any distribution family. However, selecting prior distribution that is too different from the actual distribution of parameters can lead to a false propagation of beliefs and degrade the inference results. It is suggested by [42] that in case of no prior knowledge exists, Uniform distribution is preferable since it is less likely to propagate false beliefs to the inference. In this thesis, we use uniform distribution as the prior distribution on model parameters.

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 the probability distribution function need to be re-estimated. Applying conjugated posterior when it is possible gives advantages:

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

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

Lemma 12 (Binomial-Beta Conjugation)
Binomial distribution is conjugated to beta distribution.

Proof. The observed data $D = (x_1, \ldots, x_n)$ 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 13 (Multinomial-Dirichlet Conjugation)
Multinomial distribution is conjugated to Dirichlet distribution.

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

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

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

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

We obtain

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

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

More detailed description in these cases can be found in [48] and [4]. We summarize the necessary results in the following table:

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

However, posterior conjugation applies to a subset of prior and likelihood functions. In Bayesian inference, it is common that the posterior distribution has no analytical form or its analytical form is difficult to sample directly. In these cases, we can develop 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

If 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). Firstly introduced by Metropolis [38] and later generalized by Hastings [21] Metropolis-Hastings algorithm is a *Monte Carlo Markov Chain* algorithm.

Algorithm 4 Metropolis-Hastings Algorithm

Input:

- Model $\mathcal{M}_\theta$
- D_{obs}: observation data
- $P(D|\theta)$: likelihood function
- $\pi(\theta)$: 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.

procedure METROPOLIS-HASTINGS
    Draw θ_{cand} from $\pi(\theta)$
    $\theta_1 \leftarrow \theta_{cand}$
    $w_1 \leftarrow \ln(P(D_{obs}|\theta_{cand}))$
    $i \leftarrow 2$
    while $i \leq N_{MH}$ **do**
        Draw θ_{cand} from $Q(\theta'|\theta_{i-1})$
        if $\ln(P(D_{obs}|\theta_{cand})) - \ln(P(D_{obs}|\theta_{i-1})) > 0$ **then**
            $\theta_i \leftarrow \theta_{cand}$
            $w_i \leftarrow \ln(P(D_{obs}|\theta_{cand}))$
            $i \leftarrow i + 1$
        else
            Draw a random number u from $Uniform(0, 1)$
            if $u \leq \xi$, (ξ small, e.g 10^{-2}) **then**
                $\theta_i \leftarrow \theta_{cand}$
                $w_i \leftarrow \ln(P(D_{obs}|\theta_{cand}))$
                $i \leftarrow i + 1$
    Return $(\theta_1, \ldots, \theta_N)$, $(w_1, \ldots, w_N)$

The likelihood function can be implemented as log-likelihood to avoid underflow error. Using the MH algorithm, we can estimate the parameter by posterior mean without knowing the analytical form of posterior distribution itself. A special case of the MH algorithm is the Gibbs sampling algorithm [16], in which candidates are accepted with probability 1. Gibbs sampling can be used to draw samples from multivariate posteriors when the parameter is of high dimensional. Quality of data

can also impact the estimation result from Metropolis-Hastings algorithm; in case of experiment data is acquired from a small sample size, statistical techniques to improve data quality can be used, such as *bootstrapping* [14].
Advantages of Metropolis-Hastings are:

+ Parameter transition only needs the computation of the 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 canceled out, and likelihood can be replaced by log-likelihood, Metropolis-Hastings simplifies the computation of the Bayes formula and avoid infinitesimal values.

+ Simple to implement.

Disadvantages of Metropolis-Hastings are

- Particle in Metropolis-Hastings algorithm moves in a linear Markov chain; it is highly probable to be stuck in a local maximum or minimum.

- Not parallelizable; since there is only one linear chain, and the current step depends on previous step, the Metropolis-Hastings algorithm does not scale up to multi-processors.

The following algorithm, *Sequential Monte Carlo*, addresses the issues of Metropolis-Hastings.

Sequential Monte Carlo

The Sequential Monte Carlo method is firstly proposed by Del [12]. Instead of having one particle moving in its parameter space, Sequential Monte Carlo estimates using N particles moving independently. In its initial step, Sequential Monte Carlo draws a parameter candidate from the prior distribution. In each iteration, it then mutates parameter candidates through a series of *perturbation kernels* and selects parameter candidates for the next iteration with regarding to their weights. By sampling from N independently moving particles, Sequential Monte Carlo method has a significant advantage of being easily parallelizable. Furthermore, Daviet [8] shows that Sequential Monte Carlo delivers better performance on approximating multimodal distribution compared to the Metropolis-Hastings algorithm.

Algorithm 5 Sequential Monte Carlo Algorithm

Input:

- Model $\mathcal{M}_\theta$
- D_{obs}: observation data
- $\pi(\theta)$: prior distribution
- $P(D|\theta)$: Likelihood function.
- $Q(\theta^t|\theta^{t-1})$: Transition kernel.
- $F_t(\theta^t|\theta_1^{t-1},\ldots,\theta_N^{t-1})$: Perturbation kernels
- 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 θ from $\pi(\theta)$
    $\theta_i \leftarrow \theta$
    $w_i \leftarrow P(D_{obs}|\theta_i)$
    $i \leftarrow i+1$
  $t \leftarrow 1$
  while $t \leq M$ **do**
    $i \leftarrow 1$ ▷ SMC correction step
    while $i \leq N$ **do**
      $w_i' \leftarrow \frac{w_i}{\sum_{i=1}^{N} w_i}$
    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 perturbation step
      Draw $\hat{\theta}_i^t$ from $F_t(\theta^t|\theta_1^{t-1},\ldots,\theta_N^{t-1}), 1 \leq t \leq M$
      $(\theta_1^*,\ldots,\theta_{N_{MH}}^*),(w_1^*,\ldots,w_{N_{MH}}^*) \leftarrow Metropolis-Hastings(\hat{\theta}_i^t)$
      $\theta_i \leftarrow \theta_{N_{MH}}^*$
      $w_i \leftarrow w_{N_{MH}}^*$
  Return $(\theta_1,\ldots,\theta_N)$, $(w_1,\ldots,w_N)$

Selection of kernel function for SMC is mentioned by Filippi [15], and Silk [45]. Sequential Monte Carlo algorithm has several advantages compared to Metropolis-Hastings 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 data-parallelism trivially, in contrast to Metropolis-Hastings, where no parallelization is possible.

However, Sequential Monte Carlo also has disadvantages:

- Selection of perturbation and transition kernel is not trivial.

- More difficult to implement.

Approximate Bayesian Computation

The methods mentioned before are used with an assumption that the likelihood $P(D_{obs}|\theta)$ has an analytical form; the analytical form 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 $P(D_{obs}|\theta)$, or replace it by other measures. *Approximate Bayesian Computation* (or *ABC* in short) [47] is a widely used likelihood-free method for approximating posterior distribution. Instead of estimating the likelihood $P(D|\theta)$ directly, we define a distance measure $\delta(D_1, D_2)$ where D_1 and D_2 denote observable data. Given a parameter candidate $\hat{\theta}$ that specifies a model $\mathcal{M}_\theta$. The ABC algorithm accepts $\hat{\theta}$ if a simulation run on $\mathcal{M}_\theta$ delivers observable data D_{sim} such that $\delta(D_{obs}, D_{sim}) < \epsilon$, where $\epsilon \in \mathbf{R}_{\leq 0}$ is the distance threshold. ABC algorithm can be used together with Markov chain Monte Carlo algorithm (ABC-MCMC [43] [41]), or with Sequential Monte Carlo sampling algorithm (ABC-SMC [46] [39]). Advantages of Approximate Bayesian Computation are:

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

+ Easy to implement.

However, Approximate Bayesian Computation has drawbacks:

- How to select a distance threshold ϵ so that the posterior is closely approximated? [46]

- How to choose a summary statistic to capture sufficient information? [7]

