Immune System Model Calibration by Genetic Algorithm

Presbitero, A.; Krzhizhanovskaya, V.; Mancini, E.; Brands, R.; Sloot, P.

DOI
10.1016/j.procs.2016.11.020
Publication date
2016
Document Version
Final published version
Published in
Procedia Computer Science
License
CC BY-NC-ND

Procedia Computer Science

Procedia Computer Science 101, 2016, Pages 161 – 171

YSC 2016. 5th International Young Scientist Conference on Computational Science

Immune System Model Calibration by Genetic Algorithm

Alva Presbitero[1,2*], Valeria Krzhizhanovskaya[1,2,3], Emiliano Mancini[2], Ruud Brands[4], and Peter Sloot[1,2,4]

[1]*ITMO University, St. Petersburg.*
[2]*University of Amsterdam, The Netherlands*
[3]*Saint Petersburg Polytechnic University, Russia*
[4]*Nanyang Technological University, Singapore*
[*]*avpresbitero@gmail.com, V.Krzhizhanovskaya@uva.nl, mancini.emiliano76@gmail.com, ruudrbrands@ntu.edu.sg, p.m.a.sloot@uva.nl*

Abstract
We aim to develop a mathematical model of the human immune system for advanced individualized healthcare where medication plan is fine-tuned to fit a patient's conditions through monitored biochemical processes. One of the challenges is calibrating model parameters to satisfy existing experimental data or prior knowledge about the system behavior. In this paper, we apply genetic algorithm to find model parameters reproducing the results of modeling human innate immune system by Pigozzo et al.

Keywords: Innate Immune System Model, Mathematical Modeling, Reproducibility, Genetic Algorithm

1 Introduction

The human immune system is a complex, yet delicate network of highly specialized cells, tissues, and organs that work together to defend the human body from various insults that could potentially lead to serious diseases, and in some cases, even death. Hence, a deep and thorough understanding of the human immune system is only necessary. Mathematical and computational modeling, such as differential equations [1, 2], agent-based modeling [3] and cellular automata [4] provide ways to understand and analyze underlying biochemical processes and dynamics of key players in the immune system while concurrently being able to fine-tune immunological parameters along the way.

Science advances by building on top of published results, hence corroboration is only necessary especially now that a mere 11% of published research is reproducible [5]. As an initial step towards modeling the human innate immune system, we look into a simplified model developed by Pigozzo et al. In a recent paper [6], we found that the work of [1] is not reproducible when parameters that are indicated in their paper are used. In the current work, we calibrate these parameter values using genetic algorithm by limiting these values within specific ranges that are biologically acceptable.

The article is structured as follows: first, the differential equations used by [1] will be presented and explained in section 2. We present the numerical methods used in obtaining the authors' results and implementation together with the validation of the genetic algorithm in section 3. This is followed by the presentation and discussion of our results in section 4. Finally, our conclusion will be presented in section 5.

Peer-review under responsibility of organizing committee of the scientific committee of the 5th International Young Scientist Conference on Computational Science

doi:10.1016/j.procs.2016.11.020

2 A Reduced Model of the Immune System Using Partial Differential Equations by Pigozzo et al.

Mathematical modeling of the human innate immune system through the use of partial differential equations provides a way to observe and analyze the spatial and temporal dynamics of players in the human innate immune system.

The human innate immune system is a complex system that acts as a non-specific defense mechanism of the human body against infection, which commonly manifests in the form of an inflammation. Local inflammation is often triggered by lipopolysaccharides (LPS) – a bacterial toxin that behaves as a potent immunostimulant, which triggers an acute inflammatory response. This event subsequently activates the resting macrophages to activated macrophages. The activated macrophages then secrete cytokines that leads to the opening of the endothelial barrier. Neutrophils circulating in the bloodstream enter the tissue where they empty out their granules to resolve the inflammation. If the inflammation is clear, the neutrophils go into apoptosis, or programmed death. Activated macrophages then clear them out and go back to rest. This induces an anti-inflammatory effect, which shuts down the inflammation. In the case where the insult is not resolved, the neutrophils go into the necrotic state, which leads to a massive release of cellular content moieties. This triggers even more inflammation inducing more neutrophils to enter the tissue.

Authors Pigozzo et al. focused on three key players in the human innate immune system namely: LPS, neutrophils, and cytokines. In order to model the spatial and temporal dynamics of LPS (A), neutrophils (N), and pro-inflammatory cytokines (CH), the authors utilized three partial differential equations shown in (1) to (3).

