mathematics

Article

Parameter Tuning of Agent-Based Models: Metaheuristic Algorithms

Andrei I. Vlad *, Alexei A. Romanyukha and Tatiana E. Sannikova

Marchuk Institute of Numerical Mathematics, Russian Academy of Sciences, 119333 Moscow, Russia; eburg@inm.ras.ru (A.A.R.); t.sannikova@inm.ras.ru (T.E.S.)
* Correspondence: firkhraag@gmail.com

Abstract: When it comes to modelling complex systems using an agent-based approach, there is a problem of choosing the appropriate parameter optimisation technique. This problem is further aggravated by the fact that the parameter space in complex agent-based systems can have a large dimension, and the time required to perform numerical experiments can be large. An alternative approach to traditional optimisation methods are the so-called metaheuristic algorithms, which provide an approximate solution in an acceptable time. The purpose of this study is to compare various metaheuristic algorithms for parameter tuning and to analyse their effectiveness applied to two agent-based models with different complexities. In this study, we considered commonly used metaheuristic algorithms for agent-based model optimisation: the Markov chain Monte Carlo method, the surrogate modelling approach, the particle swarm optimisation algorithm, and the genetic algorithm, as well as the more novel chaos game optimisation algorithm. The proposed algorithms were tested on two agent-based models, one of which was a simple toy model of the spread of contagious disease, and the other was a more complex model of the circulation of respiratory viruses in a city with 10 million agents and 26 calibrated parameters.

Keywords: agent-based model; model optimisation; parameter tuning; metaheuristic algorithms

MSC: 68T20; 68T42; 90C59; 93A14; 93A16; 93B30

check for updates

Citation: Vlad, A.I.; Romanyukha, A.A.; Sannikova, T.E. Parameter Tuning of Agent-Based Models: Metaheuristic Algorithms. *Mathematics* **2024**, *12*, 2208. https://doi.org/10.3390/math12142208

Academic Editors: Florin Leon and Biao Tang

Received: 15 May 2024
Revised: 27 June 2024
Accepted: 13 July 2024
Published: 15 July 2024

1. Introduction

Agent-based models (ABMs) [1] are used to model systems of varying complexity in a wide variety of applications [2], where the correct estimation of the model parameters becomes essential for the successful model application. Their parameter space can have a large dimension (>20), which makes the task of finding an optimal solution extremely difficult considering the increased demands of ABMs on the hardware and software performance, and the number of possible solutions is usually far too large for a simple enumeration.

When it comes to ABMs, finding an optimal solution to an optimisation problem is a very challenging task that is often unattainable. Due to the complexity of ABMs, most of them are incompatible with methods requiring analytical expressions, such as gradient-based methods that are model-specific and require rather simple model structure and no complex decision rules for the agents [3]. The problem is further complicated by the fact that it is often infeasible to identify marginal and conditional likelihood functions [4]. Therefore, the idea is to have an algorithm that can produce a solution that would be good enough according to the comparison of model outputs with different parameter values to empirical data using some distance-based loss function.

While there are no agreed guidelines for the choice of the optimisation algorithm for complex ABMs, the most common way to tackle the problem is by using metaheuristic methods [5], since they allow one to find solutions that are close to optimal with lower computational costs. The main advantage of metaheuristic methods lies in the fact that they do not require any knowledge about the model and can be equally applied to any model. While there are many metaheuristic algorithms that can be used for model optimisation, choosing the right one for the given problem is a challenging task considering the limited time and resources for testing different algorithms. The comparison of methods in the literature is quite limited due to the abundance of different metaheuristic algorithms [6–8].

Studies in the field [9–11] tend to introduce new optimisation methodologies without comparing them to well-established methods. This, coupled with the lack of rigorous testing of the quality of the estimates produced by them applied to complex agent-based systems, presents a difficulty for a modeller to choose one particular method over another. In this work, we analysed well-established metaheuristic methods commonly used to solve the problem of identifying the parameters in ABMs in the literature [11–19]. The methods we consider in this study are the Markov chain Monte Carlo method (MCMC), the surrogate modelling method (SM), the particle swarm optimisation (PSO) algorithm, the genetic algorithm (GA), and the chaos game optimisation (CGO) algorithm.

Most works that focus on comparing different optimisation methods only apply them to highly simplified models with a number of parameters less than 20 [4]. For testing purposes, we use two ABMs of different complexity in the area the authors are most familiar with, namely, epidemiology. The source code of the models is publicly available in the GitHub repository. The first one is a simple toy ABM with four calibrated parameters without any practical value, which is used to model the spread of the contagious disease in the population where agents are divided into three groups based on their state: susceptible to the disease, infectious, or recovered (ABM-SIR). The other model is a more complex ABM of the dynamics of acute respiratory viral infections in a city with a population of about 10 million people (ABM-ARI) with 26 calibrated parameters. The model simulates one year and has a discrete time step equal to a single day. Transmission of the viruses occurs through close contact between agents in the same social setting, which is modelled using complete graphs for households and Barabasi–Albert graphs for educational institutions and workplaces. Agents can also be in different states: susceptible to the virus, exposed, infectious, or recovered according to the stage of the infection.

This paper proceeds as follows: first, we conduct a literature analysis (Section 2). Next, we describe in detail every metaheuristic algorithm used in this study (Section 3). Then, we describe two ABMs that we use to test the proposed methods (Section 4). We present the results of numerical experiments using different model calibration techniques (Section 5). In the last section, we summarise our findings and conclude with a discussion about the obtained results (Section 6).

2. Related Work

Parameter calibration poses a significant challenge when it comes to ABMs. There have been a number of articles describing the application of different metaheuristic methods aiding agent-based model calibration.

The work [12] proposed using the Markov chain Monte Carlo method (MCMC), namely, the Metropolis algorithm, for the exploration of the parameter space of the discrete-time agent-based economic model simulating the behaviour of the consumer and financial markets. The model had 8 calibrated parameters, and simulations consisted of 2 competing firms, 50 stock traders, and 200 consumers. The authors found that the proposed method allows for the efficient exploration of large parameter spaces and leads to the reproduction of empirical phenomena. Another example of using MCMC for the model calibration is the work [13], where the Metropolis–Hastings algorithm was applied to the likelihood of the recorded number of deaths in Liberia from Ebola virus disease to find the values of three parameters of interest.

In the article [18], the authors proposed a method based on genetic algorithms (GAs) for exploring the parameter space of ABMs. The method was tested on a simple ABM simulating the ant foraging with the evolution of two parameters: the diffusion rate and the evaporation rate. The results differed significantly depending on the choice of the fitness function. Another example of using GA for parameter tuning is the article [19], where an ABM with eight parameters used to model retail petrol prices was calibrated with two different methods, such as GA and modeller-driven parameter calibration, which produced close results.

The work [11] proposed a method that combines supervised machine learning and intelligent sampling in the design of a surrogate meta-model, which constitutes a computationally cheap approximation of the real model. The authors used the XGBoost machine learning algorithm for the training of two surrogate models for the Brock and Hommes model and the Islands model with 10 and 6 calibrated parameters, respectively. This approach was further explored in [15–17] in which such algorithms as the CatBoost machine learning algorithm, support vector machine, and the neural network were used to train the surrogate modelling. The authors of [17], based on the Linked Lives ABM of social care provision in the UK with 10 parameters of interest, drew the conclusion that a deep learning approach is the most promising candidate to create a surrogate of the ABM.

In the work [14], the authors analysed the performance of such metaheuristic algorithms as the genetic algorithm, the firefly algorithm, the particle swarm optimisation algorithm, and the artificial bee colony algorithm by using three ABMs, namely, the predator–prey model with three specified parameters, the eight queens model with eight parameters, and the flow zombies model with three parameters. The results showed that each of the algorithms achieved acceptable accuracy for each ABM. However, for each problem, a different algorithm was most successful.

3. Parameter-Tuning Algorithms

While calibrating ABMs we use point estimation methods aimed at finding a single parameter combination that will produce the best fit to the data according to the defined loss function. In this study, we use the root mean square error ($RMSE$):

$$RMSE = \sqrt{\frac{1}{n}\sum_{k=1}^{n}(y_k^d - y_k^m)^2}, \tag{1}$$

where y_k^d and y_k^m are the observed values for the step k according to the data and as a result of a numerical experiment from the model, respectively.

Initial selection of parameter values can be done manually based on the domain knowledge or using statistical methods for generating parameter samples. In this work, we use the Latin hypercube sampling method [20]. The Latin hypercube of k parameters and N samples is constructed by partitioning the intervals of possible values of each of k parameters into N intervals of equal length, after which a value is randomly taken from each of these intervals, which is then randomly combined with the values of other parameters. One value of one parameter can only correspond to one value of another parameter.

For convenience, symbols used to describe parameter-tuning algorithms are summarised in Tables 1 and 2.

Table 1. General symbols and their meanings used to describe parameter-tuning algorithms.

Symbol	Description
$\Theta = \{\theta_j \in [\theta_j^{\min}, \theta_j^{\max}]\}$	Set of parameters
$RMSE_\Theta$	Root mean square error associated with Θ
N^{params}	Number of parameters
t	Step of the algorithm
T^{max}	Maximum step of the algorithm
ABM	Numerical experiment with the agent-based model

Table 2. Algorithm-specific symbols and their meanings.

