

Bayesian parameter synthesis of Markov population models

Master Project on Modeling of Complex, Self-organizing
systems

Huy Phung
University of Konstanz

July 22, 2020

Contents

1	Preliminaries	4
1.1	Discrete time Markov Chain	4
1.1.1	Definition	4
1.1.2	Parametric Discrete-time Markov Chain	5
1.1.3	Markov population model	6
1.2	Bayesian inference	6
1.2.1	Bayesian parameter estimation	6
1.2.2	Posterior conjugation	7
1.2.3	Metropolis-Hastings algorithm	9
1.2.4	Selection of prior distribution	10
1.2.5	Bayesian Parameter Estimation	10
1.2.6	Bayesian Credible Set	11
2	Related works	11
3	Modeling of bees colony	12
3.1	Biological description	12
3.2	Markov population models for bee colony	12
3.2.1	Asynchronous model	13
3.2.2	Synchronous model	14
3.2.3	Semi-synchronous model	15
3.2.4	Multiparameters model	16
3.2.5	Linear model	17
4	Bayesian parameter synthesis of Markov population model	18
4.1	Inference of terminal state distribution	18
4.2	Evaluation of terminal states distribution	18
4.3	Inference of model parameter with Metropolis-Hastings algorithm	19
5	Benchmark	21
5.1	Data synthesis	21
5.2	Evaluation measures	22
5.3	Inference of terminal state distribution	22
5.3.1	Experiment setup	22
5.3.2	Results	23

5.3.3	Conclusion	25
5.4	Compare DTMC sampling and rational functions	25
5.4.1	Experiment setup:	25
5.4.2	Results	26
5.4.3	Conclusion	28
5.5	Inference of model parameters	29
5.5.1	Experiment setup	29
5.5.2	Results	29
5.5.3	Conclusion	29
5.6	Selection of Metropolis-Hastings chain length	29
5.6.1	Experiment setup	29
5.6.2	Result	30
5.6.3	Conclusion	31
5.7	Selection of prior distribution	31
5.7.1	Experiment setup	31
5.7.2	Result	31
5.7.3	Conclusion	31
5.8	Impact of data quality	31
5.8.1	Experiment setup	32
5.8.2	Result	32
5.8.3	Conclusion	32
5.9	Linear model	32
5.9.1	Experiment setup	32
5.9.2	Results	33
5.9.3	Conclusion	35
6	Implementation	35
6.1	Class design	35
6.2	Test environment	36
6.3	Directory structure	36
7	Conclusion	37

Abstract

We study the collective behavior of a bee colony. Each bee in a colony possibly stings after observing a threat in the surrounding environment, and warn other bees by releasing a special substance, pheromone. By sensing the pheromone released in the environment, other bees in the colony may also sting. However, since stinging leads to the termination of an individual bee, it reduces the total defense capability as well. With parametric Discrete-time Markov chain as the model, we study how the actions of a single bee change with regarding to the colony size of and pheromone amount. This project propose and discuss Bayesian methods to estimate parameters of bees population models. It shows that the proposed methods is scalable and can deliver estimations of model parameters.

1 Preliminaries

1.1 Discrete time Markov Chain

Assume that each bee in a colony decides its next action (to sting or not to sting) based only on the current state of the environment, and the number of bees who sting or not sting can be modeled as a Markov process. To reduce the complexity of the model, we make another assumption that the states of the bees colony are observed after uniform time duration, hence the model is of discrete-time.

1.1.1 Definition

Definition 1.1 (Discrete Time Markov Chain). A Discrete Time Markov Chain (DTMC) is a tuple $(S, \mathbf{P}, S_{init}, AP, L)$ [2]

- S is a countable non-empty set of *states*
- $\mathbf{P} : S \times S \rightarrow [0, 1]$ is the *transition probability* function, s.t

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

- $S_{init} : S \rightarrow [0, 1]$ is the *initial distribution*, s.t

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

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

The transition probability function in DTMC is given as a stochastic matrix \mathbf{P} , which satisfies the following properties

- \mathbf{P} is a square matrix.
- $\mathbf{P}_{i,j} \in [0, 1]$.
- $\sum_j \mathbf{P}_{i,j} = 1$

Definition 1.2 (Strongly Connected Component). Let $\mathcal{M} = (S, \mathbf{P}, S_{init}, AP, L)$ a DTMC. A subset $S' \subset S$ is strongly connected if and only if for every pair $s_1, s_2 \in S'$ there is a path between s_1 and s_2 which consists of only of state in S' . If there exist no $S'' \subseteq S$, such that $S \subset S''$ and S'' is strongly connected, then S' is a *Strongly Connected Component*, or *SCC* in short.

Definition 1.3 (Bottom Strongly Connected Component). Let $\mathcal{M} = (S, \mathbf{P}, S_{init}, AP, L)$ a DTMC and $S' \in S$ a Strongly Connected Component. S' is also a *Bottom Strongly Connected Component*, or *BSCC* for short, if and only if there exist no state $s \in S$ S' that is reachable from any state in S' .

1.1.2 Parametric Discrete-time Markov Chain

In order to generalize the stochastic matrix to encompass unknown information of the system, we introduce *parametric DTMC*. Let $\theta = (\theta_1, \dots, \theta_n) \in [0, 1]^n$, we represent each element in matrix \mathbf{P} as a polynomial function of θ . Let \mathbf{Pol}_θ be the set of polynomials $P : [0, 1]^n \rightarrow [0, 1]$. We define *parametric Discrete-Time Markov Chain*, or *pMC* for short, as follow:

Definition 1.4 (Parametric DTMC). A parametric Discrete Time Markov Chain (pMC for short) is a tuple $(S, \theta, \mathbf{P}_\theta, S_{init}, AP, L)$

- S is a countable nonempty set of *states*
- θ is the set of model parameters
- $\mathbf{P} : S \times S \rightarrow \mathbf{Pol}_\theta$ is the *transition probability* function that map a transition relation between two states to a polynomial function of θ

- $S_{init} : S \rightarrow [0, 1]$ is the *initial distribution*, s.t

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

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

A concrete assignment of θ on pMC induces a DTMC. In this project, we concern about the problem of data-informed estimation of model parameter θ .

1.1.3 Markov population model

In the scope of this project we concern *population* of a bee colony with N individuals initially. We model the population of a bee colony using a pMC $\mathcal{M} = (S, \theta, \mathbf{P}_\theta, S_{init}, AP, L)$ with the following properties:

- Each state $s \in S$ represent number of individuals in the bee colony.
- There exists a set of terminal states $T \subset S$, such that t is a BSCC for all $t \in T$, and $|T| = N + 1$.

Assume we conduct a real biological experiment on a colony of N bees. At the end of the observation, we observes N' bees ($0 \leq N' \leq N$) bees. Let $tSCC_i \in T$ denotes the terminal states with i bees left. As we conduct the experiment with the same colony size multiple times, we get a probability distribution over final states T .