Equation (1) models the behavior of LPS in one dimension. The term $\mu_A A$ represents the decay of LPS through time, $\lambda_{N|A} A.N$ is for the phagocytosis, a biological term referring to the process of engulfing a solid substance, of LPS by neutrophils. An initial concentration of LPS designated by A_0 fills the space x from 0 to 1. Neumann boundary conditions are also utilized to represent zero flux of A at the boundary ($\partial\Omega$) of the domain (Ω).

$$\begin{cases} \dfrac{\partial A}{\partial t} = -\mu_A A - \lambda_{N|A} A \cdot N + D_A \Delta A \\ A(x,0) = A_0 \,\Big|\, 0 \le x < 1, \qquad \dfrac{\partial A}{\partial n}\Big| \partial\Omega = 0 \end{cases} \tag{1}$$

The dynamics of neutrophils are modeled by equation (2) where initial concentration of neutrophils is set to N_0. Neumann boundary condition is once again utilized to designate a zero flux of N at the boundary. The terms $permeability_N$ and consequently $source_N$ model how endothelium permeability depends on the local concentration of pro-inflammatory cytokines. The term $\mu_N N$ represents apoptosis, a biological term, which means programmed cell death, of neutrophils, $D_N \Delta N$ is for the diffusion of neutrophils, and $\nabla.(\chi_N N \nabla \mathrm{CH})$ is for chemotaxis, which models the neutrophil movement due to the chemical stimulus induced by the presence of pro-inflammatory cytokines.

$$\begin{cases} \dfrac{dN}{dt} = -\mu_N N - \lambda_{A|N} A \cdot N + D_N \Delta N + source_N - \nabla \cdot (\chi_N N \nabla \mathrm{CH}) \\ N(x,0) = N_0, \qquad \dfrac{\partial N}{\partial n}\Big| \partial\Omega = 0 \\ permeability_N = \left(P_N^{max} - P_N^{min}\right) \dfrac{CH}{CH + keqch} + P_N^{min} \\ source_N = permeability_N (N^{max} - N) \end{cases} \tag{2}$$

Finally, the dynamics of cytokines are represented by equation 3. Initial concentration of cytokines at $t = 0$ is zero. Neumann boundary conditions are also utilized to designate a zero flux of pro-inflammatory cytokines at the boundary of the domain. The term $\mu_{CH} CH$ models the decay of CH, $\beta_{CH|N}.N.A$ refers to the production of cytokines by neutrophils, and $D_{CH} \Delta CH$ models the diffusion of cytokines in the tissue.

$$\begin{cases} \frac{dCH}{dt} = -\mu_{CH} CH + \beta_{CH|N} N \cdot A + D_{CH} \Delta CH \\ CH(x,0) = 0, \quad \left.\frac{\partial CH}{\partial n}\right| \partial\Omega = 0 \end{cases} \tag{3}$$

The authors' numerical experiment was divided into five cases as described below:

Case 1: Only LPS is present in the system.

Case 2: LPS and Neutrophils are present in the system. The source term of Neutrophils is zero.

Case 3: LPS and Neutrophils are present in the system. A source term is added but permeability is constant.

Case 4: LPS, Neutrophils, and Pro-inflammatory Cytokines are present in the system. Chemotaxis is absent.

Case 5: LPS, Neutrophils, and Pro-inflammatory Cytokines are present in the system. Chemotaxis is present.

3 Implementation

We used Python 3.5.1 on a 3.30 GHz Intel® Core™ i7-5820K CPU with 16.0 GB RAM in all our simulations.

In order to get the numerical solution of the coupled partial differential equations in (1)-(3), we utilize finite difference approximation with numerical scheme of the second order of accuracy for spatial discretization and explicit numerical scheme of the first order for time evolution. We also used $h = 0.1$ and $\Delta t = 0.01$ in order to satisfy the Courant-Friedrichs-Lewy (CFL) condition. The partial differential equations were then evaluated for 51 data points through space and 24000 data points through time[6].

4 Genetic Algorithm

Genetic algorithm (GA), one of the most basic forms of evolutionary algorithms, is a heuristic search algorithm that is adapted from the evolutionary ideas of natural selection and genetics [7–9]. This simple evolutionary algorithm has been used in many researches for parameter optimization and calibrations [9–11].

GA works by moving from one population of chromosomes to another through controlled selection based on a pre-defined fitness that is problem-specific. This is followed by crossover and then mutation; both of which are methods patterned from genetics. In essence, the fitter the chromosomes are, the more probable they are to reproduce.

In the context of GA, chromosomes (e.g. a list of strings or integers) are composed of genes (e.g. strings or positive real numbers), where each gene is represented by an instance of a specific allele (e.g. a letter from the alphabet or positive real numbers within the range 0 to 10). Crossovers, much like in the biological recombination of two single-celled organisms, allow exchange of subparts between two chromosomes. Mutation, on the other hand, randomly changes the value of one or several alleles in the chromosome.

Our adaptation of the genetic algorithm follows the classical structure of GA with solving PDEs as additional step prior to fitness evaluation. This will be discussed in detail in section 5.1. A summary of the steps taken for our implementation of GA are shown in the figure below:

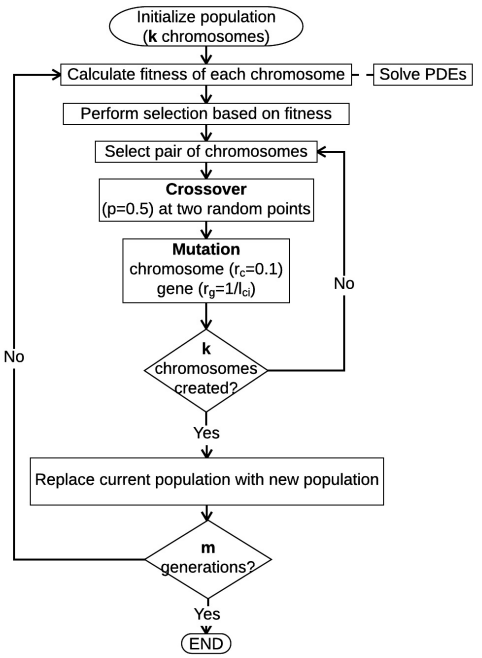

Figure 1. Summary of the steps implemented in genetic algorithm. A population of size $\boldsymbol{k = 200}$ is initialized. Fitness is calculated using equations (4)-(5), which becomes the basis for building a new population. A pair of parent chromosomes is randomly chosen from this population. Crossover between chromosomes is employed with probability $\boldsymbol{p = 0.5}$. Each chromosome undergoes mutation with probability $\boldsymbol{r_c = 0.1}$ where each allele is mutated with probability $\boldsymbol{r_g = 1/l_{c_i}}$, given $\boldsymbol{l_{c_i}}$ is the length of chromosome $\boldsymbol{i}$. Once $\boldsymbol{k}$ chromosomes are created, the newly generated population replaces the current population. This process is repeated for $\boldsymbol{m}$ generations

5 Computational Experiments Setup

5.1 Genetic Algorithm Computational Setup

Our adaptation of GA starts with the initialization of a population of $k = 200$ chromosomes composed of one, all, or a combination of the following parameters: $D_A, D_N, D_{CH}, \lambda_{N|A}, \lambda_{A|N}, \beta_{CH|N}$, and χ_N depending on the cases and methods used. More of this will be explained in detail in the succeeding paragraphs. Parameter values are randomly chosen within their respective ranges and these are summarized in Table 1.

Table 1. Summary of parameters and corresponding acceptable and calibration ranges used to reproduce the results of [1] through genetic algorithm.

Parameter	Unit	Acceptable Range	Calibration Range	Pigozzo et al.	Values Used for Validation	Reference
D_A	mm^2/day	$2.6 - 7.0$	$2.5 - 7.0$	0.002	2.5	[12]
D_N	mm^2/day	$0.037 - 0.173$	$0.04 - 3.0$	0.012096	2.94	[13]
D_{CH}	mm^2/day	$2.3 - 4.3$	$2.3 - 5.0$	0.0090216	5.0	[12]
$\beta_{CH\|N}$	$1/cell.day$	$0.1 - 0.4$	$0.1 - 0.4$	0.4	0.33	[14]
χ_N	mm^2/day	$2.6\times10^{-5} - 4.6$	$0.0 - 8.0$	0.0144	7.8	[13]
$\lambda_{A\|N}$	$1/cell.day$	$0.25 - 1.50$	$0.25 - 1.5$	0.55	0.45	[14]
$\lambda_{N\|A}$	$1/cell.day$	$0.25 - 1.50$	$0.25 - 1.5$	0.55	0.45	[14]

In order to obtain a good combination of parameters, we implement two methods that specify how these parameters are designated in a chromosome.

Method 1: Ladder-type Scheme

We pattern Method 1 from the systematic progression of cases presented by [1] in their numerical experiment. This allows optimization of parameters in a ladder-type manner. This is done by initially optimizing D_A through GA in Case 1. The chromosome for Case 1 only contains one parameter: D_A. The calibrated value for D_A in Case 1 is then passed on and utilized in cases 2 and 3. Cases 2 and 3 then proceed to calibrate D_N, $\lambda_{N|A}$, and $\lambda_{A|N}$. The chromosome for cases 2 & 3 now contains two parameters: D_N and λ where $\lambda_{N|A} = \lambda_{A|N}$ and so on. The steps taken for Method 1 are summarized in Figure 2.

Figure 2. Method 1: Ladder-type scheme. Case 1 calibrates $\boldsymbol{D_A}$. Cases 2 &3 calibrate $\boldsymbol{D_N}$, $\boldsymbol{\lambda_{N|A}}$, and $\boldsymbol{\lambda_{A|N}}$. Case 4 calibrates $\boldsymbol{D_{CH}}$ and $\boldsymbol{\beta_{CH|N}}$. Case 5 calibrates $\boldsymbol{\chi_N}$.

Method 2: Altogether Scheme

Method 2, on the other hand, deals with a fresh set of parameters for each case. This means that values of parameters that are previously calibrated will not be passed on to latter cases. Hence, GA is implemented to calibrate all the parameters in one go for a specific case.