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 π(θ)
3:     i ← 1
4:     while i ≤ N do
5:         Draw a random particle θ from π(θ)
6:         Simulate data D_sim from M_θ
7:         if d = δ(D_sim, D_obs) < ϵ then
8:             θ_i ← θ
9:             w_i = d
10:    Return (θ_1, ..., θ_N), (w_1, ..., w_N)
```

3.2 Summary

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

Chapter 4

Bayesian frameworks for parameter synthesis.

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

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

The set of satisfying parameter values $(\theta_1, \ldots, \theta_N)$ has corresponding weights $(w_1, \ldots, w_N)$. The weight $w_i, 1 \leq i \leq N$ represents the likelihood $P(D_{obs}|\theta_i)$ or its approximation. We design two frameworks for two different use cases. In the first use cases, symbolic rational functions are available for (i) the evaluation of steady-state distribution, and (ii) the evaluation of property Φ. In the second use case, we assume that there is no rational function obtainable for steady-state distribution and Φ. Instead, we (i) use simulation runs to estimate the steady-state distribution, and (ii) use statistical model checking to verify the property of interest Φ.

4.1 Generic framework

We present the generic framework for Bayesian parameter synthesis of parametric discrete-time Markov chain (concept shown in Fig. 4.1). The generic framework takes a parametric discrete-time Markov chain, a property of interest, and steady-state data as input. In the core of the framework, we use Sequential Monte Carlo to sample the parameter space. The generic framework is based on the Sequential Monte Carlo algorithm. However, in the specific implementations of the generic framework, there are two important differences in each iteration of the Sequential Monte Carlo algorithm.

1. *Computing likelihood*: do we compute the (i) exact likelihood, or (ii) do we approximate it?

2. *Model checking the property of interest*: do we use (i) rational functions to evaluate the property of interest Φ, or (ii) do we use statistical model checking?

Algorithm 7 Generic framework for Bayesian 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:
- $(\theta_1, \ldots, \theta_{N_{MH}})$, $(w_1, \ldots, w_{N_{MH}})$: N_{MH} sampled particles and their corresponding weights.

1: **procedure** GENERIC-BAYESIAN-MONTE CARLO
2: $i \leftarrow 1$
3: **while** $i \leq N$ **do**
4: Sample θ with corresponding weight w
by Sequential Monte Carlo sampling algorithm.
5: Verify instantiated model $\mathcal{M}_\theta$ against Φ
6: **if** $\mathcal{M}_\theta \models \Phi$ **then**
7: $\theta_i \leftarrow \theta$
8: Estimate w_i as exact or approximated likelihood $P(D_{obs}|\theta)$
9: Estimate $\hat{\theta}$ using posterior mean.
10: Compute $\hat{p} = P(\mathcal{M}_{\hat{\theta}} \models \Phi)$
11: Return $(\theta_1, \ldots, \theta_N)$, $(w_1, \ldots, w_N)$, $\hat{\theta}$, $\hat{p}$

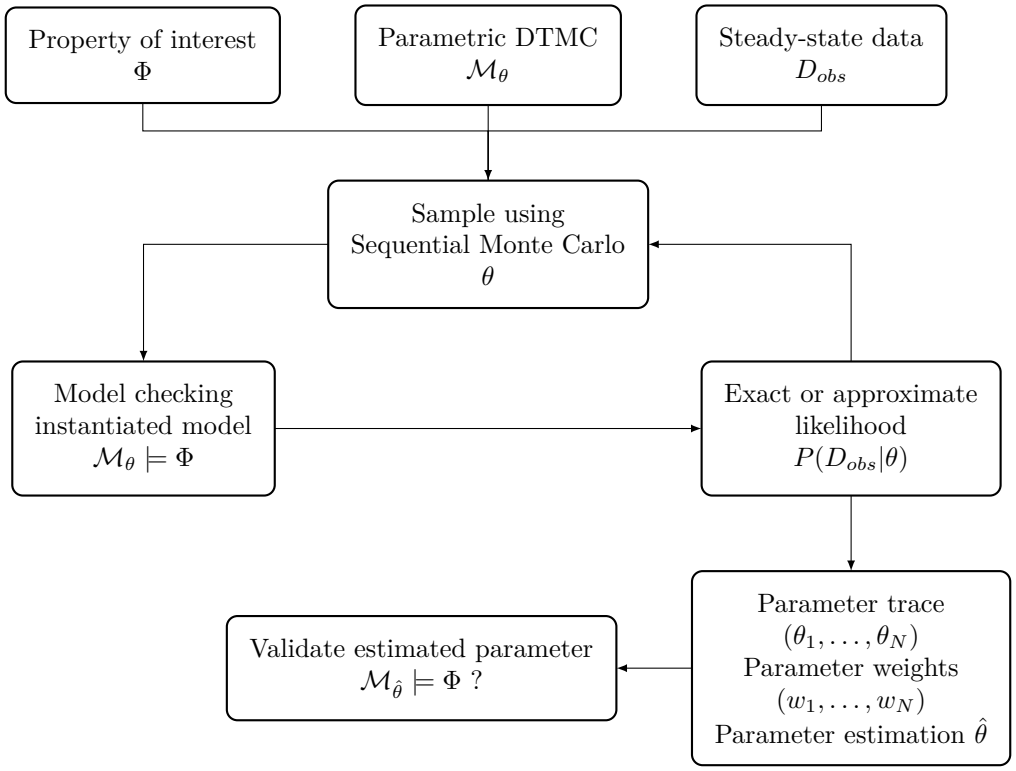


Figure 4.1: Generic framework for Bayesian parameter synthesis of parametric DTMC.

4.1.1 Selection of perturbation kernel

An analysis on the seletion of perturbation kernel is presented by Filippi [15] and Silk [45]. As there is no prior knowledge on the covariance among multi-dimensional parameters, we select component-wise perturbation kernel, in which the perturbation function applies to each dimension independently. We also use uniform perturbation kernel to avoid the propagation of false beliefs. Assume model parameter is of k-dimensional, $\theta = (\theta_1, \ldots, \theta_k)$. Each component θ_i, $i \in [1, k]$ is perturbated at pertubation t using an uniform kernel $Uniform(\theta_i - \delta, \theta_i + \delta)$, where

$$\delta = \frac{1}{2}(\max(\theta_i^1, \ldots, \theta_i^{t-1}) - \min(\theta_i^1, \ldots, \theta_i^{t-1}))$$

In the SMC algorithm with exact likelihood is available, we select Metropolis-Hastings transition kernel $Q(\theta^t|\theta^{t-1})$ identical to the perturbation kernel.

4.2 RF-SMC framework

We design a Sequential Monte Carlo framework with rational functions, using a modified Metropolis-Hastings algorithm to mutate particles in each perturbation.

Algorithm 8 Metropolis-Hastings 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.
- D_{obs}: observed data.
- $P(D_{obs}|\theta)$: likelihood function.
- N_{MH}: length of particle trace.
- $Q(\theta^t|\theta^{t-1})$: transition kernel.

Output:

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

procedure RF-MH
  $sat \leftarrow False$
  while $sat = False$ **do**
    Draw θ_{cand} from $\pi(\theta)$
    Evaluate $val \leftarrow RF_\Phi(\theta)$
    if val satisfies the boundary of Φ **then**
      $sat \leftarrow True$
  $\theta_1 \leftarrow \theta_{cand}$
  $w_1 \leftarrow \ln(P(D_{obs}|\theta_{cand}))$
  $i \leftarrow 2$
  while $i \leq N_{MH}$ **do**
    $sat \leftarrow False$
    while $sat = False$ **do**
      Draw θ_{cand} from $Q(\theta'|\theta_{i-1})$
      Evaluate $val \leftarrow RF_\Phi(\theta)$
      if val satisfies the boundary of Φ **then**
        $sat \leftarrow True$
    if $\ln(P(D_{obs}|\theta_{cand})) - \ln(P(D_{obs}|\theta_{i-1})) > 0$ **then**
      $\theta_i \leftarrow \theta_{cand}$
      $w_i \leftarrow \ln(P(D_{obs}|\theta_{cand}))$
      $i \leftarrow i + 1$
    else
      Draw a random number u from $Uniform(0, 1)$
      if $u \leq \xi$, (ξ small, e.g 10^{-2}) **then**
        $\theta_i \leftarrow \theta_{cand}$
        $w_i \leftarrow \ln(P(D_{obs}|\theta_{cand}))$
        $i \leftarrow i + 1$
  Return $(\theta_1, \ldots, \theta_{N_{MH}})$, $(w_1, \ldots, w_{N_{MH}})$

The only difference to the original Metropolis-Hastings algorithm is that the following algorithm only accepts parameter values θ that instantiate DTMC models which satisfy the property of interest.

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 perturbation 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.

Output:

- $(\theta_1,\ldots,\theta_N)$, $(w_1,\ldots,w_N)$: N sampled particles and their corresponding weights.
- $\hat{\theta}$: estimated model parameter.

procedure RF-SMC
  $i \leftarrow 1$
  while $i \leq N$ **do** ▷ SMC initialization
    Draw θ from $\pi(\theta)$
    $\theta_i \leftarrow \theta$
    $w_i \leftarrow P(D_{obs}|\theta_i)$
    $i \leftarrow i + 1$
  $t \leftarrow 1$
  while $t \leq M$ **do**
    $i \leftarrow 1$ ▷ SMC correction step
    while $i \leq N$ **do**
      $w'_i \leftarrow \frac{w_i}{\sum_{i=1}^{N} w_i}$
    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 perturbation step
      Draw $\hat{\theta}_i^t$ from $F_t(\theta^t|\theta_1^{t-1},\ldots,\theta_N^{t-1}), 1 \leq t \leq M$
      $(\theta_1^*,\ldots,\theta_{N_{MH}}^*), (w_1^*,\ldots,w_{N_{MH}}^*) \leftarrow RF - MH(\hat{\theta}_i^t)$
      $\theta_i \leftarrow \theta_{N_{MH}}^*$
      $w_i \leftarrow w_{N_{MH}}^*$
  Estimate $\hat{\theta}$ using posterior mean.
  Compute $\hat{p} = P(\mathcal{M}_{\hat{\theta}} \models \Phi)$
  Return $(\theta_1,\ldots,\theta_N)$, $(w_1,\ldots,w_N)$, $\hat{\theta}$, $\hat{p}$

4.3 SMC-ABC-SMC framework

Without the availability of analytical form to evaluate the steady-state distribution and the property of interest, 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 absence of likelihood suggests exploiting *likelihood-free methods*. In this framework, we use Approximate Bayesian Computation in combination with the Sequential Monte Carlo method.
- **Absence of rational functions for evaluation of property of interest:** Statistical Model Checking then checks the satisfaction of an instantiated model to a bounded path property.

For this case, we present Statistical Model Checking, Approximate Bayesian Computation - Sequential Monte Carlo method *(SMC-ABC-SMC)* framework. SMC-ABC-SMC differs from RF-SMC only in their perturbation step.

- In the SMC-ABC-SMC framework, we work with a *likelihood-free* setup in which there is no analytical form to evaluate the likelihood. As there is no likelihood function, we apply Approximate Bayesian Computation and accept the first particle whose simulation runs satisfies the distance threshold.
- There is also no rational function for the property of interest Φ, so we apply Statistical Model Checking with confidence level α and approximation parameter δ.

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

Input:

- $\mathcal{M}_\theta$: parametric DTMC of parameter θ
- Φ: bounded reachability property of interest.
- D_{obs}: observed data
- $\pi(\theta)$: prior distribution on θ.
- N: number of particles in the Sequential Monte Carlo trace.
- M perturbation kernels $F_t(\theta^t|\theta_1^{t-1},\ldots,\theta_N^{t-1}), 1 \leq t \leq M$
- ϵ: distance threshold for Approximate Bayesian Computation.
- δ, α: indifference and α-level for Statistical Model Checking using SPRT method.

Output:

- $(\theta_1,\ldots,\theta_N), (w_1,\ldots,w_N)$: N sampled particles and their corresponding weights.
- $\hat{\theta}$: estimated model parameter.

procedure SMC-ABC-SMC
  $i \leftarrow 1$
  while $i \leq N$ **do** ▷ SMC initialization
    Draw θ from $\pi(\theta)$
    $\theta_i \leftarrow \theta$, $w_i \leftarrow 1$
  $t \leftarrow 1$
  while $t \leq M$ **do**
    $i \leftarrow 1$ ▷ SMC correction step
    while $i \leq N$ **do**
      $w_i' \leftarrow \frac{w_i}{\sum_{i=1}^{N} w_i}$
    Sample (with replacement) $(\theta_1^t,\ldots,\theta_N^t)$
      from $(\theta_1^{t-1},\ldots,\theta_N^{t-1})$, $(w_1^{t-1},\ldots,w_N^{t-1})$ ▷ SMC selection step
    $i \leftarrow 1$
    while $i \leq N$ **do** ▷ SMC perturbation step
      $rejected \leftarrow True$
      while $rejected == True$ **do**
        $sat \leftarrow False$
        while $sat = False$ **do**
          Draw $\hat{\theta}_i^t$ from $F_t(\theta^t|\theta_1^{t-1},\ldots,\theta_N^{t-1}), 1 \leq t \leq M$
          if $SPRT - SMC(\mathcal{M}_{\hat{\theta}^t}, \Phi, \epsilon, \delta)$ is SAT **then**
            $sat \leftarrow True$
        Simulate D_{sim} from $(\mathcal{M}_{\hat{\theta}^t})$
        if $Distance(D_{sim}, D_{obs}) < \epsilon$ **then**
          $rejected \leftarrow False$
          $\theta_i \leftarrow \hat{\theta}^t$, $w_i \leftarrow d$
  Estimate $\hat{\theta}$ using posterior mean.
  Compute $\hat{p} = P(\mathcal{M}_{\hat{\theta}} \models \Phi)$
  Return $(\theta_1,\ldots,\theta_N)$, $(w_1,\ldots,w_N)$, $\hat{\theta}$, $\hat{p}$

4.4 Summary

In this chapter we presents two Sequential Monte Carlo based frameworks for Bayesian parameter synthesis of parametric DTMC:

1. *RF-SMC*: rational functions based Sequential Monte Carlo, and
2. *SMC-ABC-SMC*: simulation based Sequential Monte Carlo

In the following chapter, we benchmark the frameworks using different parametric DTMCs to evaluate the performances of presented frameworks.

Chapter 5

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 parametric DTMC models and formalize a property of interest for parameter synthesis.
3. Select model true parameters arbitrarily random and generate synthetic data from it.
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.

The first case study is *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[19]. The third case study is a stochastic SIR model of epidemics; 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

5.1 ZeroConfiguration Protocol

5.1.1 System description

Zero-configuration protocol (*zeroconf* for short) [13] 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, or abort with error.