Symbol	Description
	Markov chain Monte Carlo
$\hat{\Theta} = \{\hat{\theta}_j\}$	Candidate set of parameters
N^{reject}	Number of consecutive candidate rejections
	Surrogate model
SM	Surrogate model for ABM approximation
X	Features of the training set
Y	Labels of the training set
	Particle swarm optimisation
$N^{particle}$	Number of particles
V_i	Velocity of particle i
$w \in [w_{\min}, w_{\max}]$	Inertia weight hyperparameter
c_1	Personal acceleration hyperparameter
c_2	Social acceleration hyperparameter
P_i	Best found position of particle i
G	Best found solution by all particles
	Genetic algorithm
N^{pop}	Size of the population of parameter sets
N^{parent}	Number of parents for crossover procedure
$\tilde{\Theta} = \{\tilde{\theta}_j\}$	Offspring set of parameters
	Chaos game optimisation
N^{seed}	Number of seeds
GB	Global best position of the seed
MG	Mean position of the group containing random seeds
S_i^k	Position of the potential k-th seed spawned by i-th seed

3.1. Markov Chain Monte Carlo Method

Markov chain Monte Carlo (MCMC) methods [21] are used to draw samples from an unknown probability distribution where each next sample is dependent on the existing one. The proposed algorithm (Algorithm 1) is as follows:

1. In the first step, the intervals of possible parameter values are determined and their initial values are set $\{\theta_j^1\}_{j=1}^{N^{params}}$.
2. Next, at each step t of the algorithm, a candidate set of model parameter values $\hat{\Theta}^t = \{\hat{\theta}_j^t\}_{j=1}^{N^{params}}$ is generated based on the set parameters from the previous step Θ^{t-1}. We transform previous parameter values so that the candidates would stay within parameter boundaries:

$$\hat{\theta}_j^t = \frac{\theta_j^{\max} \exp(\delta_j^t) + \theta_j^{\min}}{1 + \exp(\delta_j^t)},$$
$$\delta_j^t \sim \mathcal{N}(\log(\frac{\theta_j^{t-1} - \theta_j^{\min}}{\theta_j^{\max} - \theta_j^{t-1}}), \sigma^2), \tag{2}$$

where σ^2 is a hyperparameter of the algorithm corresponding to the variance for generating candidate parameter values. If the parameter value is equal to the boundary value, we add a small value to keep it in the defined boundaries.

3. Next, we either accept or reject the candidates based on the loss function. If the loss for the candidate is lower than the loss on the previous step, the candidate set of parameters is accepted as the new set in the next step of the algorithm, and rejected otherwise.
4. To avoid local minima, we accept a new set of parameters after 10 consecutive failures.
5. Repeat the algorithm from Step 2 until the specified number of iterations is reached and select the set of parameter values with the lowest $RMSE$.

Algorithm 1: Markov chain Monte Carlo algorithm

```
Input: $\Theta^0$, $RMSE_{\Theta^0}$, ABM
Output: $\text{argmin}_{\Theta^t} RMSE_{\Theta^t}$
$N^{reject} \leftarrow 0$
for $t = 1$ to $T^{max}$ do
    Set candidate set of parameter values $\hat{\Theta}^t$ (2)
    Find corresponding error $RMSE_{\hat{\Theta}^t} \leftarrow$ ABM($\hat{\Theta}^t$)
    if $RMSE_{\hat{\Theta}^t} < RMSE_{\Theta^{t-1}}$ or $N^{reject} = 10$ then
        $\Theta^t \leftarrow \hat{\Theta}^t$
        $RMSE_{\Theta^t} \leftarrow RMSE^t_{\hat{\Theta}^t}$
        $N^{reject} \leftarrow 0$
    else
        $\Theta^t \leftarrow \Theta^{t-1}$
        $RMSE_{\Theta^t} \leftarrow RMSE_{\Theta^{t-1}}$
        $N^{reject} \leftarrow N^{reject} + 1$
```

3.2. Surrogate Modelling Method

The surrogate modelling approach (SM) for solving the parameter identification problem is based on the work [11]. This algorithm is defined by the fact that the results obtained from numerical experiments with ABM are approximated using the surrogate model. The process of training a surrogate model (Algorithm 2) is as follows:

1. In the first step, we build the initial training set for the surrogate model by conducting numerical experiments using sets of parameter values obtained from the Latin hypercube sampling method.
2. We use XGBoost [22], a gradient-boosting algorithm that creates an ensemble of decision trees for training the surrogate model on the resulting training set.
3. We use the Markov chain Monte Carlo method with the $RMSE$ (Section 3.1), but in this case we use the trained surrogate model to predict the value of the loss function, so we obtain a set of parameter values that gives a minimum of the loss function according to the surrogate model.
4. We conduct a numerical experiment with the ABM using set of obtained parameter values and find corresponding loss.
5. We update the training set.
6. Repeat the algorithm from Step 2 until the specified number of iterations is reached and select the set of parameter values with the lowest $RMSE$.

Algorithm 2: Surrogate modelling algorithm

```
Input: $\Theta_1^0, \ldots, \Theta_{N^{train}}^0$, $RMSE_{\Theta_1^0}, \ldots, RMSE_{\Theta_{N^{train}}^0}$, ABM, SM
Output: $\mathrm{argmin}_{\Theta^t} RMSE_{\Theta^t}$
1 $X^1 \leftarrow \cup_{k=1}^{N^{train}} \Theta_k^0$
2 $Y^1 \leftarrow \cup_{k=1}^{N^{train}} RMSE_{\Theta_k^0}$
3 for $t$ = 1 to $T^{max}$ do
4     Train SM using $X^t$ and $Y^t$
5     Find $\Theta^t$ using trained SM
6     $RMSE_{\Theta^t} \leftarrow$ ABM($\Theta^t$)
7     $X^{t+1} \leftarrow X^t \cup \Theta^t$
8     $Y^{t+1} \leftarrow Y^t \cup RMSE_{\Theta^t}$
```

3.3. Particle Swarm Optimisation Algorithm

Particle swarm optimisation (PSO) algorithm [23] simulates the behaviour of particles observing their positions in the space of the model parameters and the vector of their movement speed. In addition, particles store the position of the best solution found for a particular particle, as well as the best solution among all particles. The algorithm (Algorithm 3) is as follows:

1. In the first step, the number of particles, the values of the hyperparameters of the algorithm, as well as positions of particles, their boundaries, and initial velocities are specified.
2. We update the velocity of each particle on each step t.
3. We update the position of each particle, and if it goes out of the boundaries, we sample the parameter value from the uniform distribution.
4. We conduct numerical experiments with the new positions using the model. If the $RMSE$ of the particle is less than the lowest error found by this particle or found by all particles, we update the corresponding $RMSE$ values and best positions.
5. Repeat the algorithm from Step 2 until the specified number of iterations is reached and select the set of parameter values with the lowest $RMSE$.

3.4. Genetic Algorithm

Genetic algorithm (GA) [24] imitates the process of natural selection using selection, crossover, and mutation operators. The algorithm (Algorithm 4) proceeds as follows:

1. In the first step, we choose the size of the population consisting of parameter value combinations and the number of parents we want to select for reproduction. We also define the initial population of parameter values using Latin hypercube sampling method.
2. Selection: we select potential parents from the existing population using the binary tournament selection method where we select the best out of two random individuals from the population in each tournament round based on $RMSE$.
3. Crossover: we randomly select two parents from the pool of potential parents. Crossovers occur randomly with a given probability using one-point crossover, where we split the set of parameter values into two sets and exchange them between the parents, creating two offsprings.
4. Mutations occur randomly with a given probability by adding Gaussian noise to one or more parameter values of the offspring to avoid the stagnation of the algorithm: $\theta_{ij} = \theta_{ij} + \delta_j$, where $\delta_j \sim \mathcal{N}(0, \sigma(\theta_j^{\max} - \theta_j^{\min}))$, $i = 1, \ldots, N^{pop}$, and $j = 1, \ldots, N^{params}$. If the parameter value goes out of the boundaries, we set $\theta_{ij} \sim \mathcal{U}(\theta_j^{\min}, \theta_j^{\max})$.
5. Finally, following the elitism strategy, we replace the current population with the best individuals from the current and offspring populations combined together.

6. Repeat the algorithm from Step 2 until the specified number of iterations is reached and select the set of parameter values with the lowest $RMSE$.

Algorithm 3: Particle swarm optimisation algorithm

```
Input: $\Theta_1^0, \dots, \Theta_{N^{particle}}^0, V_1^0, \dots, V_{N^{particle}}^0, RMSE_{\Theta_1^0}, \dots, RMSE_{\Theta_{N^{particle}}^0}$, ABM
Output: $G^{T^{max}}$
1  $P_1^0, \dots, P_{N^{particle}}^0 \leftarrow \Theta_1^0, \dots, \Theta_{N^{particle}}^0$
2  $G^0 \leftarrow \mathrm{argmin}_{\Theta_i^0} RMSE_{\Theta_i^0}$
3  for $t = 1$ to $T^{max}$ do
4      for $i = 1$ to $N^{particle}$ do
5          $w^t \leftarrow (w_{\max} - w_{\min}) \frac{T_{\max} - t}{T_{\max}} + w_{\min}$
6          $r_1^t, r_2^t \sim \mathcal{U}(0,1)$
7          $V_i^t \leftarrow w^t V_i^{t-1} + c_1 r_1^t \left(P_i^{t-1} - \Theta_i^{t-1}\right) + c_2 r_2^t \left(G^{t-1} - \Theta_i^{t-1}\right)$
8          $\Theta_i^t \leftarrow \Theta_i^{t-1} + V_i^t$
9          if $\theta_{ij}^t > \theta_j^{\max}$ or $\theta_{ij}^t < \theta_j^{\min}$ then
10             $\theta_{ij}^t \sim \mathcal{U}(\theta_j^{\min}, \theta_j^{\max})$
11         $RMSE_{\Theta_i^t} \leftarrow \mathrm{ABM}(\Theta_i^t)$
12         if $RMSE_{\Theta_i^t} < RMSE_{P_i^{t-1}}$ then
13             $RMSE_{P_i^t} \leftarrow RMSE_{\Theta_i^t}$
14             $P_i^t \leftarrow \Theta_i^t$
15         else
16             $RMSE_{P_i^t} \leftarrow RMSE_{P_i^{t-1}}$
17             $P_i^t \leftarrow P_i^{t-1}$
18         if $RMSE_{\Theta_i^t} < RMSE_{G^{t-1}}$ then
19             $RMSE_{G^{t-1}} \leftarrow RMSE_{\Theta_i^t}$
20             $G^t \leftarrow \Theta_i^t$
21         else
22             $RMSE_{G^t} \leftarrow RMSE_{G^{t-1}}$
23             $G^t \leftarrow G^{t-1}$
```