1.2 Bayesian inference

1.2.1 Bayesian parameter estimation

Let D be observed data. In statistical inference, we assume that the observed data has a probability distribution of unknown parameter θ , i.e $D \sim P(D|\theta)$. In frequentist approach, the estimation of θ based on long-run property, that is, given a large enough sample size, expected value of parameter estimation $\hat{\theta}$ is equal to θ . Therefore, frequentist approach requires to gather a large

amount of data to deliver a close estimation $\hat{\theta}$. In Bayesian approach, we reuse the information *beliefs* gained from observed data to enhance the accuracy of the estimation of $\hat{\theta}$. The main advantage of Bayesian approach over frequentist approach is that it require less data to obtain an estimation $\hat{\theta}$. The beliefs obtained from prior knowledge of model parameter θ is represented by *prior distribution* $\pi(\theta)$.

Also, we have probability distribution of observed data, given parameter θ , $P(D|\theta)$. This is also called *likelihood function*.

With Bayesian formula, we have

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

$\int_{\theta} P(D|\theta)\pi(\theta)d\theta$ is called *marginal distribution*. $\pi(\theta|D)$ is called *posterior distribution*. Computing posterior distribution is the essential part of Bayesian inference, since it gives us the estimation of parameter θ .

1.2.2 Posterior conjugation

Conjugated posteriors are special cases of Bayesian inference, in which the prior and posterior distribution belongs to the same family of distribution. Conjugated posteriors give us significant benefits

1. Tractability: we have analytical form of posterior distribution.
2. Computationally effective: updating model parameter is of linear time to the dimension of parameter.

We consider two conjugated posterior: Binomial-Beta and Dirichlet-Multinomial

Lemma 1 (Binomial-Beta Conjugation). Binomial distribution is conjugated to beta distribution.

Proof. The observed data $D = (x_1, \dots, x_n)$ is sampled from *Binomial*(k, θ) 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*(α, β) 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 $Beta(\alpha + \sum_{i=1}^n x_i, \beta + nk - \sum_{i=1}^n x_i)$ \square

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

Lemma 2 (Multinomial-Dirichlet Conjugation). Multinomial distribution is conjugated to Dirichlet distribution.

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

$$P(x_1, \dots, x_n | N, \theta_1, \dots, \theta_n) = \frac{n!}{x_1! \dots x_n!} \prod_{i=1}^n \theta_i^{x_i}$$

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

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

We obtain

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

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

More detailed description in these cases can be found in [13] and [3]. We summarize the necessary results in the following table:

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

1.2.3 Metropolis-Hastings algorithm

In case the posterior distribution has no analytical form or its analytical form is difficult to sample from directly, we use *Metropolis-Hastings* algorithm (*MH* in short).

Metropolis-Hastings algorithm is a *Monte Carlo Markov Chain* algorithm. In its essential, Metropolis-Hastings algorithm draws Using the MH algorithm, we can estimate the parameter by posterior mean, without knowing the analytical form of posterior distribution itself.

Algorithm 1 Metropolis-Hastings Algorithm

Input: D is the observation data,
Output: $Trace$ is the set of accepted sampling point.

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

The likelihood function can be implemented as log-likelihood to avoid underflow error. Proposal distribution defines how do we proceed to the next parameter value on the parameter space; it can be of any distribution family. There are two advantages of using Markov Chain Monte Carlo in Bayesian

inference:

1. Parameter transition only needs the computation of likelihood function. Therefore, Monte Carlo Markov Chain can be used in general Bayesian inference, in which we are not guaranteed to have an analytical form of posterior.
2. Specifically in Metropolis-Hastings algorithm, marginal distribution is cancelled out, thus make Metropolis-Hastings a computationally efficient algorithm.

However, MH algorithm also has a drawback; its convergence becomes slower as the dimension of parameter θ increases.

1.2.4 Selection of prior distribution

Theoretically, prior can be of any distribution family. However, a selection of prior distribution that is too different than the actual distribution of parameter can lead to a false propagation of beliefs and degrade inference results. It is suggested by [11] that in case of no prior knowledge exists to help the selection of prior distribution, Uniform distribution is preferable since it is less likely to propagate false beliefs to the inference.

A systematic inference to select prior distribution family and prior distribution parameter (hyperparameters) is possible with *Hierarchical Bayes Models* [1].

1.2.5 Bayesian Parameter Estimation

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

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

In case we have samples from posterior distribution, for example the *Trace* from Metropolis-Hastings algorithm, for example when we use MH algorithm, the discrete form of posterior mean is used:

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

1.2.6 Bayesian Credible Set

Definition 1.5 (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 $(1-\alpha)$.

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

In this project, we use 0.95 credible set, i.e $\alpha = 0.05$

Definition 1.6 (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 project, the HPD is calculated using algorithm from *PyMC3* library [12]. For simplicity, we assume that in all cases which we concern, HPD is computed for unimodal distribution.

Algorithm 2 Compute Highest Posterior Density Interval

Input: S is samples from a distribution.

Input: $0 \leq \alpha \leq 1$

Output: HPD interval

- 1: **procedure** COMPUTE HPD(S)
 - 2: Compute interval width $w = |S| * \alpha$
 - 3: Find modal (peak) of sample points.
 - 4: Return minimal interval of size $|S| - w$ which contains the modal.
 - 5: **end procedure**
-

2 Related works

The definition and model checking of DTMC and pMC is studied by [2], [6], and [8].

Bayesian inference of pMC parameters is studied in [11] and [7]. In [11], the authors developed methods to synthesize parameters to satisfy a given set of PCTL properties. In [7], the authors presented methods to perform model checking of biological system using Bayesian statistic. The authors in [7] uses a Bayesian hypothesis test, where H_0 is the null hypothesis that the model

satisfies a PCTL P , and alternative hypothesis H_1 is that the system does not satisfies P .

In this project, we use bee colony model semantics from [5]. The methods and implementation in this project is designed to extend the results of [5] and its tool *DiPS*.

3 Modeling of bees colony

3.1 Biological description

We consider the collective action of a bee colony. Each bee in a colony could possibly sting after observing a threat in the surrounding environment, and warn other bees by releasing pheromone. By sensing the pheromone released in the environment, other bees in the colony may also sting. Since stinging leads to the termination of an individual bee, it reduces the total defense capability as well. We studies how the actions of a bee changes with regarding to its surrounding the environment. There are 3 assumptions on the system:

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

Under these assumption, a bee colony can be viewed as a set of agents (bees) interact with each other in a closed environment with the appearance of a factor *pheromone*. Each agent in the colony observe the following two factors of the environment: (1) amount of pheromone and (2) number of other agents. Afterward, the agent has probability to commit an action, namely *sting*. The agent is eliminated from environment after stinging.