```
1: procedure ZEROCONF
2:     Select an address ip randomly
3:     i = 1
4:     while i ≤ N do
5:         Broadcast message asking if ip is already in use.
6:         if Received reply that ip is in use then
7:             Select an address ip randomly
8:             Continue loop.
9:         if timeout then
10:            if i = N then
11:                Return ip
12:            i ← i + 1
13:            Continue loop.
```

5.1.2 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 parametric DTMC as a formalism of the Zeroconf protocol of N probes.

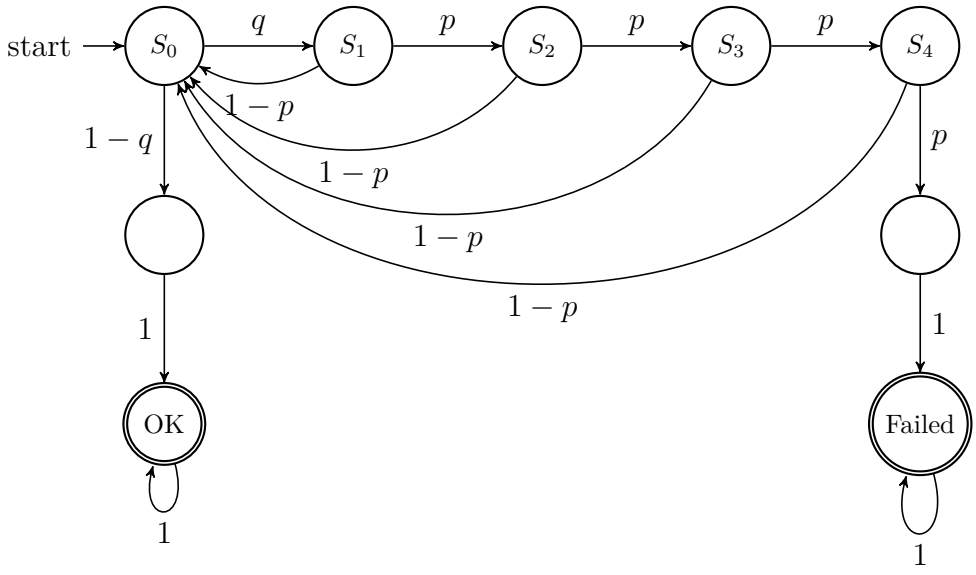


Figure 5.1: Example of an IPv4 Zeroconf model with 4 probes

We want to verify the following property: *"Eventually, an IP is successfully allocated with at most N probes with probability of at least 75 percents."* In PCTL formula

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

5.1.3 Evaluation

True parameters and synthetic data

We select a true parameter (p, q) arbitrarily random.

Model $\mathcal{M}$	Zeroconf, 4 probes	Zeroconf, 10 probes
Number of BSCCs	2	2
Number of states	9	14
True parameter $\theta = (p, q)$	(0.105547, 0.449658)	(0.197779, 0.621824)
Number of samples	10000	10000
Synthetic data D_{obs}	(41, 9959)	(22, 9978)
Property of interest Φ	$P_{\geq 0.75}(\texttt{true}\mathsf{U}^{\leq 4}(\texttt{"OK"}))$	$P_{\geq 0.75}(\texttt{true}\mathsf{U}^{\leq 10}(\texttt{"OK"}))$
Satisfaction property $P(\mathcal{M}_\theta \models \Phi)$	0.946409	0.952067

Table 5.1: Synthetic data for Zeroconf model of 4 and 10 probes.

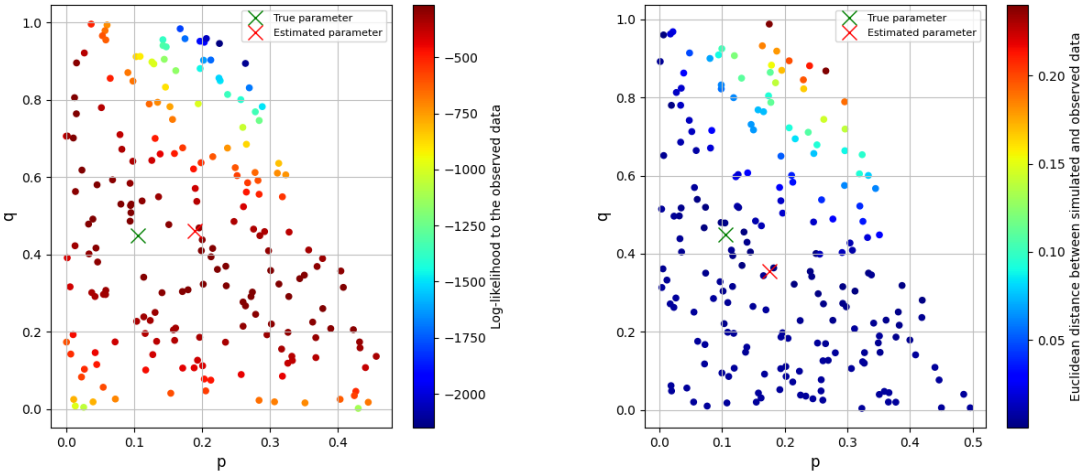
Parameter synthesis results

In the following parameter synthesis experiment, we set the number of samples in Sequential Monte Carlo $N = 200$ and the number of perturbation kernels $M = 19$. Perturbation and transition kernel are selected as described in Chapter 4. In RF-SMC, the length of Metropolis-Hasting is $N_{MH} = 50$. In SMC-ABC-SMC, the distance threshold is $\epsilon = 0.25$.
First, we present the parameter synthesis result for Zeroconf parametric DTMC of 4 probes. The property of interest is

$$P_{\geq 0.75}(\texttt{true}\mathsf{U}^{\leq 4}(\texttt{"OK"}))$$

Method	RF-SMC	SMC-ABC-SMC
Estimated parameter $\hat{\theta}$	(0.188956, 0.460554)	(0.176469, 0.355322)
True parameter θ	(0.105547, 0.449658)	(0.105547, 0.449658)
L2 distance $\|\theta, \hat{\theta}\|_2$	0.084117	0.118023
$P(\mathcal{M}_{\hat{\theta}} \models \Phi)$	0.893715	0.918133

Table 5.2: Parameter estimation results for Zeroconf model of 4 probes.



(a) Sampled particles using Rational Functions SMC

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

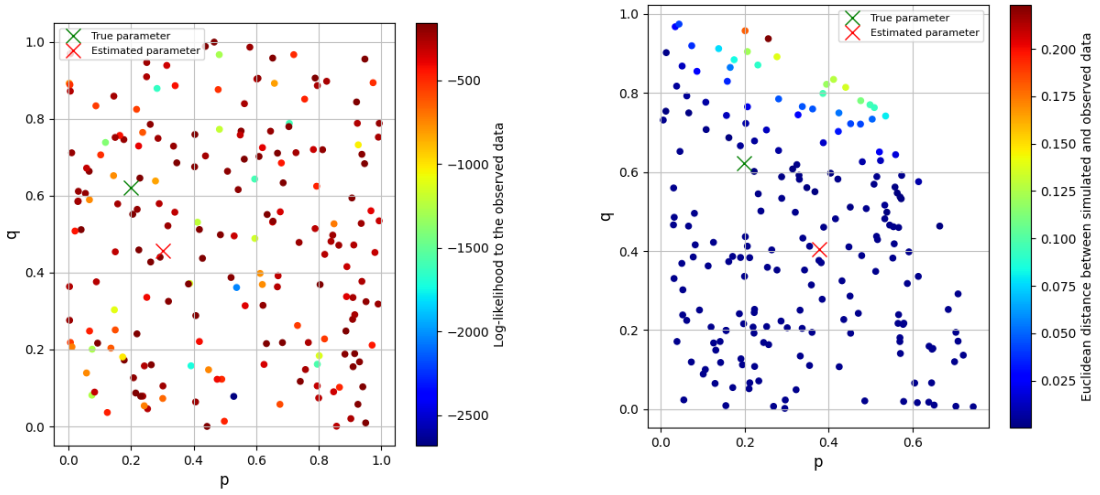
Figure 5.2: Parameter synthesis results for Zeroconf model of 4 probes.

We present parameter synthesis results for Zeroconf parametric DTMC of 10 probes. We use the same RF-SMC and SMC-ABC-SMC framework configuration as in the parameter synthesis of the Zeroconf model with four probes.

$$P_{\geq 0.75}(\texttt{true}\mathsf{U}^{\leq 10}(\texttt{"OK"}))$$

Method	RF-SMC	SMC-ABC-SMC
True parameter θ	(0.197779, 0.621824)	(0.197779, 0.621824)
Estimated parameter $\hat{\theta}$	(0.301807, 0.457090)	(0.378774, 0.405870)
L2 distance $\|\theta, \hat{\theta}\|_2$	0.194831	0.281772
$P(\mathcal{M}_{\hat{\theta}} \models \Phi)$	0.952067	0.966142

Table 5.3: Parameter estimation results for Zeroconf model of 10 probes.



(a) Sampled particles using RF-SMC

(b) Sampled particles using SMC-ABC-SMC

Figure 5.3: Parameter synthesis results for Zeroconf model of 10 probes.

5.1.4 Performance

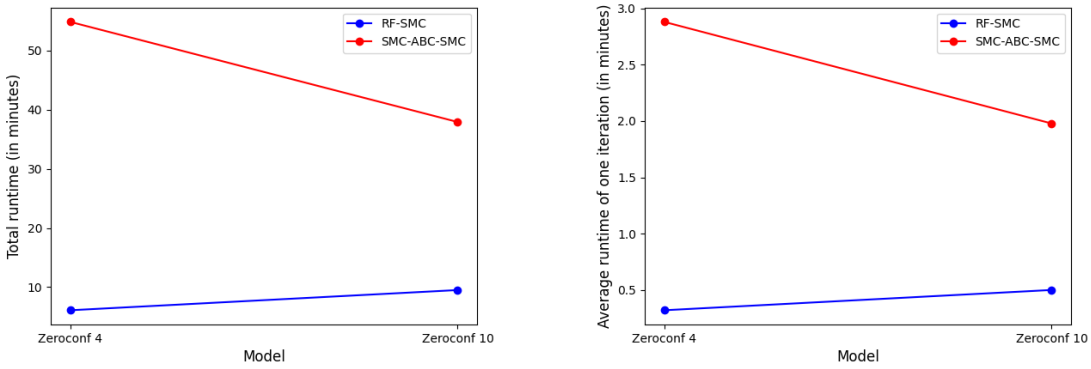
To measure the performance of RF-SMC and SMC-ABC-SMC, we measure the physical runtime to finish *(total runtime)* and the average physical run time of one perturbation kernel *(average perturbation runtime)*.

Method	RF-SMC	SMC-ABC-SMC
Total runtime (minutes)	6.083	54.867
Average perturbation runtime (minutes)	0.32	2.88

Table 5.4: Physical runtime on Zeroconf model with 4 probes.

Method	RF-SMC	SMC-ABC-SMC
Total runtime (minutes)	9.50	37.93
Average perturbation runtime (minutes)	0.501	1.978

Table 5.5: Physical runtime on Zeroconf model with 10 probes.



(a) Total runtime

(b) Average perturbation runtime

Figure 5.4: Physical runtime on Zeroconf model of different sizes.

5.1.5 Discussion

In the experiments with Zeroconf models of 4 and 10 probes, RF-SMC and ABC-SMC have comparable accuracy in parameter estimation. The parameter synthesis results suggest that parameter accepting region and rejecting region are linearly separated in the first quadrant of the 2D Cartesian plane. In terms of performance, rational function evaluation is faster than simulation in the two models. However, the

performance gap is narrower in the 10-probes model. It suggests that the simulation-based method is probably preferable for Zeroconf models with more probes in terms of performance. In the next experiment with models of honey bees colony, we investigate the accuracy and performance of the rational function-based method and simulation-based method with models of more complicated transitions.

5.2 Social feedback in honeybee colonies