Algorithm 4: Genetic algorithm

```
Input: $\Theta_1^0, \dots, \Theta_{N^{pop}}^0, RMSE_{\Theta_1^0}, \dots, RMSE_{\Theta_{N^{pop}}^0}$, ABM
Output: $\mathrm{argmin}_{\Theta^t} RMSE_{\Theta^t}$
1  for $t = 1$ to $T^{max}$ do
2      $\hat{\Theta}_1^t, \dots, \hat{\Theta}_{N^{parent}}^t \leftarrow \mathrm{selection}(\Theta_1^{t-1}, \dots, \Theta_{N^{pop}}^{t-1}, RMSE_{\Theta_1^{t-1}}, \dots, RMSE_{\Theta_{N^{pop}}^{t-1}})$
3      $\bar{\Theta}_1^t, \dots, \bar{\Theta}_{N^{pop}}^t \leftarrow \mathrm{crossover}(\hat{\Theta}_1^t, \dots, \hat{\Theta}_{N^{parent}}^t)$
4      $\tilde{\Theta}_1^t, \dots, \tilde{\Theta}_{N^{pop}}^t \leftarrow \mathrm{mutation}(\bar{\Theta}_1^t, \dots, \bar{\Theta}_{N^{pop}}^t)$
5      if $\tilde{\theta}_{ij}^t > \theta_j^{\max}$ or $\tilde{\theta}_{ij}^t < \theta_j^{\min}$ then
6          $\theta_{ij}^t \sim \mathcal{U}(\theta_j^{\min}, \theta_j^{\max})$
7      for $i = 1$ to $N^{pop}$ do
8          $RMSE_{\tilde{\Theta}_i^t} \leftarrow \mathrm{ABM}(\tilde{\Theta}_i^t)$
9      $\Theta_1^t, \dots, \Theta_{N^{pop}}^t \leftarrow \mathrm{next\_generation}(\Theta_1^{t-1}, \dots, \Theta_{N^{pop}}^{t-1}, RMSE_{\Theta_1^{t-1}}, \dots,$
         $RMSE_{\Theta_{N^{pop}}^{t-1}}, \tilde{\Theta}_1^t, \dots, \tilde{\Theta}_{N^{pop}}^t, RMSE_{\tilde{\Theta}_1^t}, \dots, RMSE_{\tilde{\Theta}_{N^{pop}}^t})$
```

3.5. Chaos Game Optimisation

The chaos game optimisation (CGO) algorithm is a novel approach to a model calibration first described in [25]. This algorithm is based on the chaos theory principles and goes as follows (Algorithm 5):

1. In the first step, we choose the number of seeds (N^{seed}) consisting of parameter value combinations and set the initial seed positions using the Latin hypercube sampling method.
2. Find GB the best seed according to $RMSE$.
3. For each seed i, we determine the mean value of the group (MG_i) defined as the group of randomly selected seeds with equal probability of including the current one.
4. For each seed i we find the positions of four new potential seeds S_i^1, S_i^2, S_i^3, and S_i^4 and check their boundary conditions. If the parameter value goes out of the boundaries, we set $s_{ij} \sim \mathcal{U}(\theta_j^{\min}, \theta_j^{\max})$.
5. Replace current seeds with N^{seed} best seeds from the current and potential seeds combined together.
6. Repeat the algorithm from Step 2 until the specified number of iterations is reached and select the set of parameter values with the lowest $RMSE$.

Algorithm 5: Chaos game optimisation algorithm

```
   Input: $\Theta_1^0, \dots, \Theta_{N^{seed}}^0, RMSE_{\Theta_1^0}, \dots, RMSE_{\Theta_{N^{seed}}^0}$, ABM
   Output: $\mathrm{argmin}_{\Theta^t} RMSE_{\Theta^t}$
1  for $t$ = 1 to $T^{max}$ do
2      for $i$ = 1 to $N^{seed}$ do
3          $GB^t \leftarrow \mathrm{argmin}_{\Theta^{t-1}} RMSE_{\Theta^{t-1}}$
4          $MG_i^t \leftarrow \mathrm{mean}\{\Theta_i^{t-1}\}_{i=i_1,\dots,i_k}$
5          $r^t, r_1^t, r_2^t \sim \mathcal{U}(0,1)$
6          Set randomly $\alpha_i^{1,2,3} \in \{r^t, 2r^t, r_1^t r^t + 1, r_2^t r^t + (1 - r_2^t)\}$
7          Set randomly $\beta_i^{1,2,3}, \gamma_i^{1,2,3} \in \{1, 2\}$
8          $S_i^{1t} \leftarrow \Theta_i^{t-1} + \alpha_i^1 \times (\beta_i^1 \times GB^t - \gamma_i^1 \times MG_i^t)$
9          $S_i^{2t} \leftarrow GB^t + \alpha_i^2 \times (\beta_i^2 \times MG_i^t - \gamma_i^2 \times \Theta_i^{t-1})$
10         $S_i^{3t} \leftarrow MG_i^t + \alpha_i^3 \times (\beta_i^3 \times GB^t - \gamma_i^3 \times \Theta_i^{t-1})$
11         Set randomly $S_i^{4t} \leftarrow \{\theta_{ij}^{t-1}$ or $s_{ij}^t \sim \mathcal{U}(\theta_j^{\min}, \theta_j^{\max})\}$
12         Check boundary conditions for $S_i^{1t}$, $S_i^{2t}$, $S_i^{3t}$, and $S_i^{4t}$
13         Find $RMSE$ for $S_i^{1t}$, $S_i^{2t}$, $S_i^{3t}$, and $S_i^{4t}$ by using ABM
14     Set $\{\Theta_i^t\}_{i=1}^{N^{seed}}$ to be best seeds from $\{\Theta_i^{t-1}, S_i^{1t}, S_i^{2t}, S_i^{3t}, S_i^{4t}\}_{i=1}^{N^{seed}}$
```

4. Agent-Based Models

In order to observe the effectiveness of the proposed methods, we use two different ABMs: the first one is a simple toy example without any practical value, while the other is a complex model with a real application, so we can see how the behaviour of the proposed methods changes with increased complexity.

4.1. Agent-Based Model of the Spread of Contagious Disease (ABM-SIR)

As a toy model, we use the ABM that simulates the spread of contagious diseases. Each agent has three states: susceptible to the disease, infectious, and recovered (SIR), similar to the SIR model [26]. The model has discrete time steps, and on each step, agents interact with each other with the number of contacts following Poisson distribution. If an infectious agent interacts with a susceptible one, the latter can be infected with a certain probability. Infectious agents recover after some time and become immune to the disease.

The population consists of 100 thousand agents. The model simulates 40 days with a step of 2.4 h.

The step of the model is as follows: we iterate over the agents, and if we find a susceptible agent, we randomly choose agents with whom it would make contact. If we choose an infectious agent, there is a defined probability that the virus will be transmitted to the susceptible agent. If we find an infectious agent, it can recover with a defined probability.

The model has four adjustable parameters: β is the probability of the disease transmission from an infectious agent to a susceptible one, c is the contact rate between agents, γ is the probability of the recovery, and I_0 is the initial number of infectious agents. Since it is just a simple model without real data behind it, we manually set the reference parameter values and get the curves that we want to reproduce (Figure 1).

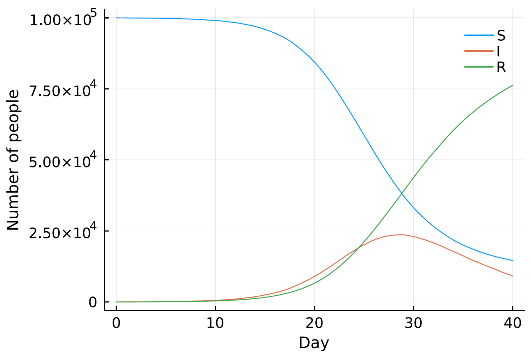


Figure 1. ABM-SIR model output showing progression of population states for susceptible (S), infected (I), and recovered (R) that we aim to reproduce by tuning parameters of the model.

4.2. Model of the Circulation of Respiratory Viruses in a City (ABM-ARI)

In order to analyse the behaviour of different parameter tuning methods on more complex models with a real application, we use the ABM, called ABM-ARI, of the co-circulation of seven respiratory viruses (influenza A and B, rhinoviruses, respiratory syncytial viruses, adenoviruses, parainfluenza and common human coronaviruses) in a city [27]. In the model, viruses differ in their properties, such as the mean viral load, duration of incubation period, duration of symptoms, and the probability of developing symptoms. Moscow was chosen as the model city. The model consists of 10 million agents representing city residents who are characterised by a set of properties. Properties such as age, sex and social status were based on the demographic and socio-economic data for various municipalities of the city according to the 2010 Russian census. Epidemiological properties include total immunoglobulin levels dependent on the age and health status of the agent. Each agent may be fully or partially susceptible to different viruses, exposed to the virus for one day, infectious with or without symptoms, or resistant to all viruses, depending on the stage of infection (Figure 2).