3.2 Markov population models for bee colony

Assume that we have a colony of n bees initially. As aforementioned, an individual bee is terminated after it stings. Thus, at the end of experiment,

the number of bees is $n' \in \{0, 1, \dots, n\}$. We model the bee colony with a DTMC $\mathcal{M} = (S, \mathbf{P}, S_{init}, AP, L)$, such that

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

Semantics of Markov population models for bees colony are developed by [5]. In this report we summarize three models of bees population

1. Synchronous model
2. Asynchronous model
3. Semi-synchronous model

3.2.1 Asynchronous model

In asynchronous experiment we assume that there is almost improbable for two bees to sting simultaneously, and any bees release pheromone immediately after its death. By that observation we can assume that each bee sting at different level of pheromone.

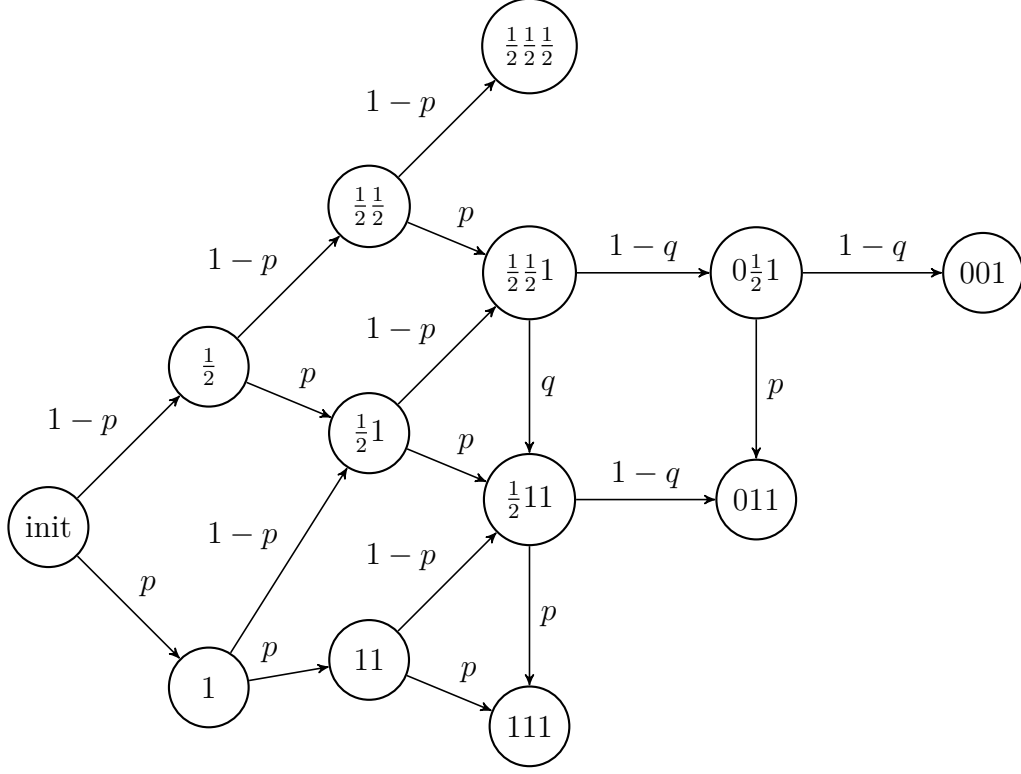


Figure 1: Example asynchronous model of 3 bees, 2 parameters

3.2.2 Synchronous model

In *fully synchronous* experiment we assume that the number of stinging bees is only counted after a fixed amount of time, so that without loss of generality we can assume the pheromone diffuse almost immediately among the bee colony, and each bee decide to sting or not to sting immediately after sensing the pheromone concentration.



Figure 2: Synchronous model of 3 bees, 2 parameters.

3.2.3 Semi-synchronous model

In *Semisynchronous model*, we assume that the behaviour is initially synchronous. That means, at the initial states we do synchronous update. From all succeeding states from initial states, the updates are of asynchronous semantics.



Figure 3: Semisynchronous model of 3 bees, 2 parameters

For a population of n bees, all three types of models share two properties:

1. Has exactly one initial state, $|S_{init}| = 1$. This assumption is obvious, since at the beginning of the experiment, all bees are alive.
2. Has $n + 1$ tSCCs ($tSCC_0, \dots, tSCC_n$). This is because the number of dead bees cannot exceed the population size. It also follows that

$$\sum_{i=0}^n P(FG \ tSCC_i) = 1$$

3.2.4 Multiparameters model

Multiparameters model of N bees has exactly N parameters r_0, \dots, r_{N-1} . In the semantics of multiparameter models, we assume that r_0, \dots, r_{N-1} is monotonically increasing, that is, $\forall i, j \in 0, \dots, N-1 : i < j \implies r_i \leq r_j$.



Linear model is similar to multiparameter models, except:

- $$r_i = a \times i + b$$

17

i The purpose of imposing linear constraint to model parameters is to reduce the number of model parameters in total. In a linear model, the number of model parameters are always 2.

4 Bayesian parameter synthesis of Markov population model

4.1 Inference of terminal state distribution

Using the Multinomial-Dirichlet conjugation, we can infer the terminal steady state distribution as we observe new experiment data $D = (a_0, \dots, a_n)$, with a_i is the number of experiment in which the population at terminal state is i . Let $p_i = P(tSCC_i)$ be the terminal state distribution.

Algorithm 3 Estimation of terminal state distribution given a sample S

```

1: procedure ESTIMATE  $p_i = P(tSCC_i)(S)$ 
2:   Initialize  $\alpha_0 = \alpha_1 = \dots = \alpha_n = 1$ 
3:   Initialize  $p_0 = p_1 = \dots = p_n = 0$ 
4:   Update  $\alpha_i = \alpha_i + a_i, 1 \leq i \leq n$ 
5:   Update  $p_i = \alpha_i / \sum_{i=1}^n \alpha_i, 1 \leq i \leq n$ 
6: end procedure

```

As later shown in the implementation with synthetic data, the Bayesian estimation of terminal state distribution converges quickly to the true parameter used for data synthesis. This inference does not directly estimate the pMC model parameters. However, it can support the parameter estimation presented [5] to estimate the model parameters with narrower credible set. Furthermore, as the inference helps to estimate terminal state distribution with less data available.

4.2 Evaluation of terminal states distribution

Given a concrete assignment of model parameter r_0, \dots, r_{N-1} , there are 2 methods that are used in the project to evaluate terminal state distribution:

1. Rational functions.

2. DTMC sampling.

Rational functions are functions of model parameter that represent the probability of finally globally reach each terminal state. The function is delivered by PRISM model checker thanks to its capability of symbolic model checking [9].

However, it is not always possible to deduct rational functions from a given model, due to the technical limitation (time, memory) and the limitation of PRISM itself. In our conducted experiment, PRISM is capable of deliver rational functions up to a population of 15 bees. For a population of more bees, we use the second approach, *DTMC sampling*.