5.2.1 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 warns 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 following action (to sting or not to sting) based only on the current state of the environment; we model the number of bees who sting or not sting by Markov population processes. To reduce the complexity of the model, we make another assumption that the states of the bee colony are observed after a 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.

5.2.2 Model and properties

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

1. Each bee releases a unit amount of pheromone immediately after stinging.
2. A bee dies after stinging and releasing a pheromone unit. In other words, no bee can sting more than once.
3. Stinging behavior only depends on the concentration of pheromone in the environment.

Under these assumptions, a bee colony can be viewed as a set of agents (bees) that interact with each other in a closed environment with the appearance of a factor *pheromone*. Afterward, the agent has a probability of committing an action, namely *sting*. The agent is eliminated from the environment after stinging. Assume that we

have a colony of n bees initially. As aforementioned, an individual bee dies after it stings. Thus, at the end of the 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 $n + 1$ BSCCs which encode the population at the end of the experiment.

Semantics of Markov population models for bees colony are developed by [19].

Example 14
Parametric DTMC model of 3 bees. We model a colony of 3 bees using parametric DTMC. 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.*
- k*: bee does not sting in* k *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 parametric DTMC*

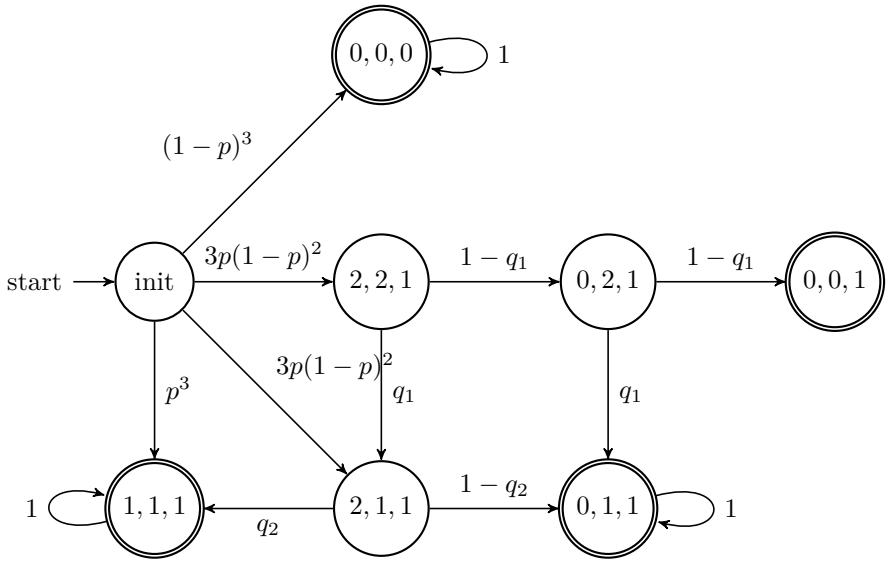


Figure 5.5: Parametric DTMC model of 3 bees with 3 parameters p, q_1, q_2

In studying the defensive behaviors of bee colonies regarding social feedback, we are interested in the number of stimulated bees who decide to sting. Therefore, we verify the following property: *"With a probability of at least 25 percent, at least a half of bees population survives."* In PCTL formula:

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

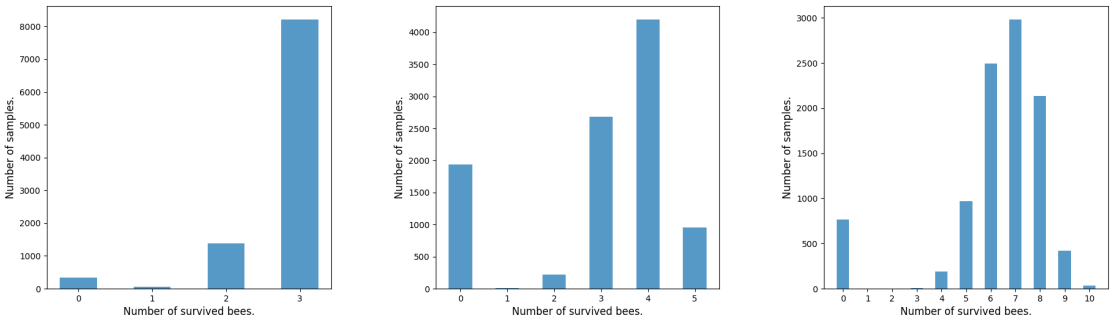
5.2.3 Evaluation

True parameters and synthetic data

We select arbitrary true parameters to generate synthetic data and obtain steady-state distribution. Let S be the number of survived bees at the steady-state. We sample DTMCs instantiated from true parameters 10000 times to obtain steady-state distribution.

Model $\mathcal{M}$	3 bees	5 bees	10 bees
Number of states	13	24	69
Number of BSCCs	4	6	11
True parameter θ	$p = 0.665623$ $q_1 = 0.830401$ $q_2 = 0.839778$	$p = 0.278370$ $q_1 = 0.305994$ $q_2 = 0.489792$ $q_3 = 0.737252$ $q_4 = 0.766581$	$p = 0.222169$ $q_1 = 0.246993$ $q_2 = 0.281934$ $q_3 = 0.446384$ $q_4 = 0.491612$ $q_5 = 0.534611$ $q_6 = 0.569409$ $q_7 = 0.684651$ $q_8 = 0.717139$ $q_9 = 0.800987$
Synthetic data D_{obs}	(344, 54, 1390, 8212)	(1940, 11, 216, 2682, 4200, 951)	(769, 0, 1, 10, 187, 972, 2494, 2982, 2133, 419, 33)
Property of interest Φ	$P_{\geq 0.25}(\text{true}\mathsf{U}(S > 3))$	$P_{\geq 0.25}(\texttt{true}\mathsf{U}(S > 5))$	$P_{\geq 0.25}(\texttt{true}\mathsf{U}(S > 8))$
$P(\mathcal{M}_\theta \models \Phi)$	0.819666	0.780172	0.737244

Table 5.6: True parameter and synthetic data for 3, 5, and 10 bees models.



(a) 3 bees model. (b) 5 bees model. (c) 10 bees model.

Figure 5.6: Histogram of steady-state data obtained from 10000 simulation runs.

Parameter synthesis results

We set the number of samples in Sequential Monte Carlo $N = 200$ and the number of perturbation kernels $M = 19$. In RF-SMC, the length of Metropolis-Hasting is $N_{MH} = 50$. In SMC-ABC-SMC, the distance threshold is $\epsilon = 0.25$. Perturbation and transition kernels are component-wise uniform distribution. Prior distribution is $Uniform(0, 1)$.
Parameter synthesis results for 3, 5, and 10 bees model:

	RF-SMC	SMC-ABC-SMC
True parameter θ	$p = 0.665623$ $q_1 = 0.830401$ $q_2 = 0.839778$	$p = 0.665623$ $q_1 = 0.830401$ $q_2 = 0.839778$
Estimated parameter $\hat{\theta}$	$p = 0.671388$ $q_1 = 0.575026$ $q_2 = 0.525502$	$p = 0.811651$ $q_1 = 0.621073$ $q_2 = 0.544130$
L2 distance $\|\theta, \hat{\theta}\|_2$	0.404992	0.390576
$P(\mathcal{M}_{\hat{\theta}} \models \Phi)$	0.623889	0.595083

Table 5.7: Parameter synthesis result for 3 bees model.

	RF-SMC	SMC-ABC-SMC
True parameter θ	$p = 0.278370$ $q_1 = 0.305994$ $q_2 = 0.489792$ $q_3 = 0.737252$ $q_4 = 0.766581$	$p = 0.278370$ $q_1 = 0.305994$ $q_2 = 0.489792$ $q_3 = 0.737252$ $q_4 = 0.766581$
Estimated parameter $\hat{\theta}$	$p = 0.576565$ $q_1 = 0.589724$ $q_2 = 0.490334$ $q_3 = 0.554397$ $q_4 = 0.524433$	$p = 0.361220$ $q_1 = 0.316007$ $q_2 = 0.545691$ $q_3 = 0.643962$ $q_4 = 0.591206$
L2 distance $\|\theta, \hat{\theta}\|_2$	0.511366	0.222594
$P(\mathcal{M}_{\hat{\theta}} \models \Phi)$	0.623889	0.595083

Table 5.8: Parameter synthesis result for 5 bees model

	RF-SMC	SMC-ABC-SMC
True parameter θ	$p = 0.222169$ $q_1 = 0.246993$ $q_2 = 0.281934$ $q_3 = 0.446384$ $q_4 = 0.491612$ $q_5 = 0.534611$ $q_6 = 0.569409$ $q_7 = 0.684651$ $q_8 = 0.717139$ $q_9 = 0.800987$	$p = 0.222169$ $q_1 = 0.246993$ $q_2 = 0.281934$ $q_3 = 0.446384$ $q_4 = 0.491612$ $q_5 = 0.534611$ $q_6 = 0.569409$ $q_7 = 0.684651$ $q_8 = 0.717139$ $q_9 = 0.800987$