Agents are assigned to households, as well as other social groups, which include groups of educational institutions and workplaces. Interactions of agents with each other are modelled using contact networks in the form of complete graphs for households and Barabasi-Albert graphs [28] for groups of educational institutions and workplaces with connectivity parameters equal to 10 and 5, respectively (Figure 3). The model simulates one year, starting from August 1, and has a discrete time step of one day.

Transmission of viruses occurs through contact between fully or partially susceptible agents and infectious agents. The risk $p_i(t)$ of infection of the agent i includes the risks of infection by each simulated virus v in each simulated social setting c that agent i attends at the model step t from each infectious agent j with whom agent i makes contact:

$$p_i(t) = \Pr\{X_i(t+1) \in I | X_i(t) \in S\} = 1 - \prod_{v \in \mathcal{V}} \prod_{c \in \mathcal{C}_i} \prod_{j \in \mathcal{A}_i^{vc}} (1 - p_{j \to i}^{vc}(t)), \quad (3)$$

where $\mathcal{V}$ is the set of modelled viruses, $\mathcal{C}_i$ is the set of social settings to which the agent i belongs, $\mathcal{A}_i^{vc}$ is the set of infectious agents with whom agent i makes a contact, $X_i(t)$ is the state of the agent's health, I and S are infectious and susceptible states, respectively.

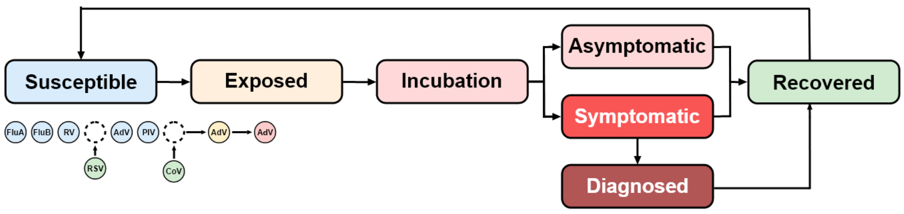


Figure 2. Flowchart for the stages of disease progression. Susceptible agents can be infected by viruses to which they do not have immunity. After getting infected agents become exposed for a day. Then, they are able to transmit the virus to others while being in an infectious state which is further subdivided into incubation, asymptomatic or symptomatic periods. During the incubation period, the viral load of an agent increases, while in asymptomatic and symptomatic periods it decreases. When agents with symptoms self-isolate at home, they are considered to be diagnosed. After recovering, they obtain immunity to the virus and do not become infected with any viruses for a short period of time.

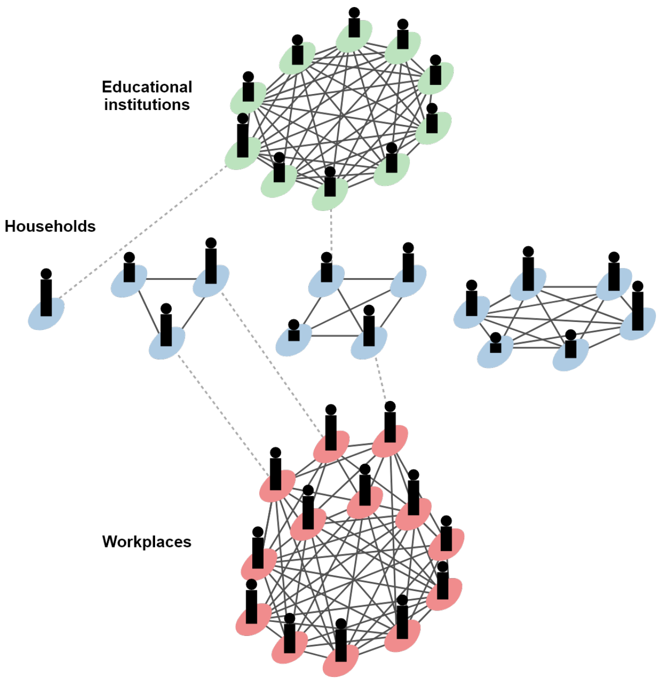


Figure 3. Contact networks for different activity settings represented as fully connected graphs for households, and Barabasi-Albert graphs for groups of educational institutions and workplaces.

The risk of virus transmission is determined by the product of five independent risks:

$$p_{j\to i}^{vc}(t) = f^v(t - t_j|j) \cdot g_1^v(i) \cdot g_2^v(t - t_i^v|i) \cdot h(t|i,j,c) \cdot k^v(t), \tag{4}$$

where f^v is the risk of transmitting the virus v from infectious agent j infected $t - t_j$ days ago, and g_1^v and g_2^v are the risks of infecting the susceptible agent i with the virus v for a given total level of immunoglobulins and a number of days $t - t_i^v$ that have passed since its last recovery from an illness caused by the virus v, respectively. h is the risk of infection transmission between two agents for a given contact duration, and k^v is the risk of transmitting the virus v for a given average daily air temperature.

The risk of transmitting the virus from an infectious agent depends on the viral load, which in turn depends on the age of the agent, the number of days passed since becoming infectious, the virus with which the agent was infected, and the presence or absence of symptoms:

$$\begin{cases} f^v(t-t_j|j) = 0, & t \le t_j \text{ or } t \ge t_j + d_{1j} + d_{2j}, \\ f^v(t-t_j|j) = \frac{l_{jv}}{2l_{\max}}(\frac{t-t_j}{d_{1j}-1} - \frac{1}{d_{1j}-1}), & t_j < t \le t_j + d_{1j} \text{ and } d_{1j} > 1, \\ f^v(t-t_j|j) = \frac{l_{jv}}{4l_{\max}}, & t_j < t \le t_j + d_{1j} \text{ and } d_{1j} = 1, \\ f^v(t-t_j|j) = \frac{l_{jv}}{l_{\max}}(\frac{t-t_j}{1-d_{2j}} - \frac{d_{1j}+d_{2j}}{1-d_{2j}}), & t_j + d_{1j} < t < t_j + d_{1j} + d_{2j}, \\ & \text{and } X_j(t) \in I_s^v, \\ f^v(t-t_j|j) = \frac{l_{jv}}{2l_{\max}}(\frac{t-t_j}{1-d_{2j}} - \frac{d_{1j}+d_{2j}}{1-d_{2j}}), & t_j + d_{1j} < t < t_j + d_{1j} + d_{2j}, \\ & \text{and } X_j(t) \in I_a^v, \end{cases} \tag{5}$$

where $l_{\max}$ and l_{jv} are the maximum and average viral load for the virus v and the age group of agent j, respectively. d_{1j} and d_{2j} are durations of the incubation period and symptoms, respectively. I_s^v and I_a^v are infectious states with symptoms and without, respectively.

The risk of infecting the susceptible agent depends on the total level of immunoglobulins, which in turn depends on the age and sex of the agent:

$$g_1^v(i) = \frac{2}{1 + \exp(\beta_v s_i)}, \tag{6}$$

where $s_i \in [0,1]$ is the normalised total immunoglobulin level of agent i, and β_v is the adjustable parameter for the virus v.

The risk of infecting the susceptible agent also depends on the level of virus-specific antibodies, which in turn depends on the number of days passed since the recovery and on the duration of the immunity to the virus:

$$\begin{cases} g_2^v(t-t_i^v|i) = 1, & t \le t_i^v \text{ or } t \ge t_i^v + r_i^v \\ g_2^v(t-t_i^v|i) = \frac{1}{r_i^v - 1}(t - t_i^v - 1), & t_i^v < t < t_i^v + r_i^v, \end{cases} \tag{7}$$

where r_i^v is the adjustable parameter denoting the duration of specific immunity of agent i to the virus.

The risk of infection transmission during the contact between two agents depends on the duration of contact, which in turn depends on the social setting where it was made:

$$h(t|i,j,c) = 1 - \exp(-\delta u_{ijc}^t), \tag{8}$$

where $u_{ijc}^t \in [0,24]$ is the duration of contact (in hours) between agents i and j in the social setting c on the step t, δ is the adjustable parameter.

The risk of transmitting the virus for a given average daily air temperature depends on the time step of the model:

$$k^v(t) = -\gamma_v \tau_t + 1, \tag{9}$$

where $\tau_t \in [0,1]$ is normalised average air temperature for the step t, and γ_v is the adjustable parameter for the virus v.

The model consists of initialisation and simulation stages (Figure 4). Simulation stage consists of multiple model steps where each step goes as follows:

1. First, we determine whether the current day is a holiday or a day off for different social settings. National holidays and Sundays are days off for everyone. Children have summer, autumn, spring, and winter vacations. Workers do not work on Saturdays.
2. We iterate over the agents, and if we find an infectious agent, we iterate over the agents with whom the agent makes contact with the current step. In the case of finding an agent, susceptible or partially susceptible to the virus, we sample the duration of their

contact from a given distribution [29] and evaluate the risk of infection transmission. If the transmission is successful, the agent becomes exposed.