DTMC sampling has advantages over rational function. First, it is less computationally expensive to evaluate a parametric DTMC thanks to its simpler symbolic expression. Second, DTMC sampling is *parallelizable*; sampling can be done with as many processor cores as possible. The second advantage makes DTMC sampling *scalable*, compare to the rational function evaluation approach, which is not scalable due to its nature of deep recursion.

Algorithm 4 Evaluate terminal state distribution by DTMC sampling

Input: Population model \mathcal{M} of N individuals

Assignment of model parameter r

Number of DTMC samples $nSamples$

Output: D terminal state distribution

```

1: procedure ESTIMATE  $p_i = P(tSCC_i)(S)$ 
2:   Initialize  $D = [0, \dots, 0]$  as an array of  $N + 1$  entries
3:   while not reached  $nSamples$  do
4:     Get  $tSCC_i$ ,  $0 \leq i \leq N + 1$  by simulating evaluated model  $\mathcal{M}$ 
5:     Increase  $D[i]$  by 1
6:   end while
7: end procedure

```

4.3 Inference of model parameter with Metropolis-Hastings algorithm

We can also use Bayesian inference to estimate pMC model parameters directly. Given a pMC \mathcal{M} with $\theta = (\theta_1, \dots, \theta_K)$ as its model parameters, $\theta_i \in [0, 1]$. Note that $\theta_1, \dots, \theta_k$ are not simplex, thus it is not possible to use

Dirichlet prior in this case.

In this method, since there is no possible use of posterior conjugation, thus the hyperparameters estimation is not in the scope of this project. Let $f_i = P(FG \text{ } tSCC_i)$ be estimation of terminal state distribution, we denote $\epsilon_i = f_i(\theta)$ as evaluations of the polynomial function with a concrete assignment of θ . The steady-state distribution has then $Multinomial(N, \epsilon = (\epsilon_0, \dots, \epsilon_n))$ with N is the sample size of experiment data D . The posterior distribution has the following form:

$$\pi(\theta|D) \sim Multinomial(N, (\epsilon_0, \dots, \epsilon_n))\pi(\theta_1) \dots \pi(\theta_k)$$

As the posterior $\pi(\theta|D)$ has no analytical form, we use Metropolis-Hastings to sample from it.

Algorithm 5 Estimation of pMC parameters given a sample D

```

1: procedure METROPOLIS-HASTINGS( $D$ , maxIteration)
2:   Select a proposal distribution  $\pi(\theta)$ 
3:   Draw a random initial point  $\theta$ 
4:   Init empty trace  $Trace$ 
5:   while maxIteration not reached do
6:     Evaluate  $\epsilon \leftarrow (f_0(\theta), \dots, f_N(\theta))$ 
7:      $L \leftarrow Multinomial(D|\epsilon)$ 
8:     Draw a point  $\theta'$  from the proposal distribution.
9:     Evaluate  $\epsilon' \leftarrow (f_0(\theta'), \dots, f_N(\theta'))$ 
10:     $L' \leftarrow Multinomial(D|\epsilon')$ 
11:    if  $\ln(L') - \ln(L) > 0$  then
12:      Add  $\theta'$  to  $Trace$ 
13:       $\theta = \theta'$ 
14:    else
15:      Draw a random number  $x$  from  $Uniform(0, 1)$ 
16:      if  $x \leq \xi$ , ( $\xi$  very small, e.g  $10^{-8}$ ) then
17:        Add  $\theta'$  to  $Trace$  (avoiding local maxima)
18:         $\theta = \theta'$ 
19:      end if
20:    end if
21:  end while
22: end procedure

```

From the $Trace$ returned by the algorithm, we estimate parameter

5 Benchmark

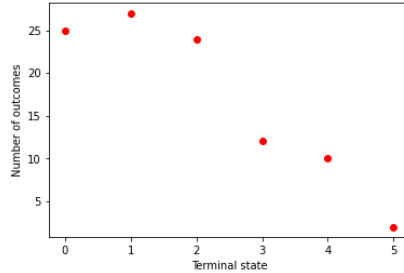
5.1 Data synthesis

In order to better evaluate the performance of Bayesian estimation of model parameters, we use synthetic data, generated from a model with known parameters. In case of rational functions are available, data synthesis for N bees population is conducted as follow:

- Select a pMC model \mathcal{M} to model the population of. Let $\theta = \theta_1, \dots, \theta_k$ is the model parameter.
- Assign a concrete value for θ which satisfies model's constraint for parameter, θ_{true}
- From \mathcal{M} , evaluate terminal state distribution of $f_i(\theta) = P(FG \models tSCC_i)$, $0 \leq i \leq N$.
- Draw sample S from multinomial distribution $Multinomial(N, (f_0(\theta), \dots, f_N(\theta)))$

Synthetic data can also be obtained directly from DTMC sampling. The using of synthetic data and known true parameter allow us to evaluate the inference result more precisely.

```
Model file: data/prism/bee_multiparam_synchronous_5.pm
Result file: data/prism/bee_multiparam_synchronous_5.txt
True parameters: [0.25, 0.3, 0.35, 0.45, 0.5]
Terminal state distribution:
tSCC[0]=0.2373046875
tSCC[1]=0.300125
tSCC[2]=0.24029687500000008
tSCC[3]=0.12100000000000022
tSCC[4]=0.07695312499999996
tSCC[5]=0.024320312499999996
Synthetic data
[25, 27, 24, 12, 10, 2]
Synthetic data distribution
[0.25, 0.27, 0.24, 0.12, 0.1, 0.02]
```



Histogram of synthesized data, 5 bees synchronous, multiparams

Figure 5: Example of data synthesis for 5 bees model.

5.2 Evaluation measures

In order to evaluate the inference result, we consider three aspects:

1. *Accuracy*: distance from estimated parameter to true parameter used to synthesize sample data.
2. *Precision*: width of the Highest Posterior Density corresponding to an estimation.
3. *Cost*: how does physical computation time increases as the size of population increases.

To measure *accuracy*, the distance from estimated parameter to true parameter, we use *Root Mean-Squared Error*.

Definition 5.1 (Root Mean-Square Error). Let $\theta = (\theta_1, \dots, \theta_n)$ and $\hat{\theta} = (\hat{\theta}_1, \dots, \hat{\theta}_n)$ be vectors of n real numbers, the Root Mean Square Error between p and \hat{p} is defined as follow:

$$RMSE(\theta, \hat{\theta}) = \sqrt{\frac{\sum_{i=1}^n (\theta_i - \hat{\theta}_i)^2}{n}}$$

In the following experiments, we use the three measures (*accuracy*, *precision*, and *cost*) to evaluate inference results.

5.3 Inference of terminal state distribution

5.3.1 Experiment setup

This experiment is conducted on multiparameters, synchronous model of 5 bees.