We evaluate the fitness of each chromosome by first solving for each player's (either A, N, or CH) root mean square error ($RMSE_{player}$) calculated with respect to the data points from the plots of [1] ($f_{pigozzo}$) and our solution for equations (1)-(3) using GA (f_{GA}). The data points were extracted from the images of the plots on the authors' paper using an online software called WebPlotDigitizer [15]. We then solve for each player's normalized RMSE ($NRMSE_{player}$) by dividing the $RMSE_{player}$ with the mean of our solution in GA ($\langle f_{GA} \rangle$). This is summarized in equation (4).

$$\begin{cases} NRMSE_{player} = \dfrac{RMSE_{player}}{\langle f_{GA} \rangle} \\ RMSE_{player} = \sqrt{\dfrac{\sum_0^d \left(f_{pigozzo} - f_{GA}\right)^2}{d}} \end{cases} \tag{4}$$

where d is the total number of data points for each plot and the subscript $player$ could either be A, N, or CH. Finally, the overall fitness of the ith chromosome (c_i) in a population of size k is computed by summing all the $NRMSE_{player}$ and is given by the following equation:

$$f(c_i) = \sum_{0}^{total\ players} NRMSE_{player} \tag{5}$$

Based on the fitness value $f(c_i)$ calculated through equations (4)-(5) a new population of chromosomes of size $k = 200$ is selected where the fitness objective is set at minimum. We also note that a chromosome can be selected more than once in our simulations. From this pool of fit chromosomes, a pair of parent chromosome is randomly chosen. This is followed by a crossover with probability $p = 0.5$ at two randomly chosen points in the chromosomes. Mutation is then implemented with probability $r_c = 0.1$ for each chromosome. Each gene (parameter) has probability $r_g = 1/l_{c_i}$ to mutate or change its value within the specified parameter range, where l_{c_i} is the length of the chromosome or the total number of parameters in the chromosome. This is repeated for all pairs of chromosomes in the parent population. The new population, which is composed of the newly generated offspring that are essentially fitter than their parents, then replaces the parent population. This process is repeated for $m = 200$ generations (i.e. reevaluation of fitness) generating a newer and fitter population each time. Note, however, that the choice for the parameters used in our implementation of GA is arbitrary. Exploring the sensitivity of these parameters will be one our goals for our future work.

5.2 Validation Experiment Setup

In order to validate the genetic algorithm described in section 4, we utilize the same equations by [1] shown in equations **(1)-(3)**. We set the values of the parameters as shown in Table 1 and solve the partial differential equations using the methods described in [16] namely: finite difference method and explicit time integration. The generated solution then serves as our reference data.

We then proceed with parameter optimization using two methods in genetic algorithm as described in section **4** where parameters $\boldsymbol{D_A}$, $\boldsymbol{D_N}$, $\boldsymbol{D_{CH}}$, $\boldsymbol{\beta_{CH|N}}$, $\boldsymbol{\chi_N}$, $\boldsymbol{\lambda_{A|N}}$, and $\boldsymbol{\lambda_{N|A}}$ are to be calibrated within the ranges specified in Error! Reference source not found.. The goal here is to calibrate all the parameters to the exact values used in generating the reference data (see Error! Reference source not found.) by obtaining a minimum fitness value $\boldsymbol{f(c_i)}$ or total **NRMSE** between the reference data and the solution generated using GA.

6 Validation Results

In order to validate the genetic algorithm, we utilize equations (1)-(3) to calibrate parameters D_A, D_N, D_{CH}, $\beta_{CH|N}$, χ_N, $\lambda_{A|N}$, and $\lambda_{N|A}$ using the two methods described in section 4

Method 1 : Cases 1 to 5

Method 1 is a ladder-type scheme that allows parameters to be calibrated one case at a time. The calibrated parameters in earlier cases are then used in latter cases. Using equations (1)-(3) as test equations for validating the GA, we are able to come obtain the log_{10} fitness values $f(c_i)$ or total NRMSE values for each player corresponding to cases 1-5 over 5000 iteration steps. We see here that the fitness values go as low as 10^{-12} in all 5 cases. Note that the oscillations in this case are brought about by chromosome mutations.

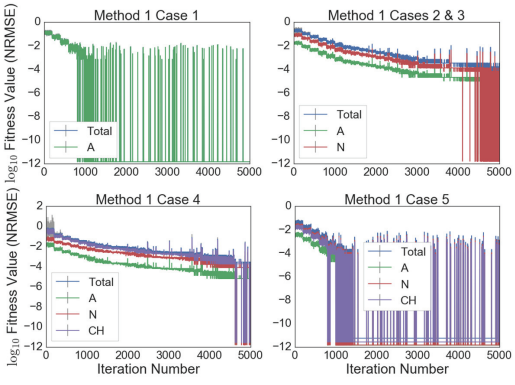

Figure 3. Method 1 Average $\boldsymbol{log_{10}}$ (NRMSE) with respect to iteration number simulated over twenty (20) trials. Gray lines correspond to the standard deviation.

We see in Figure 4 that parameter values converge to specific values in as early as 1000 iteration steps. By utilizing Method 1 in GA, we obtain the exact values for D_A, D_N and D_{CH} as that shown in Table 1.