3. Next, we update the agents' properties based on their health state:
   (a) Susceptible. There is a small chance that the agent may have been exposed to a virus from an unknown source at the current step. The virus is selected randomly, and if it is able to overcome the level of specific immunity, the agent becomes exposed.
   (b) Immune to at least one of the viruses. We decrease the level of specific immunity to the virus.
   (c) Infected at the current step. We find the duration of the incubation period and the duration of symptoms or the duration of the asymptomatic period from the given distribution [30–32].
   (d) Infectious. The agent can exhibit symptoms or progress to the asymptomatic stage after the end of the incubation period, self-isolate and become diagnosed [33], or recover.
   (e) Recovered. We check if it is able to be infected with viruses again.
4. In the end, we update the model date and air temperature.

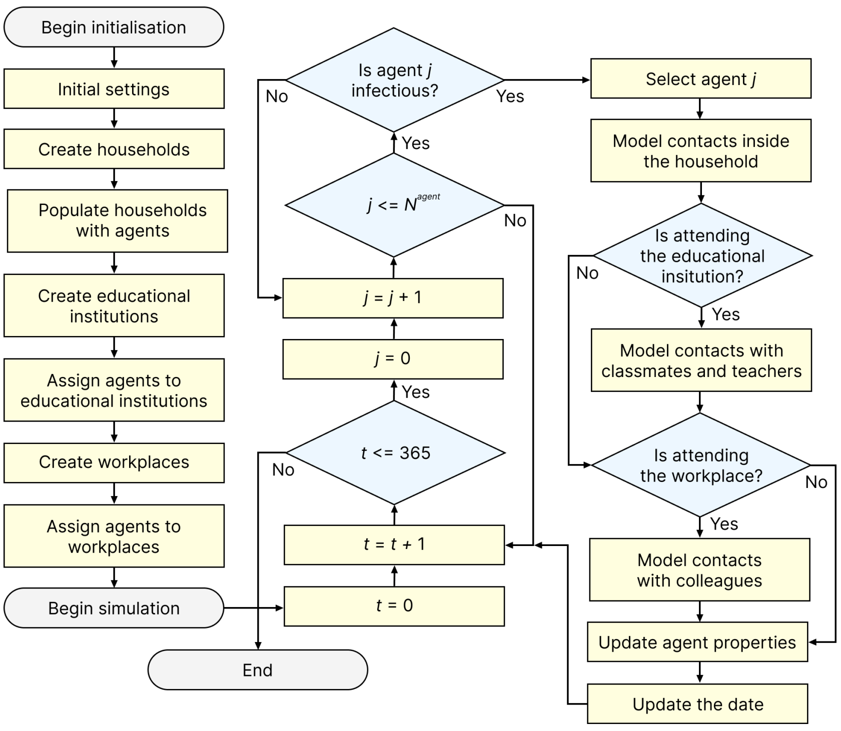

Begin initialisation
Initial settings
Create households
Populate households with agents
Create educational institutions
Assign agents to educational institutions
Create workplaces
Assign agents to workplaces
Begin simulation
End
Is agent j infectious?
No
Yes
j <= N^agent
No
j = j + 1
j = 0
Yes
t <= 365
No
t = t + 1
t = 0
Select agent j
Model contacts inside the household
Is attending the educational insitution?
No
Yes
Model contacts with classmates and teachers
Is attending the workplace?
No
Yes
Model contacts with colleagues
Update agent properties
Update the date

Figure 4. Process algorithm of ABM-ARI.

The model has 26 adjustable parameters $\Theta = \{\theta_i\}_{i=1}^{26}$ divided into 5 groups: a group corresponding to the dependence of the risk of infection on the duration of contact (δ), a group corresponding to the dependence of the risk of infection on the total level of immunoglobulins (β_v), a group corresponding to the dependence of the risk of infection on the air temperature (γ_v), a group corresponding to the average durations of immunity to each virus (r_v), where $v = 1, \ldots, 7$ is the virus, and a group corresponding to the probabilities of infection from an unknown source (p_a), where $a = 1, \ldots, 4$ is the age group.

The model is used to reproduce the average weekly incidence of acute respiratory infections in Moscow for different age groups and viruses for a single year based on the data from 1997 to 2002. It consists of three groups of incidence curves: an overall incidence curve (Figure 5), incidence curves for age groups 0–2, 3–6, 7–14 and 15+ years, and incidence curves for seven modelled viruses.

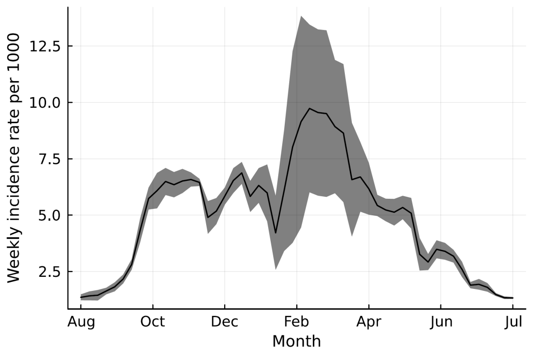


Figure 5. Weekly incidence of acute respiratory infections in Moscow averaged over the period 1997–2002. The 95% confidence intervals are represented as shaded areas.

5. Results

In this section, we compare the simulation results produced by the following algorithms: the Markov chain Monte Carlo method (MCMC), the surrogate modelling approach (SM), the particle swarm optimisation (PSO) algorithm, the genetic algorithm (GA), and the chaos game optimisation algorithm (CGO). The proposed methods are tested on two agent-based models (ABMs) of different complexity: simple ABM for simulating the spread of contagious disease where each agent has three states: susceptible, infectious, and recovered (ABM-SIR), and more complex ABM with a real application that simulates the spread of acute respiratory infections in a city (ABM-ARI). We manually set the hyperparameter values for the proposed algorithms, which are set to be the same for both models (Table 3). We set the number of iterations for each method to be equal to 200.

Table 3. Hyperparameters of the parameter tuning algorithms: Markov chain Monte Carlo method (MCMC), surrogate modelling approach (SM), particle swarm optimisation (PSO) algorithm, genetic algorithm (GA), and chaos game optimisation (CGO) algorithm. The maximum number of iterations for each algorithm is 200.

Method	**Parameter**	**Description**	**Value**
MCMC	σ	Standard deviation for the parameter candidates	0.1
SM	N^{train}	Number of initial training samples	1000
	η	Learning rate	0.1
	m^{depth}	Maximum depth of a tree	10
	n^{round}	Number of rounds for boosting	150
PSO	$N^{particle}$	Number of particles	10
	w_{min}	Minimum inertia weight	0.4
	w_{max}	Maximum inertia weight	0.9
	$c1$	Personal acceleration coefficient	2.0
	$c2$	Social acceleration coefficient	2.0
GA	N^{pop}	Size of population	10
	N^{parent}	Number of parents	5
	k^{tour}	Tournament size	2
	p^{cross}	Probability of crossover	0.8
	p^{mut}	Probability of mutation	0.15
	σ	Mutation deviation coefficient	0.33
CGO	N^{seed}	Number of seeds	10

For the simulation experiments, we use the Julia programming language with the Threads package for parallel computing. We run the models on a Windows 10 machine with an Intel Core i5-7300HQ quad-core processor and 16 GB of RAM.

5.1. ABM-SIR Model Results

ABM-SIR has four parameters that we want to estimate in order to reproduce the defined dynamics of the spread of contagious disease in a population of 100 thousand agents. Table 4 describes each parameter, their constraints, and the set of reference values that we manually set ourselves.

Table 4. ABM-SIR model parameters, their ranges and the reference set of parameter values that we want to reproduce.

Parameter	Description	Range	Reference
β	Probability of the transmission	[0.02, 0.2]	0.05
c	Contact rate	[5, 25]	10
γ	Probability of recovery	[0.01, 0.05]	0.02
I_0	Initial number of infectious agents	[1, 50]	25

As the criterion of comparison between different methods, we use root mean square error ($RMSE$). For the ABM-SIR model, we use:

$$RMSE = \sqrt{\frac{1}{1200}\sum_i \left((S_i^d - S_i^m)^2 + (I_i^d - I_i^m)^2 + (R_i^d - R_i^m)^2\right)}, \tag{10}$$

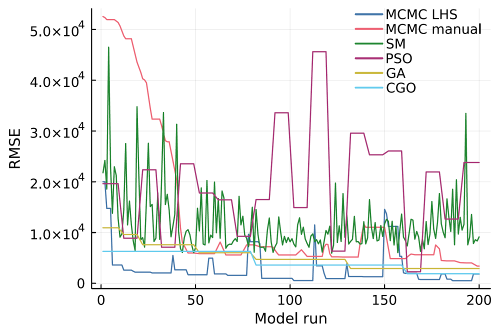
where S_i^d and S_i^m are the number of susceptible agents in the step i based on the reference data and model simulation, I_i^d and I_i^m are the number of infectious agents in the step i based on the reference data and model simulation, and R_i^d and R_i^m are the number of recovered agents in the step i based on the reference data and model simulation. The model simulates 400 steps, with one step being 0.1. Initially, there are no recovered agents.

Initial parameter values are set by either using the Latin hypercube sampling method (LHS) or manually. For MCMC, we test both LHS with 10 initial samples and manual selection by selecting a random point in the parameter space. For other methods, we only use 10 initial samples of parameter values obtained from LHS. The results of the model calibration using different methods with 9 algorithm runs and 200 model runs in each of the run are presented in Figure 6 and Table 5. In general, MCMC produced better solutions compared to the other methods. While SM was able to reach its best value on average under 100 steps, it had the worst average performance out of all methods and also took the most time to execute. PSO, GA, and CGO showed similar results with no particular winner. For the best run, all of the methods were very close to the reference curves.