Estimated parameter $\hat{\theta}$	$p = 0.604881$ $q_1 = 0.472557$ $q_2 = 0.281484$ $q_3 = 0.500706$ $q_4 = 0.49340$ $q_5 = 0.495508$ $q_6 = 0.466596$ $q_7 = 0.510167$ $q_8 = 0.474153$ $q_9 = 0.484061$	$p = 0.391313$ $q_1 = 0.485688$ $q_2 = 0.424056$ $q_3 = 0.381489$ $q_4 = 0.440681$ $q_5 = 0.578865$ $q_6 = 0.594232$ $q_7 = 0.564557$ $q_8 = 0.547804$ $q_9 = 0.520006$
L2 distance $\|\theta, \hat{\theta}\|_2$	0.665837	0.487042
$P(\mathcal{M}_{\hat{\theta}} \models \Phi)$	0.933287	0.907478

Table 5.9: Parameter synthesis result for 10 bees model

5.2.4 Performance

To measure the performance of RF-SMC and SMC-ABC-SMC, we measure the physical runtime to finish *(total runtime)* an the average physical run time of one perturbation kernel *(average perturbation runtime)*.

Method	RF-SMC	SMC-ABC-SMC
Total runtime (minutes)	5.917	68.783
Average perturbation runtime (minutes)	0.312	3.614

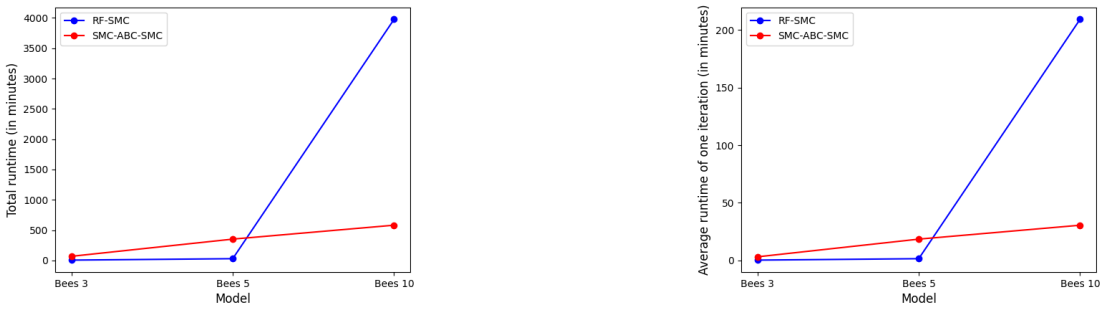
Table 5.10: Physical runtime on 3 bees model.

Method	RF-SMC	SMC-ABC-SMC
Total runtime (minutes)	29.517	352.083
Average perturbation runtime (minutes)	1.553	18.518

Table 5.11: Physical runtime on 5 bees model.

Method	RF-SMC	SMC-ABC-SMC
Total runtime (minutes)	3976.117	581.833
Average perturbation runtime (minutes)	209.237	30.592

Table 5.12: Physical runtime on 10 bees model.



(a) Total runtime

(b) Average perturbation runtime

Figure 5.7: Physical runtime on Zeroconf model of different sizes.

5.2.5 Discussion

The case study with bee colony observed that overall, the simulation-based method delivers estimations that are closer to the true parameter than rational-function-based frameworks. In the experiments with 3 bees and 5 bees models, the results show that the evaluation of rational functions is approximately twelve times less computationally expensive than the simulation-based method. However, the computational cost of the rational-function-based framework RF-SMC grows faster as the model state-space increases in size. From 15 bees model, the experiment shows that RF-SMC is approximately seven times slower than the simulation-based SMC-ABC-SMC, without any significant differences in estimation results. As the parameter models are high-dimensional (more than 3 dimensions), we do not visualize parameter accept regions and reject regions to investigate the parameter space further. In terms of performance, the cost of rational functions evaluation grows intermittently faster from 5 bees model to 10 bees model. It suggests that for a model of more than 10 bees, the simulation-based method is preferable.

5.3 SIR model

5.3.1 System description

SIR model (Kermack [33]) is a model widely used in modeling epidemics. In a SIR model, each individual is of one among three types:

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

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

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

A SIR model is defined by a tuple of initial population (S_0, R_0, I_0) and rates (α, β). From an initial population, we construct a CTMC model using the following algorithm.

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 $\mathcal{C}$

procedure EXPLORE(s,i,r)
  if $i = 0$ **then**
    Mark (s, i, r) as a BSCC.
    Return
  if $(s > 0) \wedge (i > 0)$ **then**
    if State $(s - 1, i + 1, r)$ already visited **then**
      Return
    Add $(s - 1, i + 1, r)$ to state space
    Explore($s - 1$, $i + 1$,r)
  if $(i > 0)$ **then**
    if State $(s, i - 1, r + 1)$ already visited **then**
      Return
    Add $(s, i - 1, r + 1)$ to state space
    Explore(s, $i - 1$,$r + 1$)
procedure SIR-EXPLORE-STATESPACE(s_0, i_0, r_0)
  Sir-Explore-Statespace(s_0, i_0, r_0)

5.3.2 Model and properties

Theorem 15 (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, I_0, R_0 are constants, if there exists a path fragment

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

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 $i \neq i'$). □

Corollary 15.1
A CTMC constructed by Algorithm 12 using reactions R_1, R_2 has BSCCs and the BSCCs are trivial.

Example 16
Example of an SIR CTMC model with initial population $(S_0, I_0, R_0) = (3, 1, 0)$

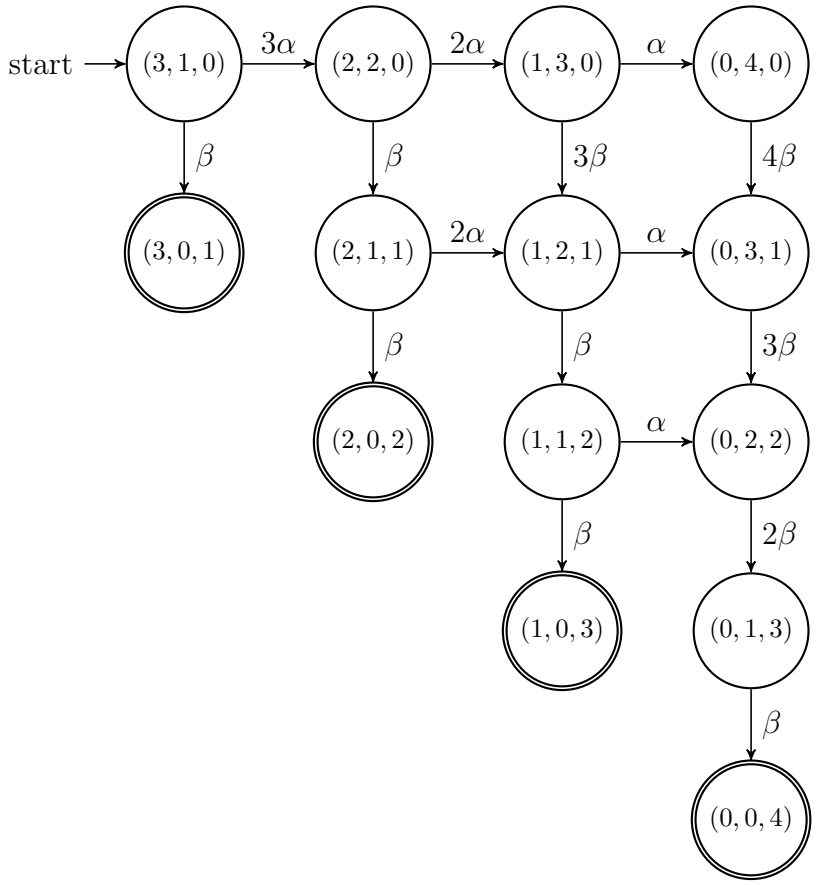


Figure 5.8: $SIR(3, 1, 0)$ CTMC model with parameters(α, β)

Example 17
Uniformize the chain with uniformization rate $(3\alpha + 4\beta)$, we derive the following uniformized DTMC:

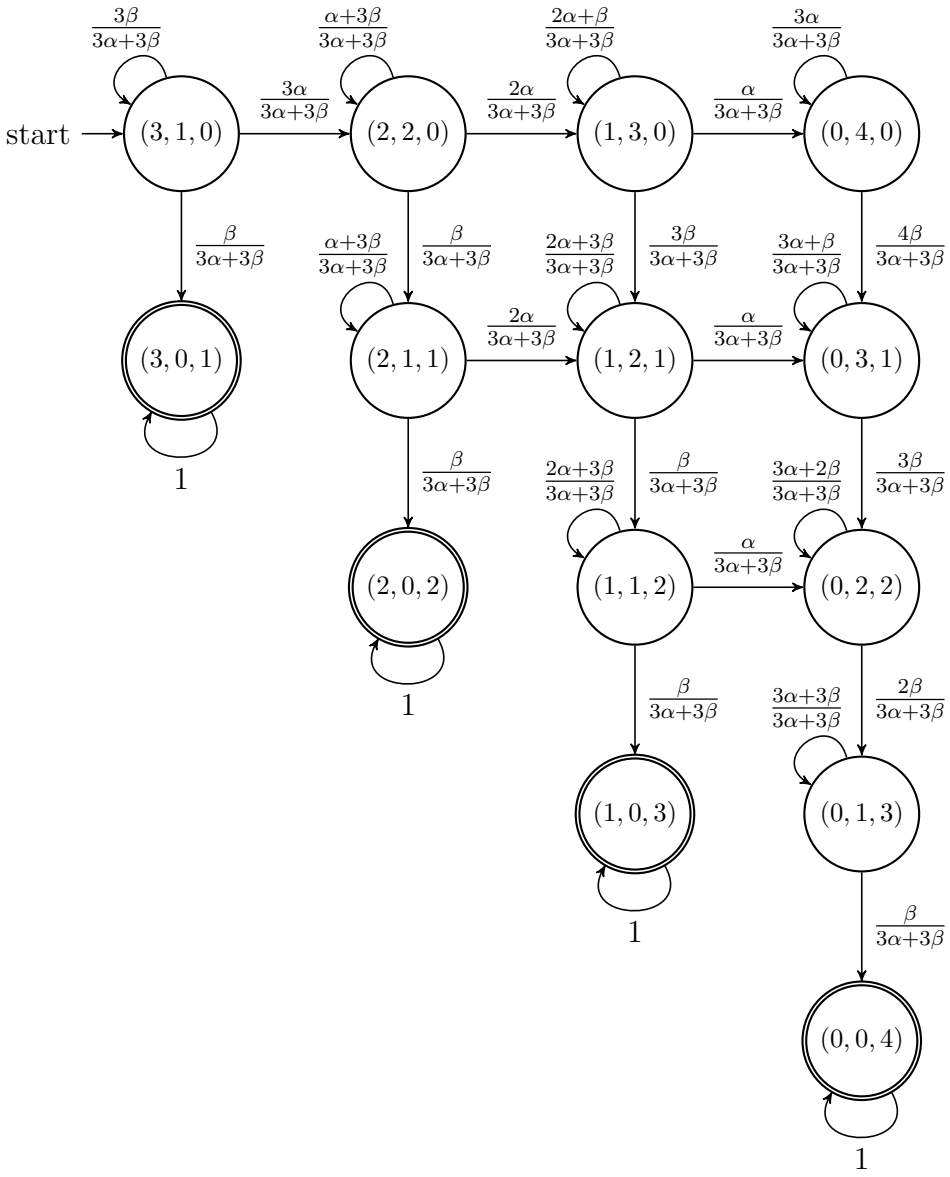


Figure 5.9: $SIR(3, 1, 0)$ Uniformized DTMC model with parameters(α, β) and uniformization rate $(3\alpha + 4\beta)$

As uniformization of a CTMC preserves bounded until properties, we conduct parameter synthesis experiments on uniformized DTMC. We check the following property: *"With the probability of at least 25 percent, the number of infected individuals does not exceed half of the population until the system is in its steady-state."* Let

$N = S_0 + I_0 + R_0$. In the PCTL formula we have:

$$P_{\geq 0.25}(!(i > N/2) \quad \mathsf{U}^{\leq N} \quad (i = 0))$$

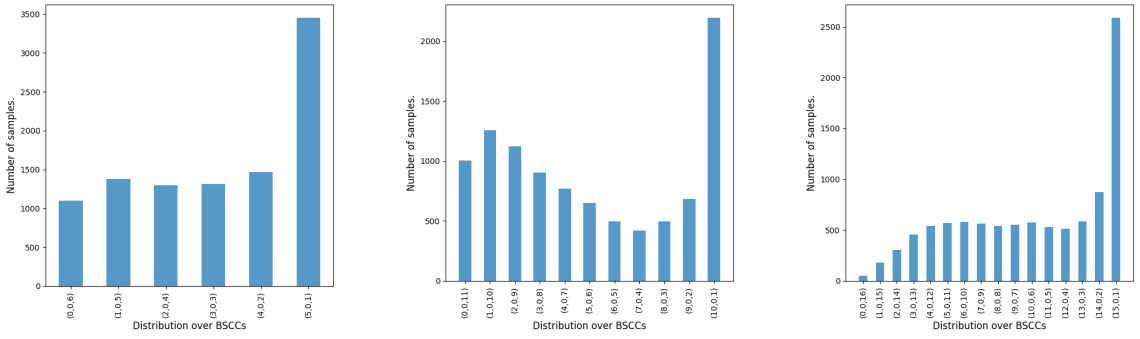