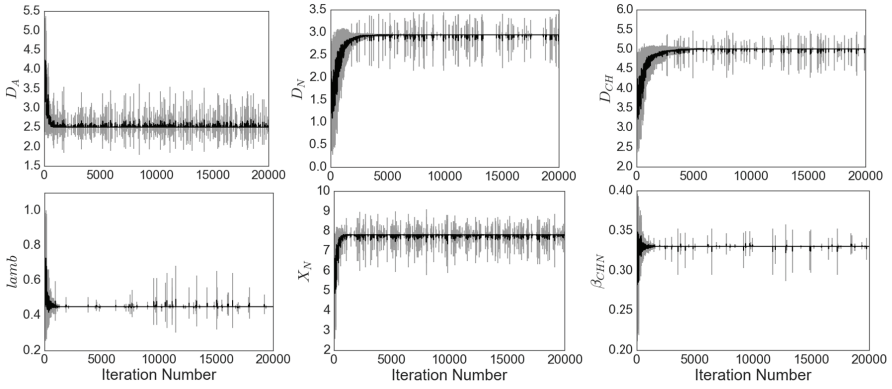

Figure 4. Method 1 Average concentrations of $\boldsymbol{D_A}$, $\boldsymbol{D_N}$, $\boldsymbol{D_{CH}}$, $\boldsymbol{\beta_{CH|N}}$, $\boldsymbol{\chi_N}$, $\boldsymbol{\lambda_{A|N}}$, and $\boldsymbol{\lambda_{N|A}}$with respect to iteration number simulated over twenty (20) trials. Gray lines correspond to the standard deviation.

Method 2 : Case 5 Only

Method 2 Case 5, on the other hand, deals with a set of five parameters that are calibrated at the same time. Averaging over 20 trials gives us the average NRMSEs and average concentrations of D_A, D_N, D_{CH}, $\beta_{CH|N}$, χ_N, $\lambda_{A|N}$, and $\lambda_{N|A}$ with respect to the iteration number as shown in Figure 5.

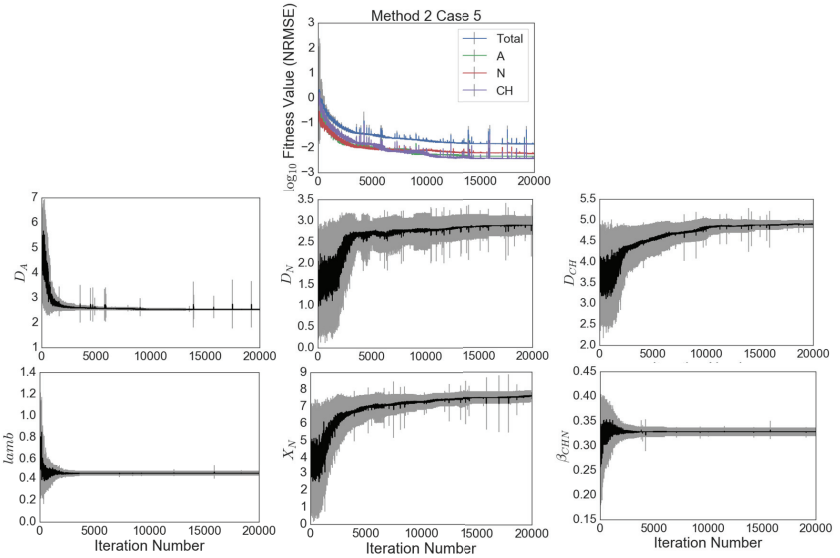

Figure 5. Method 2 Average $\boldsymbol{log_{10}}$ (NRMSE) and average concentrations of $\boldsymbol{D_A, D_N, D_{CH}}$ with respect to the iteration number over twenty (20) trials. Gray lines correspond to the standard deviation.

Parameter values start to converge at around 5000 iteration steps, which is much later than what is observed in Method 1. This, however, is expected since more parameters are being calibrated per iteration step. The average NRMSEs in this case go as low as 10^{-2}. As opposed to that in Method 1, this time, however, we are only able to calibrate the values for D_A, D_N, $\beta_{CH|N}$, $\lambda_{A|N}$, and $\lambda_{N|A}$ but not D_{CH} and χ_N. Method 2 using GA gave us $D_{CH} = 4.9$ and $\chi_N = 7.6$ instead of exact values $D_{CH} = 5.0$ and $\chi_N = 7.8$. Looking at all the trials, we found that only 6 out of 20 trials were able to arrive at the exact values for all parameters. Moreover, we were able to obtain the minimum fitness value of 10^{-12} in these combinations. This result tells us that our genetic algorithm using Method 2 does not go into global minimum majority of the time.

We plot the solutions that we have obtained using the calibrated values from methods 1 (solid lines) and 2 (dashed lines) together with the reference data (markers) in Figure 6. Only the best set of

parameters in the 20 trials were used to plot the solutions for Methods 1 and 2. We see here that the solutions for both methods 1 and 2 seem to coincide perfectly with each other. This tells us that the slight deviation of D_{CH} and χ_N from the preferred values does not lead to a drastic change in plot configurations as shown below.