- **Model file:** bee_multiparams_synchronous5.pm
- **PRISM result file:** bee_multiparams_synchronous5.txt
- **True parameter:** [0.25, 0.3, 0.35, 0.45, 0.5]
- **True tSCC distribution:** [0.2373046875, 0.300125, 0.240296875, 0.1210, 0.076953125, 0.0243203125]

In order to compare with the work by [5], we also visualize the interval calculated by the following formula:

$$\theta_i \pm (z_{\alpha/2} \sqrt{\frac{\theta_i(1 - \theta_i)}{N}})$$

where N is the sample size. We setup the experiment as follow:

1. Generate a sample terminal state data of 10 experiments.
2. Estimate terminal states distribution using Bayesian
3. Estimate terminal states and frequentist approach using accumulated data.
4. Visualize the estimated terminal state distribution from both Bayesian and Frequentist approach.
5. Go back to 1.

5.3.2 Results

The following results are from experiment with 5 bees model. Blue line segments visualize result from Bayesian conjugation. Green line segments visualizes result from Frequentist approach. The blue (green) slash in the middle of each line segment visualize corresponding point estimation.

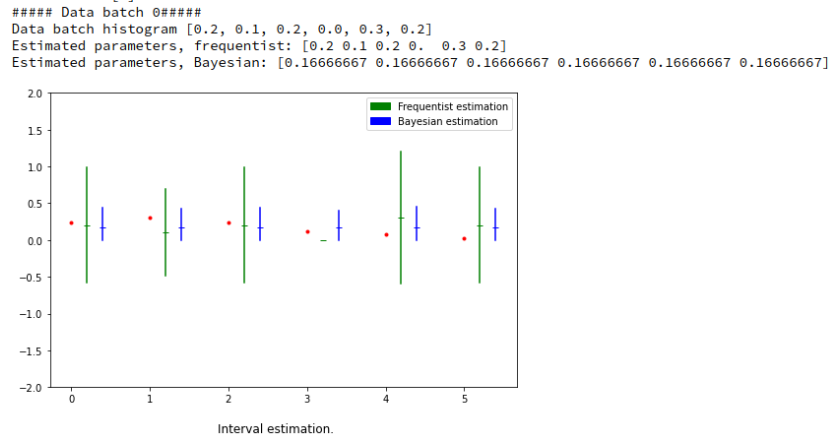


Figure 6: First batch of data. The Bayesian conjugation method suffers from initial false beliefs.

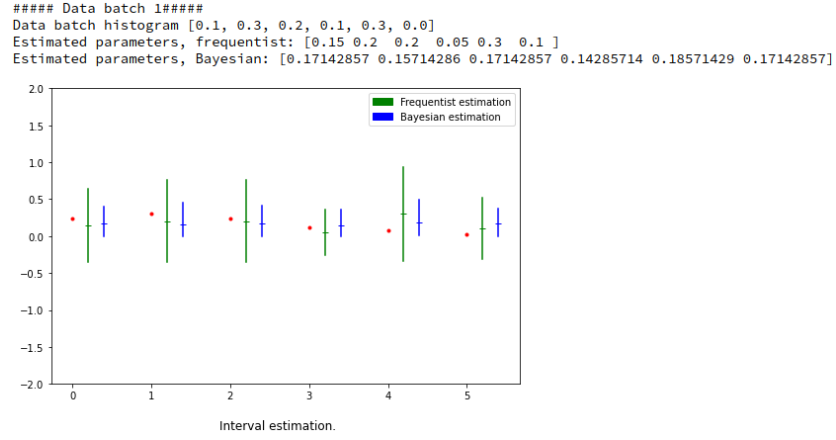


Figure 7: Second batch. Estimation changes significantly as we have a prior belief from previous batch.

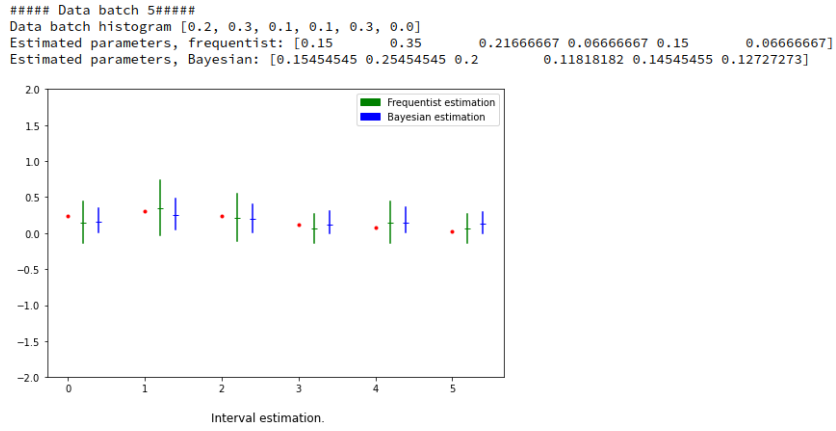


Figure 8: Sixth batch. Estimation improved; Bayesian Credible Sets are narrower.

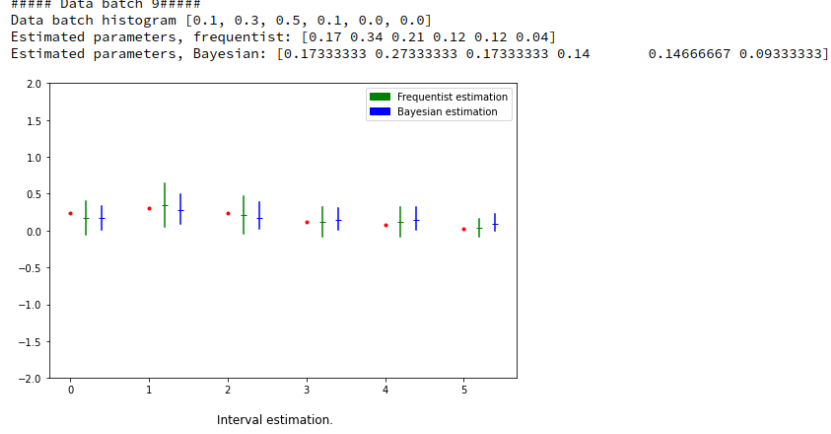


Figure 9: Tenth batch. At this batch, the difference between Bayesian and frequentist estimation is not significant, as now long-run property is satisfied.

5.3.3 Conclusion

With Bayesian Conjugation Multinomial-Dirichlet, we can estimate more precise terminal state distribution than relying only on long-run property as in frequentist approach.

5.4 Compare DTMC sampling and rational functions

5.4.1 Experiment setup:

Given different models and their corresponding (synthesized) true parameters, we compare the accuracy and cost of estimating terminal state distribution by DTMC sampling and rational functions evaluation.