5.3.3 Evaluation

True parameters and synthetic data

We select true parameters are selected arbitrarily random to test the frameworks.

Model $\mathcal{M}$	SIR(5,1,0)	SIR(10,1,0)	SIR(15,1,0)
Number of BSCCs	6	11	16
Number of states	27	77	152
True parameter (α, β)	(0.034055, 0.087735)	(0.025490, 0.069298)	(0.011499, 0.062111)
Synthetic data D_{obs}	(1098, 1377, 1296, 1312, 1466, 3451)	(1002, 1258, 1123, 902, 770, 651, 497, 420, 496, 685, 2196)	(50, 181, 302, 455, 539, 567, 582, 566, 541, 553, 574, 528, 512, 586, 875, 2589)
Property of interest Φ	$P_{\geq 0.25}(i \leq 3 \ \mathsf{U}^{\leq 6} \ i = 0)$	$P_{\geq 0.25}(i \leq 5 \ \mathsf{U}^{\leq 11} \ i = 0)$	$P_{\geq 0.25}(i \leq 8 \ \mathsf{U}^{\leq 16} \ i = 0)$
$P(\mathcal{M}_{(\alpha,\beta)} \models \Phi)$	0.3474444	0.265815	0.327446

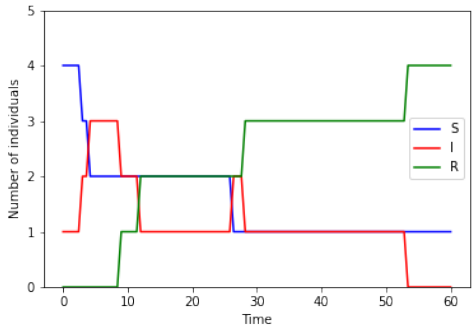
Table 5.13: Synthetic data for SIR(5,1,0), SIR(10,1,0), SIR(15,1,0)



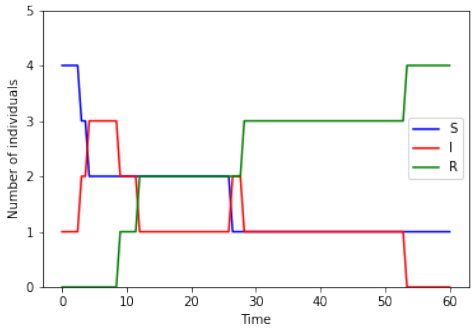
(a) SIR(5,1,0) (b) SIR(10,1,0) (c) SIR(15,1,0)

Figure 5.10: Histograms of synthetic data D_{obs}.

We show examples of examples and counter-examples with simulations of CTMC. Using Gillespie algorithm [17] to simulate CTMC of selected (α, β), we show (i) example of the property *"the number of infected individuals does not exceed half of the population until the system is in its steady-state"* being satisfied, and (ii) its corresponding counter-example.

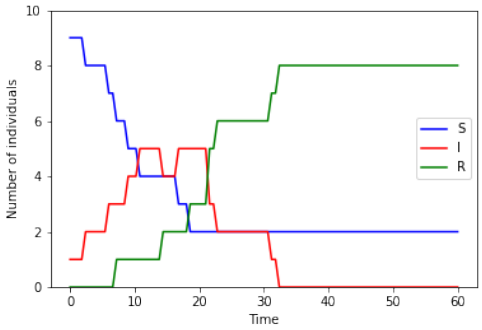


(a) Example with SIR(5,1,0)

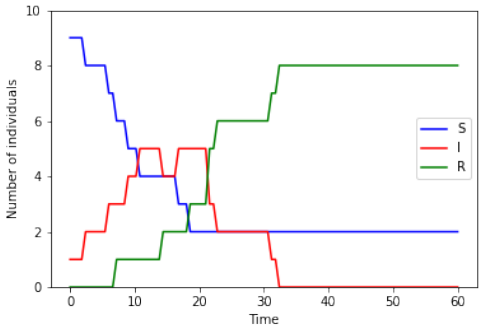


(b) Counter-example with SIR(5,1,0)

Figure 5.11: Example and counter-example on SIR(5,1,0) CTMC with $(\alpha, \beta) = (0.034055, 0.087735)$.

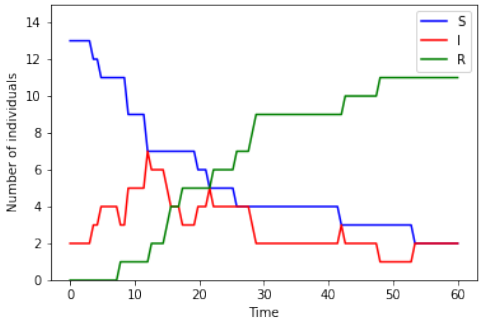


(a) Example with SIR(10,1,0)

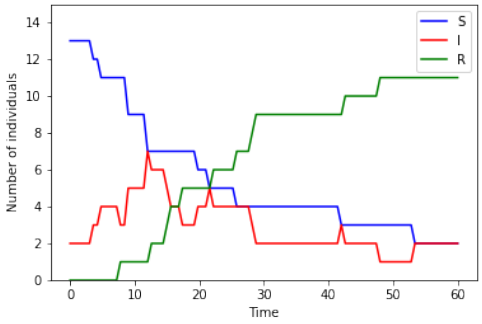


(b) Counter-example with SIR(10,1,0)

Figure 5.12: Example and counter-example on SIR(10,1,0) CTMC with $(\alpha, \beta) = (0.025490, 0.069298)$.



(a) Example with SIR(15,1,0)



(b) Counter-example with SIR(15,1,0)

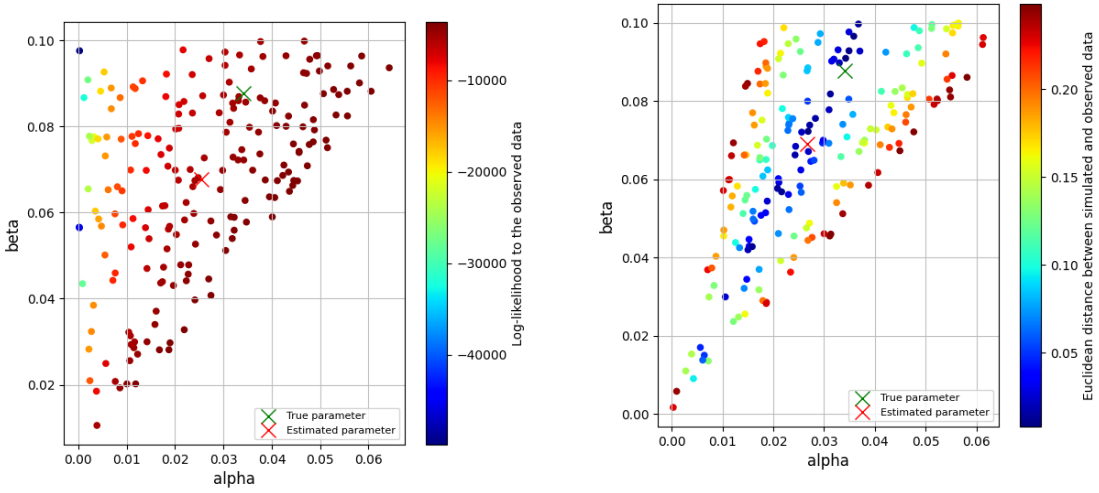
Figure 5.13: Example and counter-example on SIR(15,1,0) CTMC with $(\alpha, \beta) = (0.011499, 0.062111)$.

Parameter synthesis result

Parameter synthesis results for SIR(5,1,0) model:

Method	RF-SMC	SMC-ABC-SMC
Estimated parameter $\hat{\theta}$	(0.034055, 0.087734)	(0.034055, 0.087734)
True parameter θ	(0.025473, 0.067613)	(0.023077, 0.064812)
L2 distance $\|\theta, \hat{\theta}\|_2$	0.021875	0.020120
$P(\mathcal{M}_{\hat{\theta}} \models \Phi)$	0.352182	0.347407

Table 5.14: Parameter estimation results for SIR(5,1,0) model.



(a) Sampled particles using RF-SMC

(b) Sampled particles using SMC-ABC-SMC

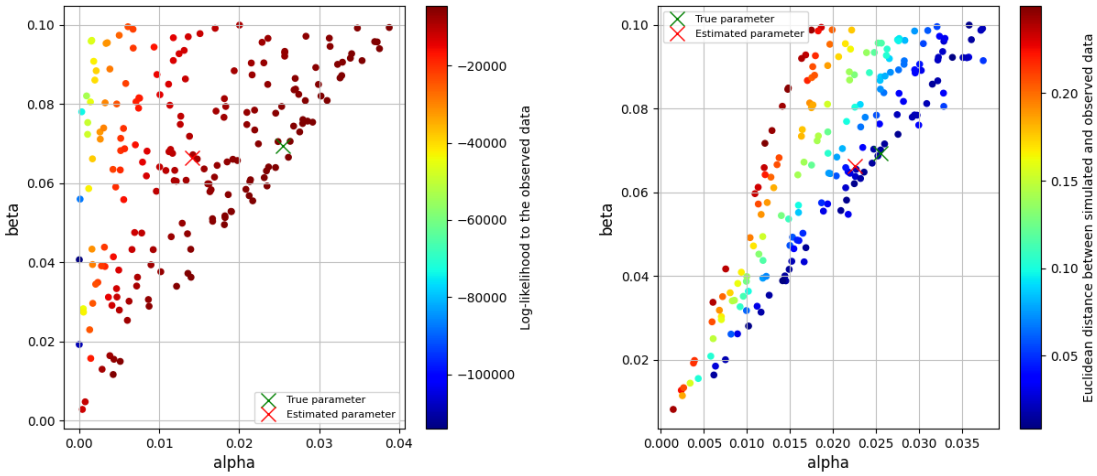
Figure 5.14: SIR(5,1,0) parameter synthesis results.

In the experiment with $SIR(5, 1, 0)$ model we observe that the rational-function-based method and simulation-based method deliverresults without any significant difference.

Parameter synthesis results for SIR(10,1,0) model:

Method	RF-SMC	SMC-ABC-SMC
Estimated parameter $\hat{\theta}$	(0.014095, 0.066328)	(0.022552, 0.066416)
True parameter θ	(0.025490, 0.06930)	(0.025490, 0.06930)
L2 distance $\|\theta, \hat{\theta}\|_2$	0.011776	0.004116
$P(\mathcal{M}_{\hat{\theta}} \models \Phi)$	0.363570	0.281154

Table 5.15: Parameter estimation results for SIR(10,1,0) model.



(a) SIR(10,1,0), sampled particles using RF-SMC

(b) SIR(10,1,0), sampled particles using SMC-ABC-SMC

Figure 5.15: parameter synthesis results.

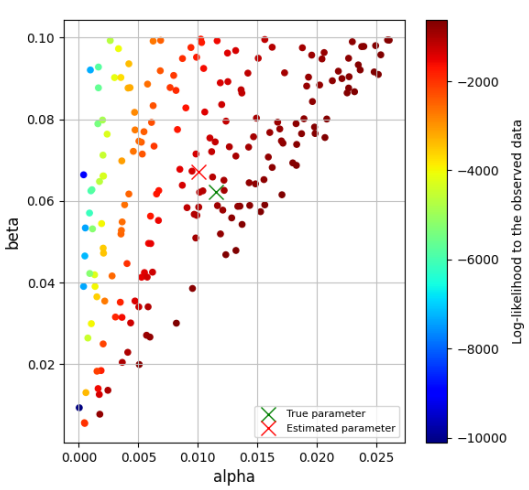
In the experiment with $SIR(10, 1, 0)$ model we can see that the simulation-based method delivers a closer estimation to true parameter. However, rational-function-based method gives an estimation with higher satisfaction probability.

Parameter synthesis results for SIR(15,1,0) model:

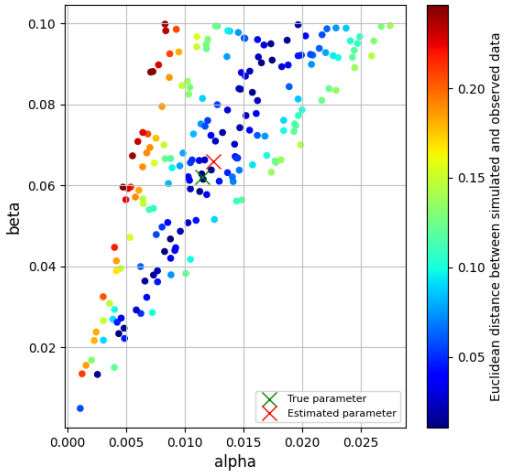
Method	RF-SMC	SMC-ABC-SMC
Estimated parameter $\hat{\theta}$	(0.012444, 0.065862)	(0.010022, 0.067230)
True parameter θ	(0.011499, 0.062111)	(0.011499, 0.062111)
L2 distance $\|\theta, \hat{\theta}\|_2$	0.006545	0.005520
$P(\mathcal{M}_{\hat{\theta}} \models \Phi)$	0.366493	0.323579

Table 5.16: Parameter estimation results for SIR(15,1,0) model.

In the experiment with $SIR(10, 1, 0)$ model we can see that the simulation-based method delivers a slightly closer estimation to true parameter. Rational-function-based method gives an estimation with higher satisfaction probability. However, overall, the estimations are comparable in terms of accuracy an satisfaction probability



(a) Sampled particles using RF-SMC



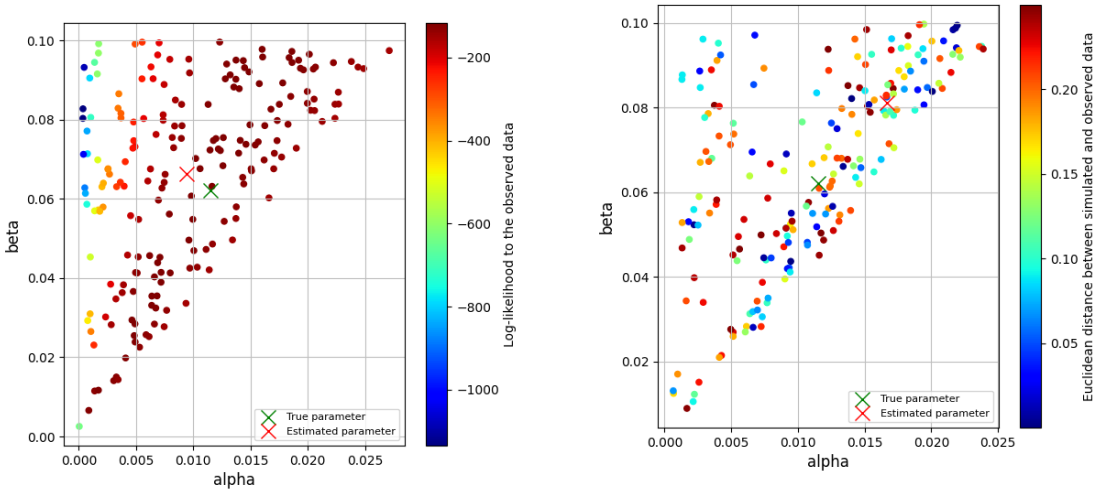
(b) Sampled particles using SMC-ABC-SMC

Figure 5.16: SIR(15,1,0) parameter synthesis results.

We study the behavior of both frameworks when less data is available. Instead of using synthetic data of 10000 samples, we use 200 samples as observed data for parameter synthesis and estimation. We also introduce uncertainty by merging BSCCs into 2 bins: one consists of BSCCs with more than half of the population remain uninfected, and the other consists of all BSCCs left. This adjustment to the DTMC is made by merging BSCCs into a single BSCC.