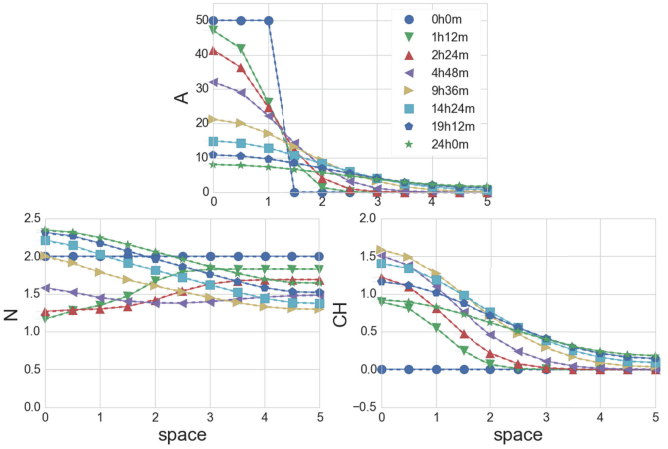

Figure 6. Data (markers) and solutions for Method 1 (solid lines), and Method 2 (dashed lines) using parameters calibrated through genetic algorithm for Case 3. Only the best set of parameters in the 20 trials were used to plot the solutions for Methods 1 and 2. Solutions coincide with the data.

With these results in mind, we now proceed with applying the GA to calibrate the sets of equations by [1] in the next section.

7 Immune System Model Calibration

Method 1 : Cases 1 to 5

Method 1 is a ladder-type scheme that allows the calibration of parameters starting from Case 1 and systematically moves to latter cases while incorporating parameters calibrated previously. Here we focus on Case 5, where $D_A, D_N, D_{CH}, \lambda_{N|A}, \lambda_{A|\mathrm{N}}$ and $\beta_{CH|N}$ have been previously calibrated in cases 1 to 4, and optimize the parameter χ_N. Figure 7 shows the average log_{10} (NRMSE) per iteration step simulated over twenty (20) trials. The average log_{10} (NRMSE) converge to minimum implying a decreasing point-to-point difference between the plots of [1] and our numerical solutions using the calibrated parameters in GA.

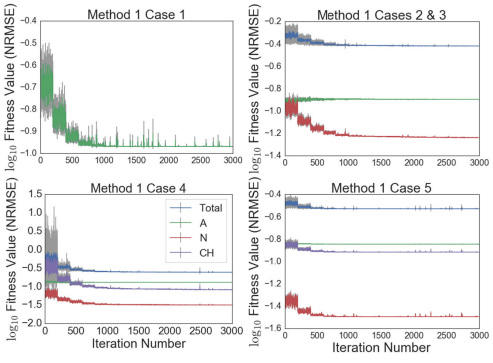

Figure 7. Method 1 Average $\boldsymbol{log_{10}}$ (NRMSE) for each player with respect to the iteration number simulated over twenty (20) trials. Gray lines correspond to the standard deviation.

Average concentrations of $D_A, D_N, D_{CH}, \lambda_{N|A}, \lambda_{A|N}$ and $\beta_{CH|N}$ over a span of 20000 iterations for 20 trials, on the other hand, are shown in Figure 8. We see here these concentrations converge to specific values in as early as 1000 iterations.

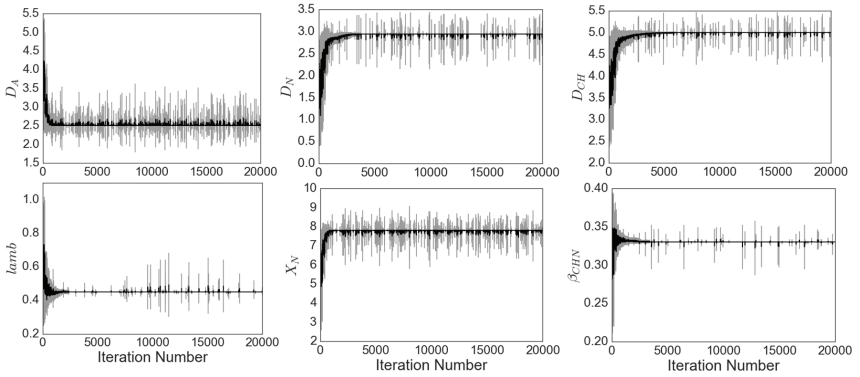

Figure 8. Method 1 Average concentrations of $\boldsymbol{D_A, D_N, D_{CH}, \lambda_{N|A}, \lambda_{A|N}}$ and $\boldsymbol{\beta_{CH|N}}$ with respect to the iteration number over twenty (20) trials. Gray lines correspond to the standard deviation.

A summary of our results for Method 1 are shown in Table . Based on our results, it seems that the calibrated value for D_N is beyond the acceptable range by a factor of 10. These values tell us that neutrophils are more mobile than LPS, which in fact does not reflect their actual behavior. In the biological standpoint, these values do not make sense since neutrophils are expected to move slower than LPS. The chemotaxis coefficient χ_N with calibrated value of 7.8, a rate almost 1.5 times the diffusion rate of pro-inflammatory chemokines, means that neutrophils respond to the gradient of chemokines at a rate faster than the diffusion of chemokines themselves – yet another behavior that is not biologically sound. However, the discrepancies in calibration may be contributed by the presence of chemotaxis in the modeled environment. Chemotaxis would have in turn affected the diffusion value of neutrophils especially in Case 5 where both diffusion and chemotaxis are taken into account.

Table 2. Summary of parameters and their corresponding calibrated values.