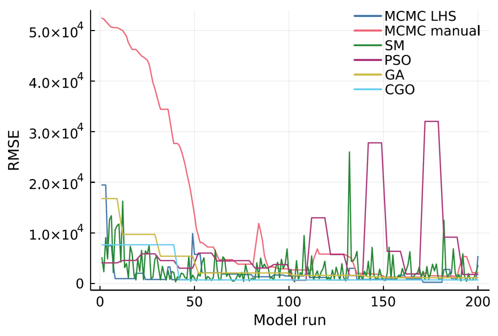
Table 5. The comparison of different methods for ABM-SIR model using the root mean square error ($RMSE$) based on 200 iterations for each algorithm over 9 algorithm runs.

Method *	Avg Initial *RMSE*	Avg Best *RMSE*	Best *RMSE*	Avg Best Decrease	Best Decrease	Avg Best Run	Best Run **	Avg Time ***
MCMC LHS	17,868	791	243	96%	99%	148	171	169 s
MCMC manual	52,899	3342	1248	94%	98%	171	150	158 s
SM	17,842	7728	310	57%	98%	96	103	171 s
PSO	16,163	2360	627	85%	96%	121	101	100 s
GA	17,475	3175	1140	82%	93%	164	181	115 s
CGO	14,219	1614	704	89%	95%	27	11	105 s

* Markov chain Monte Carlo method with Latin hypercube sampling for initial parameter values selection (MCMC LHS) or with manually selected initial parameter values (MCMC manual), surrogate modelling approach (SM), particle swarm optimisation (PSO) method, genetic algorithm (GA), and chaos game optimisation (CGO) algorithm. ** Best run is the number of iterations that reached the lowest value of $RMSE$ among all runs. *** Avg time is the average time needed for one algorithm run.



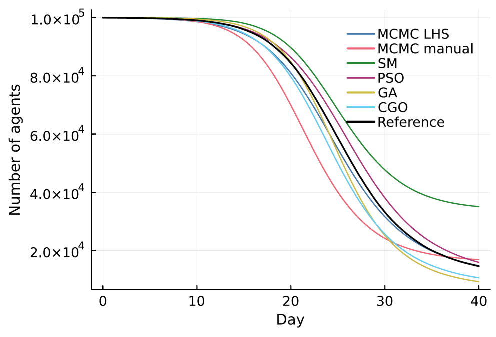
(**a**) Median



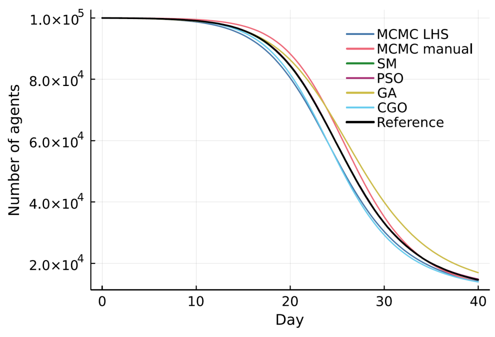
(**b**) Best

Figure 6. Rate of convergence of different parameter tuning algorithms applied to the ABM-SIR model using the root mean square error ($RMSE$) based on 9 runs of the following methods: Markov chain Monte Carlo method (MCMC) with Latin hypercube sampling method (LHS) used for selecting initial parameter values or with them being selected manually (manual), surrogate modelling approach (SM), particle swarm optimisation (PSO) method, genetic algorithm (GA), and chaos game optimisation (CGO) algorithm.

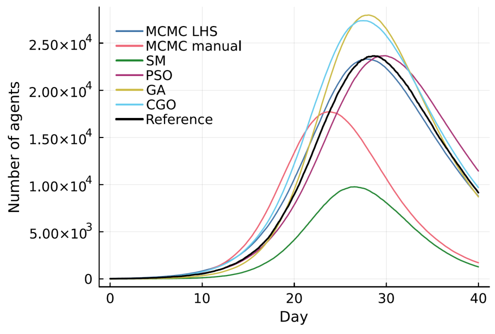
Figure 7 shows three curves indicating the dynamics of the number of susceptible, infectious, and recovered agents by using the best parameter values obtained from the model calibration procedures. From the figures, we can see that MCMC with LHS is the closest to the reference curves with PSO being second.



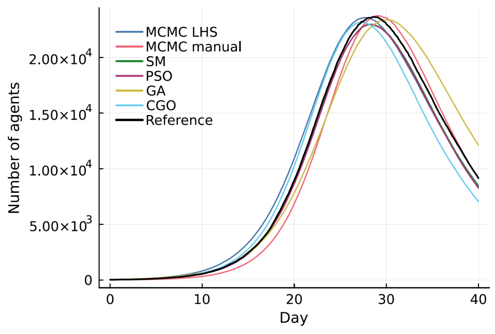
(**a**) Median Susceptible (S)



(**b**) Best Susceptible (S)

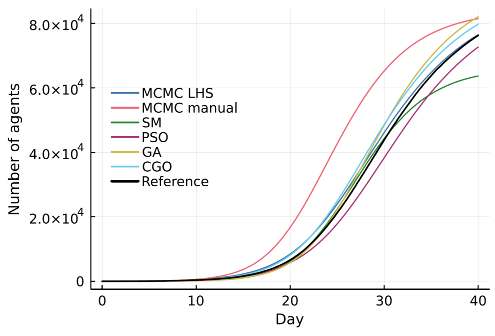


(**c**) Median Infectious (I)

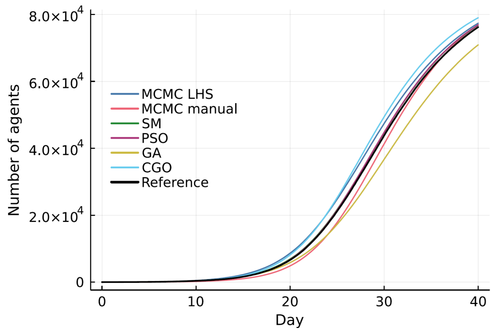


(**d**) Best Infectious (I)

Figure 7. *Cont.*



(**e**) Median Recovered (R)



(**f**) Best Recovered (R)

Figure 7. Simulation results of toy ABM-SIR model compared to the author-defined reference data using best parameter values obtained from the following algorithms: Markov chain Monte Carlo method (MCMC) with Latin hypercube sampling (LHS) or with manual initial parameter values selection, surrogate modelling approach (SM), particle swarm optimisation (PSO) method, genetic algorithm (GA), and chaos game optimisation (CGO) algorithm. Results are based on 9 runs of each method of 200 model runs each.

5.2. ABM-ARI Model Results

ABM-ARI has 144 parameters, with 26 of them being estimated to reproduce the seasonal increase in the incidence of acute respiratory infections throughout a year in a city with a population of 10 million residents. Simulations start on August 1, which corresponds to the last month of summer vacation for children. The model has a time step of one day. Table 6 describes every calibrated parameter of the model in detail. For model evaluation, the data on the average weekly number of diagnosed cases of acute respiratory infections in Moscow over the period 1997–2002 [34] are used together with the data on the distribution of viruses throughout the year for the dynamics of detected cases of different respiratory infections by PCR in Russia over from 2014 to 2016 [35].

Table 6. ABM-ARI model parameters, corresponding group and their ranges.

Parameter	Description	Group *	Range
c	Influence of the duration of contact	All	[0.1, 1]
β_1	Susceptibility parameters	FluA	[1, 7]
β_2		FluB	
β_3		RV	
β_4		RSV	
β_5		AdV	
β_6		PIV	
β_7		CoV	
γ_1	Temperature parameters	FluA	[0.01, 1]
γ_2		FluB	
γ_3		RV	
γ_4		RSV	
γ_5		AdV	
γ_6		PIV	
γ_7		CoV	
r_1	Mean immunity durations	FluA	[30, 365]
r_2		FluB	
r_3		RV	
r_4		RSV	
r_5		AdV	
r_6		PIV	
r_7		CoV	

Table 6. *Cont.*

Parameter	Description	Group *	Range
p_1	Probabilities of	0–2 y.o.	[0.0008, 0.0012]
p_2	infections from	3–6 y.o.	[0.0005, 0.001]
p_3	unknown	7–14 y.o.	[0.0002, 0.0005]
p_4	sources	15+ y.o.	[0.000005, 0.00001]

* Influenza A (FluA), influenza B (FluB), rhinovirus (RV), respiratory syncytial virus (RSV), adenovirus (AdV), parainfluenza (PIV) and common human coronavirus (CoV).

For the comparison between different methods, we use root mean square error ($RMSE$):

$$RMSE = \sqrt{\frac{1}{1456}\sum_{w=1}^{52}\sum_{a=1}^{4}\sum_{v=1}^{7}(y_{wav}^{d} - y_{wav}^{m})^2}, \quad (11)$$

where y_{wav}^{d} and y_{wav}^{m} are the number of diagnosed cases of infections with the virus v in the week w in the age group a according to the data and the results of a numerical experiment.

Initial parameter values are set by either using the Latin hypercube sampling method (LHS) or manually. For MCMC, we test both LHS with 1000 initial samples and manual selection by selecting a point in the parameter space based on domain knowledge. For SM, we use 1000 initial samples of parameter values obtained from LHS. For PSO, GA, and CGO we use LHS with the number of samples equal to the number of particles, the size of the population, and the number of seeds, respectively, that we set equal to 10. The results of the model calibration using different methods with 3 runs and 200 iterations in each method are presented in Figure 8 and Table 7. We can see that MCMC produced the best $RMSE$ due to a good initial parameter selection in the case of a manual selection or using a large number of numerical experiments in the case of using LHS. SM takes the most time to execute due to the large number of numerical experiments required for creating the initial training set. In the case of a small number of initial samples (10), it demonstrates much worse performance. SM provided the second-best $RMSE$. PSO, GA, and CGO all showed similar performance. While CGO is the fastest method to achieve its best $RMSE$, it stagnates at a local minimum afterwards. PSO has strong fluctuations from iteration to iteration opposite to GA, which decreases more steadily.