We perform the experiment on SIR(15,1,0) with only two BSCCs: (i) BSCCs with $S \leq 8$, and (ii) BSCC with $S > 8$. The results are then summarized into the following table:

Method	RF-SMC	SMC-ABC-SMC
Estimated parameter $\hat{\theta}$	(0.00945054, 0.06634182)	(0.016698, 0.081153)
True parameter θ	(0.011499, 0.062111)	(0.011499, 0.062111)
L2 distance $\|\theta, \hat{\theta}\|_2$	0.004701	0.019740
Satisfaction property $P(\mathcal{M}_{\hat{\theta}} \models \Phi)$	0.374375	0.306351

Table 5.17: Parameter estimation results for SIR(15,1,0) model with merged BSCC.



(a) Sampled particles using RF-SMC

(b) Sampled particles using SMC-ABC-SMC

Figure 5.17: Parameter synthesis results for SIR(15,1,0) model with merged BSCC.

Unlike experiments with SIR(15,1,0) without merging BSCCs, the experiment with merged BSCCs and simulation-based framework shows no general pattern of distance among satisfying parameter values. It is hypothetically because of the noisy data that leads to a non-converging evaluation of distance between simulated data and observed data [46]. A proposed way to solve the problem is to use a decreasing threshold scheme, in which ϵ decreases after each perturbation. We experiment again with the distance threshold decreases by a factor 0.95 after each perturbation round. The result is then visualized:

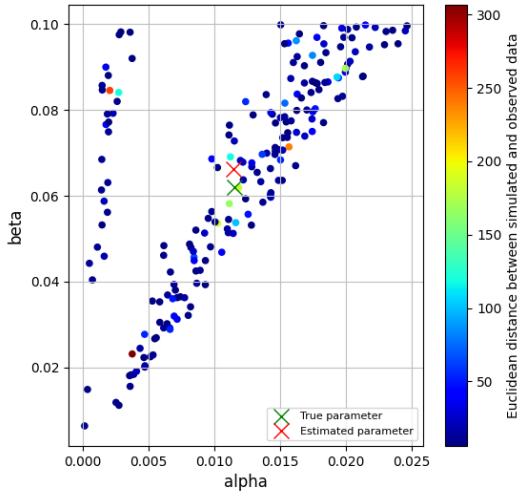


Figure 5.18: Parameter synthesis results for SIR(15,1,0) model with merged BSCC and decreasing distance threshold.

The estimated parameter is $(0.01148671, 0.06624936)$, closer to the true parameter (Euclidean distance is 0.004138), and the satisfaction probability $P(\mathcal{M}_{\alpha,\beta}) = 0.336849$.

5.3.4 Performance

To measure the performance of RF-SMC and SMC-ABC-SMC, we measure the physical runtime to finish *(total runtime)* and the average physical run time of one perturbation kernel *(average perturbation runtime)*.

Method	RF-SMC	SMC-ABC-SMC
Total runtime (minutes)	5.917	68.783
Average perturbation runtime (minutes)	0.312	3.614

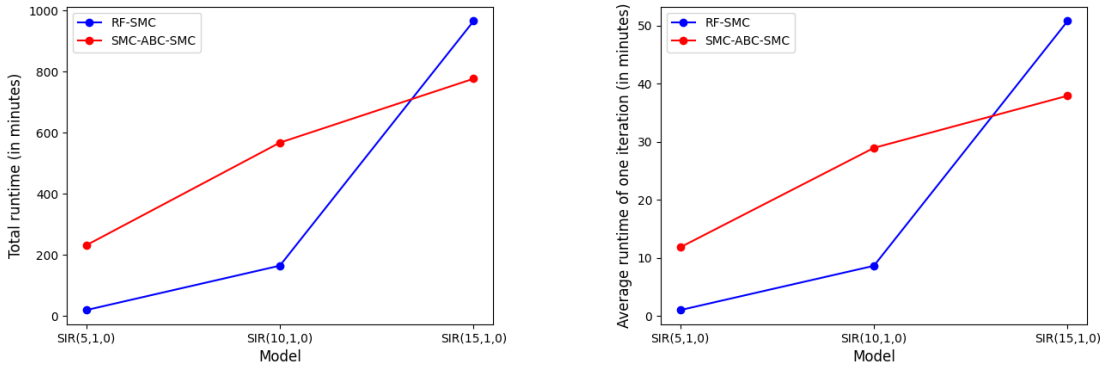
Table 5.18: Physical runtime on SIR(5,1,0) model.

Method	RF-SMC	SMC-ABC-SMC
Total runtime (minutes)	29.517	352.083
Average perturbation runtime (minutes)	1.553	18.518

Table 5.19: Physical runtime on SIR(10,1,0) model.

Method	RF-SMC	SMC-ABC-SMC
Total runtime (minutes)	3976.117	581.833
Average perturbation runtime (minutes)	209.237	30.592

Table 5.20: Physical runtime on SIR(15,1,0) model.



(a) Total runtime

(b) Average perturbation runtime

Figure 5.19: Physical runtime on SIR models of different population sizes.

5.3.5 Discussion

As the performance statistics show, as the state-space of a parametric DTMC grows, the rational-function-based method becomes more expensive in computational cost than the simulation-based method. However, as the last experiment shows, Approximate Bayesian Computation does not accurately approximate the posterior when less data is provided. However, we also shown that combining with a decreasing distance threshold to approximate the likelihood more precisely, SMC-ABC-SMC shows a comparable accuracy to rational-function-based RF-SMC. Therefore, experiments with SIR models suggest that the simulation-based method is superior to rational-function-based method.

Chapter 6

Conclusion

We presented frameworks to perform data-informed parameter synthesis. The frameworks are tested against different case studies and show that they can deliver both a set of satisfying parameter values and an estimated parameter value close to the original value used to synthesize test data. Therefore, these frameworks are applicable when we need to estimate the unknown attributes and proactively verify the system against the property of interest.
In the case studies, the comparisons between rational-function-based RF-SMC and simulation-based SMC-ABC-SMC show that the computational cost of SMC-ABC-SMC increases slower than that of RF-SMC, while the two methods are comparable in terms of accuracy for parameter estimation. Therefore, we suggest that given an arbitrary parametric DTMC, simulation-based method SMC-ABC-SMC is preferable over the rational-function-based RF-SMC.
There are possible extensions to the presented frameworks, including but not limited to:

- *Statistical Model Checking*: we can use Absolute-Error Massart Bounds (proposed by Molyneux [39], but currently not supported by PRISM) on Statistical Model Checking to achieve a better bound on the number of simulations.
- *Bayesian Model Checking*: Jha [27], and Zuliani [51] presents a novel method that improves Statistical Model Checking by using Bayesian inference. By incorporating prior knowledge about the model being verified, the method requires less simulation runs than the frequentist statistical model checking, thus reducing the computational cost.
- *Sampling algorithms*: different sampling algorithms can be used to estimate posterior distribution. For example, PyMC3 library [44] uses No-U-Turn Sam-

pling (Hoffman [25]) algorithm by default, as it exploits the gradient of the likelihood to approximate the posterior distribution with less iterations.