1. Multiparameters, synchronous, 3 bees
 - Model: `bee_multiparams_synchronous_3.pm`
 - Rational functions: `bee_multiparams_synchronous_3.txt`
 - True parameters: `[0.13816683073844171, 0.43244787684875174, 0.9883355614905787]`
2. Multiparameters, synchronous, 5 bees
 - Model: `bee_multiparams_synchronous_5.pm`

- Rational functions: `bee_multiparams_synchronous_5.txt`
- True parameters: `[0.07309162012783865, 0.0732745394348604, 0.3454227397545041, 0.5755326160799704, 0.7400394821424422]`

3. Multiparameters, synchronous, 10 bees

- Model: `bee_multiparams_synchronous_10.pm`
- Rational functions: `bee_multiparams_synchronous_10.txt`
- True parameters: `[0.0010163431033164416, 0.14606188527823205, 0.21868312430906223, 0.29198765300886687, 0.36577363178466926, 0.48812228794389945, 0.5244272612264091, 0.7505607627245182, 0.8753562816925659, 0.9045785878228252]`

4. Multiparameters, synchronous, 15 bees

- Model: `bee_multiparams_synchronous_15.pm`
- Rational functions: `bee_multiparams_synchronous_15.txt`
- True parameters: `[0.06187489978768601, 0.08936613780246316, 0.10191039750975828, 0.3515471096412084, 0.5311583575817167, 0.5652253809036302, 0.6564729149073196, 0.6672538977242365, 0.6715846867598035, 0.7099157316864673, 0.7108067756300934, 0.7839472728483634, 0.8458734212454695, 0.853911885975256, 0.8668668044007563]`

Evaluation by DTMC sampling is conducted with sample size of 2000, 4000, 8000, 10000. We show that the distance between rational function evaluation and DTMC sampling reduces as the sample size increases.

5.4.2 Results

Detailed evaluation results is logged to `example/log/evidence/compare_eval_methods_log.txt`. In this report, first we observe the distance measured in RMSE between estimation by DTMC sampling and rational functions evaluation.

Model	2000 samples	4000 samples	8000 samples	10000 samples
3 bees	0.00722	0.00582	0.01149	0.00356
5 bees	0.00319	0.00347	0.00176	0.00269
10 bees	0.00515	0.00523	0.00212	0.00213
15 bees	0.00321	0.00162	0.00182	0.00181

Table 1: RMSE distance between estimation by DTMC sampling and rational functions evaluation.

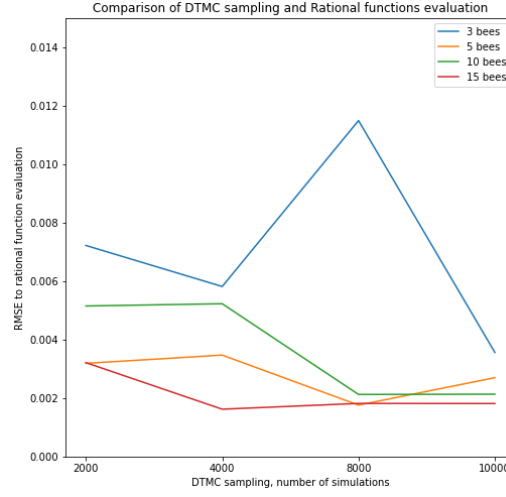


Figure 10: RMSE distance between estimation by DTMC sampling and rational functions evaluation.

As the results show, as we do DTMC sampling with more chain simulations, the distance between DTMC sampling and rational functions evaluation decreases as a result from long-run property.

Model	Rational function	2000 samples	4000 samples	8000 samples	10000 samples
3 bees	0.000396	0.222496	0.318133	0.625835	0.955569
5 bees	0.004574	0.349875	0.427233	0.870164	0.923021
10 bees	0.070688	0.990263	1.527305	2.642697	4.104111
15 bees	2.935404	28.32838	47.95574	86.77052	104.0558

Table 2: Physical runtime comparison, measured in seconds

The rational function evaluation time can be reduced by factorizing rational functions to their simpler form with the help of [10].

Model	Non-factorized	Factorized
3 bees	0.000396	0.000368
5 bees	0.004574	0.000744
10 bees	0.070688	0.036026
15 bees	2.935404	1.215543

Table 3: Physical runtime comparison of evaluation non-factorized and factorized rational functions, measured in seconds

The time needed to process rational functions with factorization increases exponentially to the footprint of rational functions.

Model	Non-factorized rational functions	Factorized rational functions
3 bees	0.002133	0.040748
5 bees	0.024780	0.167829
10 bees	0.326980	20.498155
15 bees	8.906071	10649.155044

Table 4: Physical runtime comparison of parsing rational functions, with and without factorization, measured in seconds

It must be noticed that the cost of factorization, while very high, is paid only once at the model processing, and later the reduction in rational functions evaluation is significant, as the number of rational functions evaluation is linear to the Metropolis-Hastings chain length.

5.4.3 Conclusion

DTMC sampling can be used as an alternative method for terminal state distribution evaluation. It is due to the long-run property that as we do more chain simulations, the estimated terminal state distribution becomes closer to the distribution evaluated from rational functions. DTMC sampling has a significant advantage that it does not need a symbolic model checker to deliver rational functions; it also eliminates the overhead that is induced by reading and parsing large rational functions.

5.5 Inference of model parameters

5.5.1 Experiment setup

We show examples of parameter inference for different bees model in both DTMC sampling and rational functions evaluation methods. Other setups:

- Prior distribution: $Uniform(0, 1)$
- Metropolis-Hastings chain length: 500
- Number of simulations in DTMC sampling mode: 1000.

5.5.2 Results

Results with 3 bees model, multiparameters, synchronous. Results with 5 bees model, multiparameters, synchronous. Results with 10 bees model, multiparameters, synchronous. Results with 15 bees model, multiparameters, synchronous. For large population models which due to the technical issues PRISM cannot deliver rational functions, we can only use DTMC sampling to estimate terminal state. However, as it has been shown on the comparison between rational function evaluation and DTMC sampling, cost for a DTMC simulation also increases as the number of states in the DTMC increases.

5.5.3 Conclusion

The experiment shows the results on parameter inference for agnostic models of 3, 5, 10, and 15 bees.

5.6 Selection of Metropolis-Hastings chain length

5.6.1 Experiment setup

Chain length in Metropolis-Hastings algorithm is the number of accepted point that is added to the sample set, which is returned by the algorithm. In this experiment we evaluate the impact of different Metropolis-Hastings chain length to parameter estimation.

We use 10 bees, multiparameters, synchronous model. Using the same true parameter and synthetic data, we do the parameter inference in different

Metropolis-Hastings trace length, and observe the changes in parameter estimation.