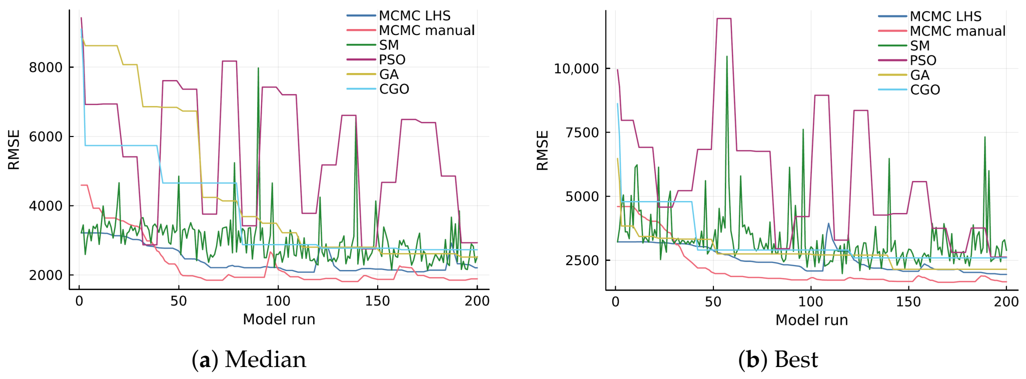


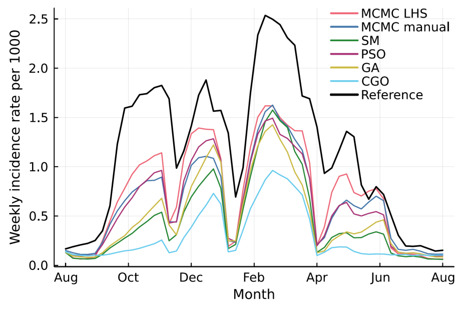
Figure 8. Rate of convergence of different parameter tuning algorithms applied to the ABM-ARI model using the root mean square error ($RMSE$) based on 3 runs of the following methods: Markov chain Monte Carlo method (MCMC) with Latin hypercube sampling method (LHS) used for selecting initial parameter values or with them being selected manually (manual), surrogate modelling approach (SM), particle swarm optimisation (PSO) method, genetic algorithm (GA), and chaos game optimisation (CGO) algorithm.

Table 7. The comparison of different methods for ABM-ARI model using the root mean square error ($RMSE$) based on 200 iterations for each algorithm over 3 algorithm runs.

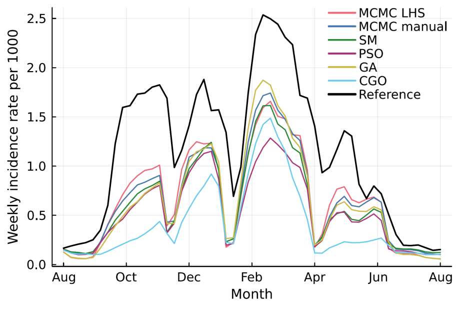
Method *	Avg Initial *RMSE*	Avg Best *RMSE*	Best *RMSE*	Avg Best Decrease	Best Decrease	Avg Best Run	Best Run **	Avg Time ***
MCMC LHS	3214	2104	1945	35%	39%	167	196	≈60 h
MCMC manual	4580	1757	1639	62%	64%	145	164	≈9.5 h
SM	3214	2093	1978	35%	38%	146	116	≈60.5 h
PSO	11,927	2889	2622	76%	78%	111	191	≈10 h
GA	9120	2428	2147	73%	76%	164	141	≈10 h
CGO	12,460	2813	2592	77%	79%	38	31	≈10 h

* Markov chain Monte Carlo method with Latin hypercube sampling for initial parameter values selection (MCMC LHS) or with manually selected initial parameter values (MCMC manual), surrogate modelling approach (SM), particle swarm optimisation (PSO) method, genetic algorithm (GA), and chaos game optimisation (CGO) algorithm. ** Best run is the number of iteration that reached the lowest value of $RMSE$ among all runs. *** Avg time is the average time needed for one algorithm run including LHS procedure.

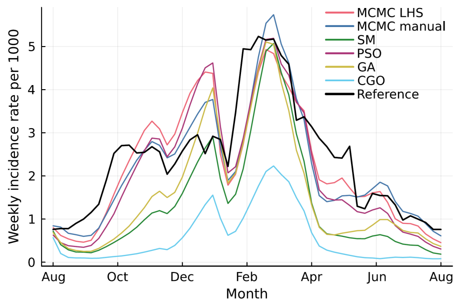
Figure 9 shows two curves indicating the incidence of acute respiratory infections in two age groups: the 7–14-year-old group and the over-15-year-old group. We only show the results for two age groups since they represent 94% of the model population and show the impact of different model calibration methods the best. The incidence for other age groups and specific viruses can be found at [27]. From the figures, we can see that most methods underestimated the incidence for the 7–14-year-old group and overestimated the incidence for the over-15-year-old group.



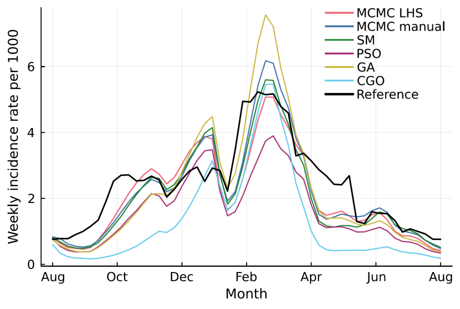
(**a**) Median 7–14-year-old



(**b**) Best 7–14-year-old



(**c**) Median over-15-year-old



(**d**) Best over-15-year-old

Figure 9. Simulation results of ABM-ARI for two age groups compared to the real data using best parameter values obtained from the following algorithms: Markov chain Monte Carlo method (MCMC) with Latin hypercube sampling (LHS) or with manual initial parameter values selection, surrogate modelling approach (SM), particle swarm optimisation (PSO) method, genetic algorithm (GA), and chaos game optimisation (CGO) algorithm. Results are based on 3 runs of each method of 200 model runs each.

6. Discussion

The aim of this work was to broaden the understanding of how classical metaheuristic algorithms behave when applied to a more complex agent-based model with a number of parameters of more than 20 and a number of agents being around 10 million, and to compare the findings to a simple model. Due to computational limitations associated with ABMs, most of the work conducted in this field focuses on simpler models with a number of parameters often lower than 10 [4,36,37]. While we use well-established methods for the model calibration, we implement our own versions of those methods, in particular our implementation of the Markov chain Monte Carlo method.

We tested five different approaches for solving the model calibration problem using two ABMs of different complexity. Our results suggest that the Markov chain Monte Carlo method (MCMC) with the usage of the loss function is the most promising method of parameter tuning. Unlike the surrogate modelling approach, MCMC algorithms do not require conducting a large number of numerical experiments before using the algorithm. It is possible to set an initial set of parameter values for the algorithm based on some domain knowledge, and it can reach acceptable values of the loss function pretty quickly even if the initial guess is not that good. Another advantage of MCMC is the ease of setting the hyperparameters of the algorithm, since they are only needed for generating the candidates from the distribution.

One of the challenges of using surrogate modelling for ABMs is that they have a strong dependence on the machine learning algorithm used for the training. Even though they significantly reduce the computational efforts of predicting the outcomes of ABM, there is still a computational cost that lies in their training process. Neural networks, for example, can take a long time to train from scratch. Another problem is that they require extensive parameter space exploration before they can start making accurate predictions. One of the benefits of using surrogate modelling is the ability to easily assess the relative importance of model parameters.

The particle swarm optimisation (PSO), genetic algorithm (GA), and chaos game optimisation (CGO) methods showed similar levels of performance. While they perform very well for a simple ABM, they struggle with local minima when the parameter space becomes large. For both PSO and GA, it takes a large number of model runs to reach acceptable values of the loss function. One of the challenges of using PSO is that particles tend to go out of the parameter boundaries creating a need for tackling this problem. PSO also showed stronger fluctuations of the error function occurring because of choosing values for both of the acceleration coefficients equal to 2, which is commonly used in the literature. CGO was able to reach its best value in the least amount of steps, but it stagnated afterwards while being stuck in a local minimum.

While this work provides the comparison of commonly used metaheuristic algorithms applied to one simple toy ABM and one complex ABM with a real application, further work can be used to expand on the results of this study. For example, the SM presented in this article uses the XGBoost algorithm and a single output in the form of root mean square error. It is possible to analyse the influence of other machine learning methods and the impact of the multiple outputs on the problem. Another area is hyperparameter tuning. We also did not consider other swarm intelligence algorithms, such as the artificial bee colony algorithm or the firefly algorithm. Another problem is the stochasticity of the proposed methods; for example, the ABM-SIR model shows that the average best error and the best error can differ significantly; therefore, there is a need for multiple runs of the same method in order to fully analyse its performance.