- *Implementation improvement*: Currently, StormPy technically prohibits our implementation from being parallelized since StormPy's core classes are not serializable [1]. Porting to the C++ language has several benefits by achieving the higher performance of C++ and exploiting the data-parallelism of Sequential Monte Carlo algorithm.

[1] `https://github.com/moves-rwth/stormpy/issues/36`

Bibliography

[1] Gul Agha and Karl Palmskog. "A survey of statistical model checking". In: *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 28.1 (2018), pp. 1–39.

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

[3] Christel Baier et al. "Simulation for continuous-time Markov chains". In: *International Conference on Concurrency Theory*. Springer. 2002, pp. 338–354.

[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] Katalin Csilléry et al. "Approximate Bayesian computation (ABC) in practice". In: *Trends in ecology & evolution* 25.7 (2010), pp. 410–418.

[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] Leonardo De Moura and Nikolaj Bjørner. "Z3: An efficient SMT solver". In: *International conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer. 2008, pp. 337–340.

[11] 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.

[12] 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.

[13] *Dynamic Configuration of IPv4 Link-Local Addresses*. `https://tools.ietf.org/html/rfc3927`. Accessed: 2021-03-30.

[14] Bradley Efron. “Bootstrap methods: another look at the jackknife”. In: *Breakthroughs in statistics*. Springer, 1992, pp. 569–593.

[15] Sarah Filippi et al. “On optimality of kernels for approximate Bayesian computation using sequential Monte Carlo”. In: *Statistical applications in genetics and molecular biology* 12.1 (2013), pp. 87–107.

[16] Stuart Geman and Donald Geman. “Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images”. In: *IEEE Transactions on pattern analysis and machine intelligence* 6 (1984), pp. 721–741.

[17] Daniel T Gillespie. “A general method for numerically simulating the stochastic time evolution of coupled chemical reactions”. In: *Journal of computational physics* 22.4 (1976), pp. 403–434.

[18] 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.

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

[20] Hans Hansson and Bengt Jonsson. “A logic for reasoning about time and reliability”. In: *Formal aspects of computing* 6.5 (1994), pp. 512–535.

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

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

[23] Thomas Hérault et al. “Approximate probabilistic model checking”. In: *International Workshop on Verification, Model Checking, and Abstract Interpretation*. Springer. 2004, pp. 73–84.

[24] 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.

[25] Matthew D Hoffman and Andrew Gelman. “The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo.” In: *J. Mach. Learn. Res.* 15.1 (2014), pp. 1593–1623.

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

[27] 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.

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

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

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

[31] 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.

[32] Joost-Pieter Katoen et al. “Faster and symbolic CTMC model checking”. In: *Joint International Workshop von Process Algebra and Probabilistic Methods, Performance Modeling and Verification*. Springer. 2001, pp. 23–38.

[33] William Ogilvy Kermack and Anderson G McKendrick. “A contribution to the mathematical theory of epidemics”. In: *Proceedings of the royal society of london. Series A, Containing papers of a mathematical and physical character* 115.772 (1927), pp. 700–721.

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

[35] Donald E Knuth, KNUTH DE, and YAO AC. “THE COMPLEXITY OF NONUNIFORM RANDOM NUMBER GENERATION.” In: (1976).

[36] 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.

[37] 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.

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

[39] 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.

[40] 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.

[41] Vincent Plagnol and Simon Tavaré. "Approximate Bayesian computation and MCMC". In: *Monte Carlo and Quasi-Monte Carlo Methods 2002*. Springer, 2004, pp. 99–113.

[42] 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.

[43] Mojtaba Sadegh and Jasper A Vrugt. "Approximate bayesian computation using Markov chain Monte Carlo simulation: DREAM (ABC)". In: *Water Resources Research* 50.8 (2014), pp. 6767–6787.

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

[45] 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).

[46] Scott A Sisson, Yanan Fan, and Mark M Tanaka. "Sequential monte carlo without likelihoods". In: *Proceedings of the National Academy of Sciences* 104.6 (2007), pp. 1760–1765.

[47] 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.

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

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

[50] 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.

[51] 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.