- Model: 10 beesm,

5.6.2 Result

Precision of inference results with different Metropolis-Hastings chain length in the set (1000, 2000, 3000, 4000, 5000) is visualized in the following figure.

```
Model file: data/prism/bee_multiparam_synchronous_10.pm
BSCC file: data/prism/bee_multiparam_synchronous_10.txt
Terminal state evaluation mode: rational functions
True parameters: [0.012152382449267307, 0.06280031298108557, 0.21270070164299149, 0.4288273880974319, 0.5235603692706758, 0.6189846759889027, 0.826236904896725, 0.8961046252598239, 0.9773543220460572, 0.9872661333157495]
```

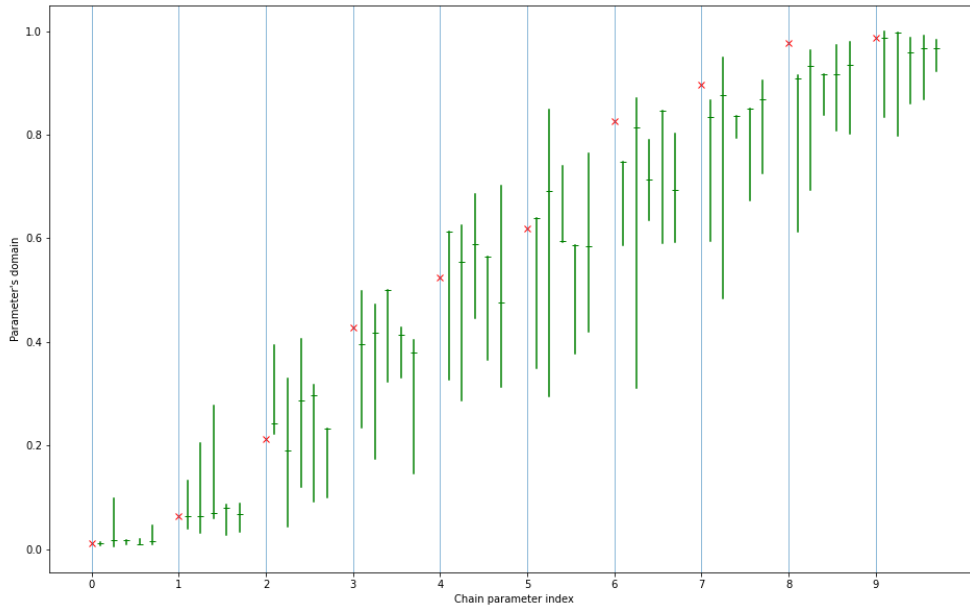


Figure 11: HPD of inference result. Green lines from left to right visualize the HPD of inference with Metropolis-Hastings chain length 1000, 2000, 3000, 4000, 5000. The red cross visualizes true parameter.

Accuracy and runtime:

MH Chainlength	RMSE	Run time (s)
1000	0.050052749138867626	63.83112
2000	0.03084112520832641	125.08478
3000	0.0601108203086657	183.794119
4000	0.04131949555520387	246.105021
5000	0.051880135378818214	306.694507

Table 5: Accuracy (measured by RMSE) and run time (in seconds), inference of 10 bees, multiparameters, synchronous model with different Metropolis-Hastings chain length.

5.6.3 Conclusion

From the results of experiment, we cannot conclude that increasing Metropolis-Hastings chain length also increases parameter estimation accuracy. It is because that there are too many outliers in the results. Due to the fact that

5.7 Selection of prior distribution

5.7.1 Experiment setup

This experiment is to show the impact of prior distribution selection on a given bees model. In this particular setup, we compare $Uniform(0, 1)$ and $Beta(\alpha, \beta)$. Other settings:

- Metropolis-Hastings chain length: 500
- Terminal state distribution evaluation: rational functions evaluation.

5.7.2 Result

5.7.3 Conclusion

As the results show, the selection of prior distribution has a strong impact on For example, in the case of linear model, certain assumption can be made on model parameters so that we can limit the parameter space.

5.8 Impact of data quality

An advantage of using synthetic data over real data obtained from biological experiment is that:

- We know the true parameters.
- We have control over data quality.

Biological experiment, in this project is the experiment with bees, is not able to be conducted massively, for example 1000 experiments, due to the time and effort cost. To few experiments to guarantee the long-run property leads to incorrect parameter inference. In this experiment, we evaluate the impact of low quality data (too few experiment data) to parameter inference.

5.8.1 Experiment setup

Experiment is conducted as follow

-

5.8.2 Result

5.8.3 Conclusion

5.9 Linear model

5.9.1 Experiment setup

Linear models use the same model semantics to multiparameters model. The difference is that it assume that parameters are linearly related.

1. Multiparameters, synchronous, 5 bees
 - Model: `bee_multiparams_synchronous_5.pm`
 - Rational functions: `bee_multiparams_synchronous_5.txt`
 - True parameters: `[0.1, 0.2]`
 - Synthetic data (10000 trials): `[3253, 2421, 1763, 1257, 870, 436]`
2. Multiparameters, synchronous, 10 bees
 - Model: `bee_multiparams_synchronous_10.pm`
 - Rational functions: `bee_multiparams_synchronous_10.txt`
 - True parameters: `[0.01, 0.02]`

- Synthetic data (10000 trials): [8161, 1523, 267, 46, 3, 0, 0, 0, 0, 0, 0]

As linear constraint in linear model imposes that $r_i = a \times i + b$, in which a and b are model parameters. This results in two important observations on prior distribution:

1. Values range in $[0, 1]$
2. Strongly skewed to small values of a and b in range $[0, 1]$, otherwise it results in $r_i > 1$, which is invalid.

Using these observations as our prior beliefs, we select the distribution $Beta(\alpha = 1, \beta = 10)$ as prior distribution.

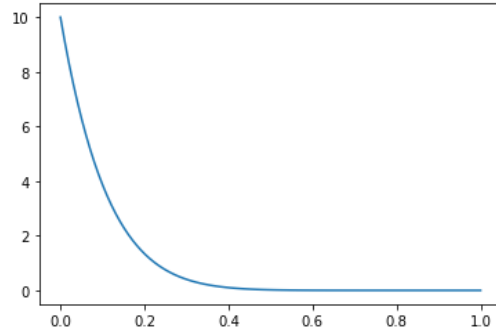


Figure 12: $Beta(\alpha = 1, \beta = 10)$ prior distribution. The distribution is skewed to lower values of rang $[0, 1]$.

5.9.2 Results

The following results are from experiment with 5 bees and 10 bees models. The green bars visualize HPD of estimations, with a dash in betwwen to visualize the point estimation.

True parameters: $a=0.1$, $b=0.2$
 Estimated parameters: $a=0.092014759787157$, $b=0.20614608892292768$
 RMSE: 0.007125253395430705
 Physical runtime: 2.35766604 secs
 AIC: 59.695671

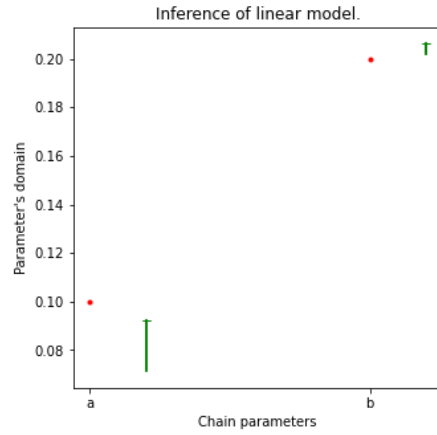


Figure 13: Inference of Linear 5 bees model.

Result with 10 bees model.