Author Contributions: Conceptualisation, A.I.V. and A.A.R.; funding acquisition, A.I.V. and A.A.R.; investigation, A.I.V.; resources, A.I.V., A.A.R. and T.E.S.; data curation, A.I.V., A.A.R. and T.E.S.; methodology, A.I.V.; software, A.I.V.; supervision, A.A.R. and T.E.S.; writing, A.I.V. All authors have read and agreed to the published version of the manuscript.

Funding: This work was carried out with the financial support of the Moscow Center of Fundamental and Applied Mathematics at INM RAS (Agreement with the Ministry of Education and Science of the Russian Federation No.075-15-2022-286).

Data Availability Statement: The original data presented in the study are openly available at https://github.com/Firkhraag/JuliaABM (accessed on 28 June 2024).

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ABM	Agent-based model
MCMC	Markov chain Monte Carlo method
SM	Surrogate modelling method
PSO	Particle swarm optimisation algorithm
GA	Genetic algorithm
CGO	Chaos game optimisation
LHS	Latin hypercube sampling
RMSE	Root mean square error

References

1. Macal, C.M.; North, M.J. Tutorial on agent-based modeling and simulation. In Proceedings of the Winter Simulation Conference, Orlando, FL, USA, 4–7 December 2005; pp. 2–15. [CrossRef]
2. Macal, C.M. Everything you need to know about agent-based modelling and simulation. *J. Simul.* **2016**, *10*, 144–156. [CrossRef]
3. Kotthoff, F.; Hamacher, T. Calibrating agent-based models of innovation diffusion with gradients. *J. Artif. Soc. Soc. Simul.* **2022**, *25*, 4. [CrossRef]
4. Carrella, E. No free lunch when estimating simulation parameters. *J. Artif. Soc. Soc. Simul.* **2021**, *24*, 7. [CrossRef]
5. Hussain, K.; Mohd Salleh, M.N.; Cheng, S.; Shi, Y. Metaheuristic research: A comprehensive survey. *Artif. Intell. Rev.* **2019**, *52*, 2191–2233. [CrossRef]
6. Gandomi, A.H.; Yang, X.S.; Talatahari, S.; Alavi, A.H. Metaheuristic algorithms in modeling and optimization. *Metaheuristic Appl. Struct. Infrastruct.* **2013**, *1*, 1–24. [CrossRef]
7. Abdel-Basset, M.; Abdel-Fatah, L.; Sangaiah, A.K. Metaheuristic algorithms: A comprehensive review. In *Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Applications*; Elsevier: Amsterdam, The Netherlands, 2018; pp. 185–231. [CrossRef]
8. Hare, W.; Nutini, J.; Tesfamariam, S. A survey of non-gradient optimization methods in structural engineering. *Adv. Eng. Softw.* **2013**, *59*, 19–28. [CrossRef]
9. Gilli, M.; Winker, P. A global optimization heuristic for estimating agent based models. *Comput. Stat. Data Anal.* **2003**, *42*, 299–312. [CrossRef]
10. Chen, S.; Desiderio, S. A regression-based calibration method for agent-based models. *Comput. Econ.* **2022**, *59*, 687–700. [CrossRef]
11. Lamperti, F.; Roventini, A.; Sani, A. Agent-Based Model Calibration using Machine Learning Surrogates. *J. Econ. Dyn. Control* **2018**, *90*, 366–389. [CrossRef]
12. Sallans, B.; Pfister, A.; Karatzoglou, A.; Dorffner, G. Simulation and Validation of an Integrated Markets Model. *J. Artif. Soc. Soc. Simul.* **2003**, *6*. Available online: https://jasss.soc.surrey.ac.uk/6/4/2.html (accessed on 12 July 2024).
13. Merler, S.; Ajelli, M.; Fumanelli, L.; Gomes, M.F.; Piontti, A.P.; Rossi, L.; Chao, D.L.; Longini, I.M.; Halloran, M.E.; Vespignani, A. Spatio-temporal spread of the Ebola 2014 outbreak in Liberia and the effectiveness of non-pharmaceutical interventions: A computational modelling analysis. *Lancet Infect. Dis.* **2018**, *15*, 204–211. [CrossRef] [PubMed]
14. Tan, R.K.; Bora, S. Adaptive parameter tuning for agent-based modeling and simulation. *SIMULATION Trans. Soc. Model. Simul. Int.* **2019**, *95*, 003754971984636. [CrossRef]
15. Zhang, Y.; Li, Z.; Zhang, Y. Validation and calibration of an agent-based model: A surrogate approach. *Discret. Dyn. Nat. Soc.* **2020**, *2020*, 6946370. [CrossRef]
16. Perumal, R.; van Zyl, T.L. Surrogate Assisted Methods for the Parameterisation of Agent-Based Models. In Proceedings of the 2020 7th International Conference on Soft Computing & Machine Intelligence (ISCMI), Stockholm, Sweden, 14–15 November 2020; pp. 78–82. [CrossRef]
17. Angione, C.; Silverman, E.; Yaneske, E. Using machine learning as a surrogate model for agent-based simulations. *PLoS ONE* **2022**, *17*, e0263150. [CrossRef] [PubMed]
18. Calvez, B.; Hutzler, G. Automatic tuning of agent-based models using genetic algorithms. In Proceedings of the International Workshop on Multi-Agent Systems and Agent-Based Simulation, Utrecht, The Netherlands, 25 July 2005; Springer: Berlin/Heidelberg, Germany, 2005, pp. 41–57. [CrossRef]

19. Heppenstall, A.J.; Evans, A.J.; Birkin, M.H. Genetic algorithm optimisation of an agent-based model for simulating a retail market. *Environ. Plan. B Urban Anal. City Sci.* **2007**, *34*, 1051–1070. [CrossRef]
20. McKay, M.D.; Beckman, R.J.; Conover, W.J. Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code. *Technometrics* **1979**, *21*, 239–245. [CrossRef]
21. Brooks, S.P. Markov Chain Monte Carlo Method and Its Application. *J. R. Stat. Soc. Ser. D Stat.* **1998**, *47*, 69–100. [CrossRef]
22. Chen, T.; Guestrin, C. XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 13–17 August 2016; KDD '16; pp. 785–794.
23. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICNN'95—International Conference on Neural Networks, Perth, Australia, 27 November–1 December 1995; Volume 4, pp. 1942–1948. [CrossRef]
24. Holland, J.H. Genetic algorithms. *Sci. Am.* **1992**, *267*, 66–73. [CrossRef]
25. Talatahari, S.; Azizi, M. Chaos game optimization: A novel metaheuristic algorithm. *Artif. Intell. Rev.* **2021**, *54*, 917–1004. [CrossRef]
26. Kermack, W.O.; McKendrick, A.G. A Contribution to the Mathematical Theory of Epidemics. *Proc. R. Soc. Lond. Ser. A Contain. Pap. Math. Phys. Character* **1927**, *115*, 700–721.
27. Vlad, A.I.; Romanyukha, A.A.; Sannikova, T.E. Circulation of Respiratory Viruses in the City: Towards an Agent-Based Ecosystem model. *Bull. Math. Biol.* **2023**, *85*, 100. [CrossRef]
28. Albert, R.; Barabasi, A.L. Statistical mechanics of complex networks. *Rev. Mod. Phys.* **2002**, *74*, 47–97. [CrossRef]
29. Mossong, J.; Hens, N.; Jit, M.; Beutels, P.; Auranen, K.; Mikolajczyk, R.; Massari, M.; Salmaso, S.; Tomba, G.S.; Wallinga, J.; et al. Social Contacts and Mixing Patterns Relevant to the Spread of Infectious Diseases. *PLoS Med.* **2008**, *5*, e74. [CrossRef]
30. Lessler, J.; Reich, N.G.; Brookmeyer, R.; Perl, T.M.; Nelson, K.E.; Cummings, D.A.T. Incubation periods of acute respiratory viral infections: A systematic review. *Lancet Infect. Dis.* **2009**, *9*, 291–300. [CrossRef]
31. Carrat, F.; Vergu, E.; Ferguson, N.M.; Lemaitre, M.; Cauchemez, S.; Leach, S.; Valleron, A.J. Time Lines of Infection and Disease in Human Influenza: A Review of Volunteer Challenge Studies. *Am. J. Epidemiol.* **2008**, *167*, 775–785. [CrossRef]
32. Warrell, D.A.; Cox, T.M.; Firth, J.D.; Torok, E. *Oxford Textbook of Medicine: Infection*; Oxford University Press: Oxford, UK, 2012.
33. Elveback, L.R.; Fox, J.P.; Ackerman, E.; Langworthy, A.; Boyd, M.; Gatewood, L. An influenza simulation model for immunization studies. *Am. J. Epidemiol.* **1976**, *103*, 152–165. [CrossRef] [PubMed]
34. Romanyukha, A.A.; Sannikova, T.E.; Drynov, I.D. The origin of acute respiratory epidemics. *Her. Russ. Acad. Sci.* **2011**, *81*, 31–34. [CrossRef] [PubMed]
35. Karpova, L.S.; Volik, K.M.; Smorodintseva, E.A.; Stolyarova, T.P.; Popovtseva, N.M.; Stolyarov, K.A. The Impact of Influenza of Different Etiologies on other ARVI in Children and Adults in 2014 to 2016. *Epidemiol. Vaccinal Prev.* **2018**, *17*, 35–47. [CrossRef]
36. Platt, D. A comparison of economic agent-based model calibration methods. *J. Econ. Dyn. Control* **2020**, *113*, 103859. [CrossRef]
37. Dyer, J.; Cannon, P.; Farmer, J.D.; Schmon, S.M. Black-box Bayesian inference for agent-based models. *J. Econ. Dyn. Control* **2024**, *161*, 104827. [CrossRef]