Parameter	Unit	Acceptable Range	Calibration Range	Pigozzo et al.	Calibrated in Case(s)	Calibrated Value	Final Fitness Value (Mean)
D_A	mm^2/day	$2.6 - 7.1$	$2.5 - 7.0$	0.002	1	2.5	0.11
D_N	mm^2/day	$0.037 - 0.173$	$0.04 - 3.0$	0.012096	2 & 3	2.94	0.38
$\lambda_{A\|N}$	$1/cell.day$	$0.25 - 1.50$	$0.25 - 1.5$	0.55	2 & 3	0.45	0.38
$\lambda_{N\|A}$	$1/cell.day$	$0.25 - 1.50$	$0.25 - 1.5$	0.55	2 & 3	0.45	0.38
D_{CH}	mm^2/day	$2.33 - 4.32$	$2.3 - 5.0$	0.0090216	4	5.0	0.24
$\beta_{CH\|N}$	$1/cell.day$	$0.1 - 0.4$	$0.1 - 0.4$	0.4	4	0.33	0.24
χ_N	mm^2/day	$2.6\times10^{-5} - 4.6$	$0 - 8.0$	0.0144	5	7.8	0.30

Method 2 : Case 5 Only

Method 2 deals with a fresh set of parameters for each case. But just like in Method 1, Method 2 also deals with a single parameter for Case 1: D_A. Moving on to Cases 2 and 3 (the union of these two cases has been explained previously), the method now allows the calibration of four parameters: $D_A, D_N, \lambda_{N|A}$ and $\lambda_{A|N}$, where $\lambda_{N|A}$ is assumed equal to $\lambda_{A|N}$. Calibrations of parameters then proceed per case and are treated independent of each other.

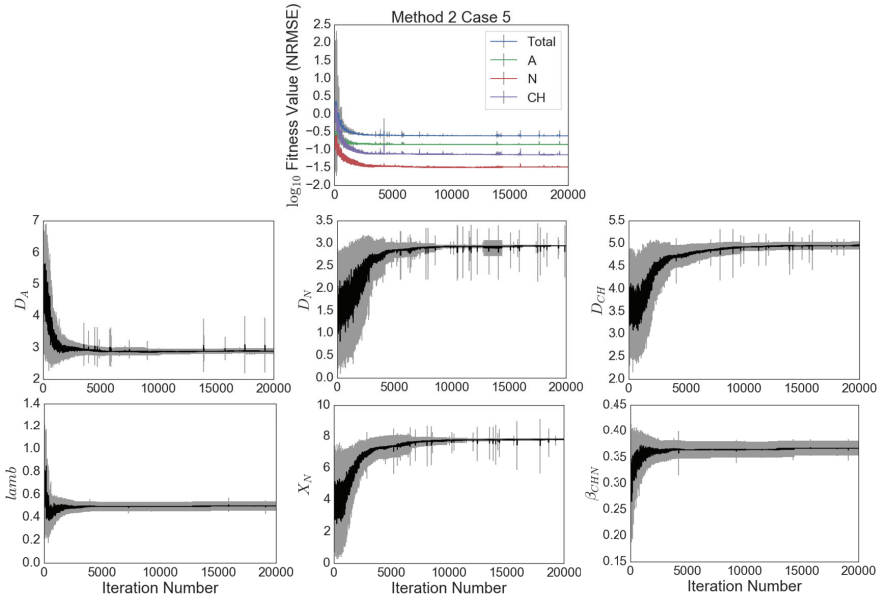

Figure 9. Method 2 Average Log Fitness Value (NRMSE) and concentrations of $\boldsymbol{D_A, D_N, D_{CH}, \lambda_{N|A}, \lambda_{A|\mathrm{N}}}$ and $\boldsymbol{\beta_{CH|N}}$ with respect to the iteration number over twenty (20) trials. Gray lines correspond to the standard deviation

Here we focus on Case 5 where we calibrate parameters $D_A, D_N, D_{CH}, \lambda_{N|A}, \lambda_{A|\mathrm{N}}$ and $\beta_{CH|N}$ using GA. Figure 9 shows the average log_{10}(NRMSE) and average concentrations for $D_A, D_N, D_{CH}, \lambda_{N|A}$, $\lambda_{A|\mathrm{N}}$ and $\beta_{CH|N}$ per iteration simulated over twenty (20) trials.

Using method 2, we were able to calibrate all parameters with values similar to that in method 1 in 5 out of 20 trials. Figure 10 plots the data (markers) extracted from the plots of [1] with our numerical solutions for Method 1 (solid lines) and Method 2 (dashed lines) of the PDEs in equations (1)-(3) by using the calibrated values for parameters $D_A, D_N, D_{CH}, \lambda_{N|A}, \lambda_{A|\mathrm{N}}$ and $\beta_{CH|N}$ obtained through GA.