True parameters: $a=0.01$, $b=0.02$
 Estimated parameters: $a=0.009878094156066811$, $b=0.020278274487981494$
 RMSE: 0.0002148228635951324
 Physical runtime: 129.908486 secs
 AIC: 33.6277592

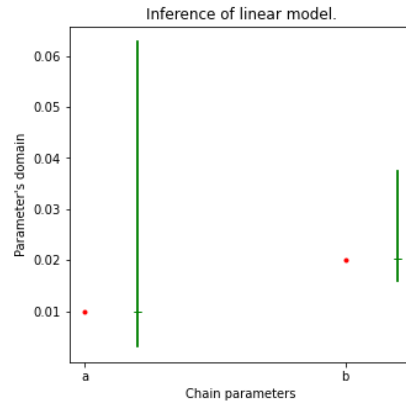


Figure 14: Inference of Linear 10 bees model.

5.9.3 Conclusion

Linear model helps to reduce the number of model parameters by enforcing a linear constraint, thus all multiparameters model reduced into 2 parameters. With the proper selection of prior distribution to avoid too many invalid parameter evaluation, parameters of linear models can be estimated with high likelihood and relatively cheap physical runtime cost.

6 Implementation

6.1 Class design

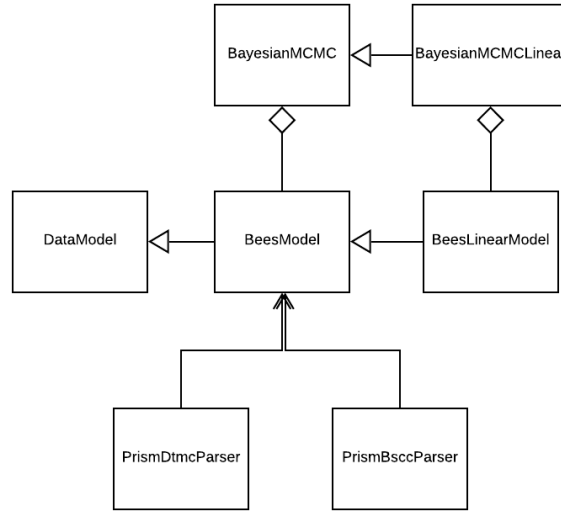


Figure 15: Class diagram of the project

Class explanation:

1. **PrismModelParser** and **PrismBscParser** parses PRISM model file and symbolic model checking results. In details, the two classes do the following tasks:
 - Read PRISM model and result files.
 - Extract and preprocess expressions from model and result.

- Factorize expressions if needed
 - Parse expressions in model and result file. store in Python byte-code.
 - Create `DataModel` object to store these parsed expressions.
2. `DataModel`, `BeesModel`, and `BeesLinearModel` store information about the population model.
 3. `BayesianMcmc` and `BayesianMcmcLinear` implements Metropolis-Hastings algorithm.

6.2 Test environment

The Bayesian inference methods are implemented in Python 3 and tested in the followin system configuration:

- Intel Core i5-8265U @ 1.60GHz
- 16GB RAM
- OpenSUSE Tumbleweed 20200427
- Anaconda 3 2019.10 for linux x86_64

6.3 Directory structure

- `data` folder contains PRISM models and model checking results.
- `docs` folder contains final report
- `scripts` folder contains python implementation.
- `examples` folder contains experiment schemes.
 - `examples/evidence` contains log of the examples that are used as data for this report.

7 Conclusion

The goal of this project is to experiment Bayesian approach on parameter inference of Markov population models. The results shows that Bayesian inference is efficient to deliver an estimation of model parameters. Given that in the biological system that is modelled, each parameters represent a real-world biological property, such as the probability of a bee sting, given a certain amount of pheromone, the parameter estimation presented in this project helps biologists to better understand the biological mechanism.

The implementation in this project is capable of estimating parameters in both cases, when rational functions are available and is not available. This is an advantage in case that model checker cannot deliver rational functions due to technical issues. Also, with the parallelization of DTMC sampling, the implementation proves that it has significant scalability for models of large populations. The implementation also works with linear models and 2 parameters model, with significant reduction of physical runtime.

While the implementation in this project only gives point estimations and their corresponding credible set, the parameter synthesis method presented in [5] gives a detailed analysis of accepted regions and rejected regions on parameter space and does not need to manually select a proper prior distribution. We suggest that the implementation in this project should be intergrated to the tool DiPS in [5] as an extension, so user can use it as an option. By doing so, we can exploit the advantages of both methods depend on particular situations.

There are still problems, which are not completely solved in the scope of this project:

1. *Number or parameters*: Accuracy and precision of an estimation decreases, while the computational cost increases, as the number of model parameter increases.
2. *Selection of prior*: a false selection of prior distribution and its parameters could propagate false beliefs over the whole inference. So far we have been using Uniform distribution by default to prevent This project has not coped with the problem of hyperparameters estimation.
3. Using *STORM* [4] as a more effective model checker in stead of PRISM.

STORM is capable of handling rational functions with larger footprint and factorizing these functions. However due to technical problems and time constraint, *STORM* is not used in this project.

4. There is still a problem of number overflow that appears randomly in case of evaluation of parametric DTMC in large model (for example, 20 bees multiparameters, synchronous). However, the root cause of this issue is unknown and thus it is not solved.

References

- [1] Greg M Allenby, Peter E Rossi, and RE McCulloch. “Hierarchical Bayes Models: A Practitioners Guide. Grover R, Vriens M, eds”. In: *SSRN Electron J* (2005).
- [2] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT press, 2008.
- [3] Michael Baron. *Probability and statistics for computer scientists*. CRC Press, 2019.
- [4] 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.
- [5] Matej Hajnal et al. “Data-Informed Parameter Synthesis for Population Markov Chains”. In: *International Workshop on Hybrid Systems Biology*. Springer. 2019, pp. 147–164.
- [6] Lisa Hutschenreiter, Christel Baier, and Joachim Klein. “Parametric Markov chains: PCTL complexity and fraction-free Gaussian elimination”. In: *arXiv preprint arXiv:1709.02093* (2017).
- [7] 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.
- [8] 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.
- [9] M. Kwiatkowska, G. Norman, and D. Parker. “PRISM 4.0: Verification of Probabilistic Real-time Systems”. In: *Proc. 23rd International Conference on Computer Aided Verification (CAV’11)*. Ed. by G. Gopalakrishnan and S. Qadeer. Vol. 6806. LNCS. Springer, 2011, pp. 585–591.
- [10] Aaron Meurer et al. “SymPy: symbolic computing in Python”. In: *PeerJ Computer Science* 3 (2017), e103.
- [11] 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.

- [12] John Salvatier, Thomas V Wiecki, and Christopher Fonnesbeck. “PyMC3: Python probabilistic programming framework”. In: *ascl* (2016), ascl–1610.
- [13] Stephen Tu. “The dirichlet-multinomial and dirichlet-categorical models for bayesian inference”. In: *Computer Science Division, UC Berkeley* (2014).