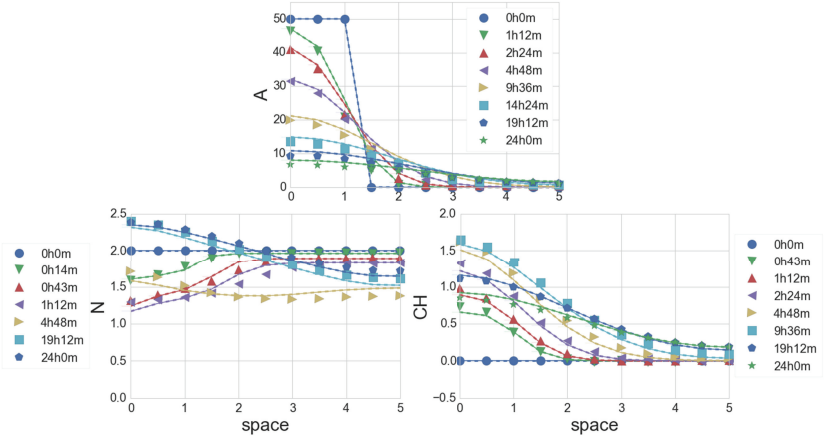

Figure 10. Data (markers) and solutions for Method 1 (solid lines), and Method 2 (dashed lines) using parameters calibrated through genetic algorithm for Case 5.

8 Conclusion

In this paper, we applied the genetic algorithm (GA) to calibrate parameters of the human innate immune system model by Pigozzo et al. We utilized two methods that vary the parameter composition of each chromosomes. The first method deals with a ladder-type scheme which makes use of previously calibrated parameters on subsequent cases. The second method, on the other hand, deals with a fresh set of parameters each time. We first tested the GA using Pigozzo et al.'s model by comparing a sample generated data, where parameters $D_A, D_N, D_{CH}, \lambda_{N|A}, \lambda_{A|N}, \beta_{CH|N}$, and χ_N are to be determined, with that of our solution. From this we were able to obtain the exact same parameters for Method 1 with fitness values (NRMSEs) approaching10^{-12}. We were able to calibrate all parameters in 5 out of 20 trials for Method 2. Plotting our test data and the solutions for methods 1 and 2 using GA gave similar configurations.

We proceeded with calibrating the parameters from Pigozzo et al.'s model. We found that all calibrated parameters are the same for methods 1 and 2. But this is only true for 5 out of 20 trials in Method 2. Increasing the number of iterations by increasing the generation and population sizes could contribute to better parameter optimization in Method 2.

GA provides a simple algorithm that allows us to do parameter optimization. However, it has its limitations. For instance, there is no guarantee that the simulation reaches an optimal solution due to factors including but not limited to mutation and crossover probabilities, initial population, choice of fitness function, and number of generations. Moving forward, we plan on exploring the sensitivity of the parameters used in our implementation of GA. We also intend to incorporate the concept of weights in our fitness evaluation to emphasize parts of the configurations that are important. For instance, we could implement multipliers as weights in solving fitness values at points $x = 0mm$ and $x = 5mm$. In this way, the endpoints are rendered more important than the rest of the configurations through space and time.

References

1. Pigozzo, A.B., MacEdo, G.C., Dos Santos, R.W., Lobosco, M.: Implementation of a computational model of the innate immune system. Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics). 6825 LNCS, 95–107 (2011).
2. Rocha, P.: Modelling the innate immune system. Bio-Inspired Comput. Algorithms Their Appl. (2012).
3. Chiacchio, F., Pennisi, M., Russo, G., Motta, S., Pappalardo, F.: Agent-based modeling of the immune system: NetLogo, a promising framework. Biomed Res. Int. 2014, (2014).
4. Celada, F., Seiden, P.E.: A computer model of cellular interactions in the immune system. Immunol. Today. 13, 56–62 (1992).
5. Begley, C.G., Ellis, L.M.: Drug development: Raise standards for preclinical cancer research. Nature. 483, 531–3 (2012).
6. Presbitero, A., Krzhizhanovskaya, V., Mancini, E., Brands, R., Sloot, P.: Reproducibility of Two Innate Immune System Models. Lect. Notes Comput. Sci. in print, (2016).
7. Holland, J.H.: Adaptation in Natural and Artificial Systems. (1975).
8. Mitchell, M.: An Introduction to Genetic Algorithms. 209 (1998).
9. Whitley, D.: An overview of evolutionary algorithm: practical Issues and common pitfalls. Inf. Softw. Technol. 43, 817–831 (2001).
10. Yu, W., Li, B., Jia, H., Zhang, M., Wang, D.: Application of multi-objective genetic algorithm to optimize energy efficiency and thermal comfort in building design. Energy Build. 88, 135–143 (2015).
11. Espinosa, O.B.: A Genetic Algorithm for the Calibration of a Micro- Simulation Model. (2011).
12. Goodhill, G.J.: Diffusion in axon guidance. Eur. J. Neurosci. 9, 1414–1421 (1997).
13. Saltzman, W.M.: Tissue engineering : engineering principles for the design of replacement organs and tissues. Oxford University Press (2004).
14. Su, B., Zhou, W., Dorman, K.S., Jones, D.E.: Mathematical Modelling of Immune Response in Tissues. Comput. Math. Methods Med. 10, 9–38 (2009).
15. Rohatgi, A.: WebPlotDigitizer - Extract data from plots, images, and maps, http://arohatgi.info/WebPlotDigitizer/index.html.
16. Presbitero, A., Krzhizhanovskaya, V., Sloot, P.: Reproducibility of Three Innate Immune System Models. Lect. Notes Comput. Sci. in print, (2016).