Chapter 7
Optimization in Biology Parameter Estimation and the Associated Optimization Problem

Gunnar Cedersund, Oscar Samuelsson, Gordon Ball, Jesper Tegnér and David Gomez-Cabrero

Abstract Parameter estimation—the assignment of values to the parameters in a model—is an important and time-consuming task in computational biology. Recent computational and algorithmic developments have provided novel tools to improve this estimation step. One of these improvements concerns the optimization step, where the parameter space is explored to find interesting regions. In this chapter we review the parameter estimation problem, with a special emphasis on the associated optimization methods. In relation to this, we also provide concepts and tools to help you select the appropriate methodology for a specific scenario.

Keywords Parameter estimation · Optimization · Heuristic · Fitness function

7.1 Introduction

Mathematical models have been part of biological research for more than half a century. One of the starting points for this development was the now classical model for the action potential in an axon, developed by Hodgkin and Huxley [27, HHM]. HHM

G. Cedersund (✉)
Integrative Systems Biology, Department of Biomedical Engineering, Linköping University, 58185 Linköping, Sweden
e-mail: gunnar.cedersund@liu.se

G. Cedersund
Department of Clinical and Experimental Medicine, Linköping University, 58185 Linköping, Sweden

O. Samuelsson · G. Ball
Stockholm, Sweden

J. Tegnér · D. Gomez-Cabrero
Unit of Computational Medicine, Department of Medicine, Karolinska Institutet, Solna, Sweden
e-mail: david.gomezcabrero@ki.se

J. Tegnér · D. Gomez-Cabrero
Center for Molecular Medicine, Stockholm, Sweden

L. Geris and D. Gomez-Cabrero (eds.), *Uncertainty in Biology*,
Studies in Mechanobiology, Tissue Engineering and Biomaterials 17,
DOI 10.1007/978-3-319-21296-8_7

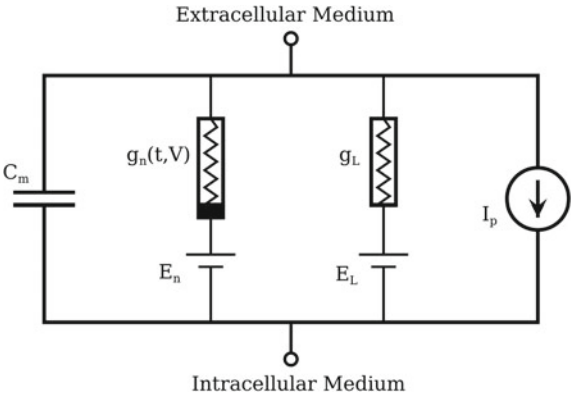

Fig. 7.1 Schematic of the basic Hodgkin-Huxley model (HHM) of a neuron. The figure depicts the basic components of HHM models that represent the biophysical characteristics of cell membranes. In the model Capacitance (*Cm*) represents the lipid bilayer. Nonlinear (*gn*) and linear (*gL*) conductances represent voltage-gated and leak ion channels respectively. Batteries (*E*) represent the electrochemical gradients driving the flow of ions. Current sources (*Ip*) represent ion pumps and exchangers. Figure obtained from Wikimedia, under Creative Commons CC0 1.0 Universal Public Domain Dedication. Author: Krishnavedala

explains these action potentials using a set of coupled nonlinear ordinary differential equations (ODEs), which are associated to membrane channels with specific time-and voltage-dependent properties (Fig. 7.1). The HHM is of major relevance because (i) it was the culmination and integration of a large number of experiments [17, 52] where the model provided a mechanistically unified vision of the system; and because (ii) HHM generated specific predictions that were subsequently validated using new single-channel recording techniques [26]. For these reasons, Hodgkin and Huxley were awarded the Nobel Prize in 1963, the first Nobel Prize in physiology awarded for the development of a mathematical model. An illuminating account of the development of both experiments and model is available in [26].

However, those early successes by Hodgkin and Huxley were possible because they could bypass one of the biggest hurdles in most current biological modeling: the simultaneous determination of all the values of the parameters in the model from systems-level data. These parameters that need to be estimated appear in the ODEs, and in the case of HHM, these parameters could be experimentally characterized from initial targeted experiments. In other words, their only problem was that of forward simulation, which was possible even in the pre-computer era. For most other biological models, parameters cannot be determined directly in specific experiments, but instead need to be estimated simultaneously from systems-wide data. Particularly, one needs to search the space that is spanned by the unknown parameters in the model, in order to characterize which parts of the parameter space are consistent with experimental data and prior knowledge. This task of determining parameters from data is known as parameter estimation, and a key step in this task is to optimize the agreement between model and data. In other words, optimization research enters as a natural component in research on parameter estimation for biological models

[2, 22, 23, 36]. This optimization step is not trivial, and there are a wide variety of methods to choose from.

In this chapter, we will describe the basics of the parameter estimation problem, look in depth at the associated optimization problem, and at the available tools to solve it. In Sect. 7.2, the parameter estimation problem is introduced using an intuitive example and in Sect. 7.3, the problem is treated from a more mathematical viewpoint. Section 7.4 describes several meta-heuristic optimization approaches and Sect. 7.5 compares these approaches and provides useful inputs for selecting among them. Finally, Sect. 7.6 provides a closing overview of the chapter and provides links to open challenges, most of which are discussed in other chapters of this book.

7.2 The Parameter Estimation Problem

Let us start by introducing the parameter estimation problem from an intuitive point of view, through a specific biological model. The example is formulated as an ODE model, which is the most widespread formalism in systems biology [10, 31, 37, 48]. An ODE model can be formulated in state-space form, which is a mathematically well explored representation [1, 6], written as:

$$\acute{x}(t) = f(x(t), u(t), P) \tag{7.1}$$

$$y(t) = h(x(t), u(t), P) \tag{7.2}$$

where $x(t)$ are the states, $\acute{x}$ are the time-derivate of the states, f and h are smooth nonlinear functions, are the parameters, and $u(t)$ and $y(t)$ are input and output signals, respectively. All symbols are vectors.

Let us now consider the specific model in Fig. 7.2. This figure depicts an interaction-graph for a simple model of the insulin signaling network, taken from [43]. As can be seen, the model describes how insulin binds to the insulin receptor (IR), which thus becomes phosphorylated (IRp), and able to phosphorylate the protein IRS, which phosphorylates the protein PKB, which finally stimulates transport of Glut4 to the membrane, which together with Glut1 transports glucose into the cell. There is a straightforward approach to go from an interaction-graph to a state-space formulation [50], and the full ODEs are given in [43]. A short summary of this process is as follows. The states correspond to the concentrations (or, alternatively, the amounts) of the different molecules, i.e. $[IR]$, $[IRSp]$, etc. The concentrations are affected by the reactions, and the rate of these reactions is the first thing that needs to be defined. In this model, mass-action kinetics are assumed, which means that the rates are given by the product of the concentrations of the substrates and the regulators multiplied by an unknown rate constant [3]. The unknown rate constants are called kinetic rate constants, here denoted k_i, and they thus regulate the speed of each individual reaction. It is often these rate constants that need to be determined from experimental data. These rates are summed together to form the ODEs, so that rates corresponding to ingoing reactions appear with a plus sign, and rates corresponding to outgoing reactions appear with a minus sign. In other words, the ODE for $[IRS]$

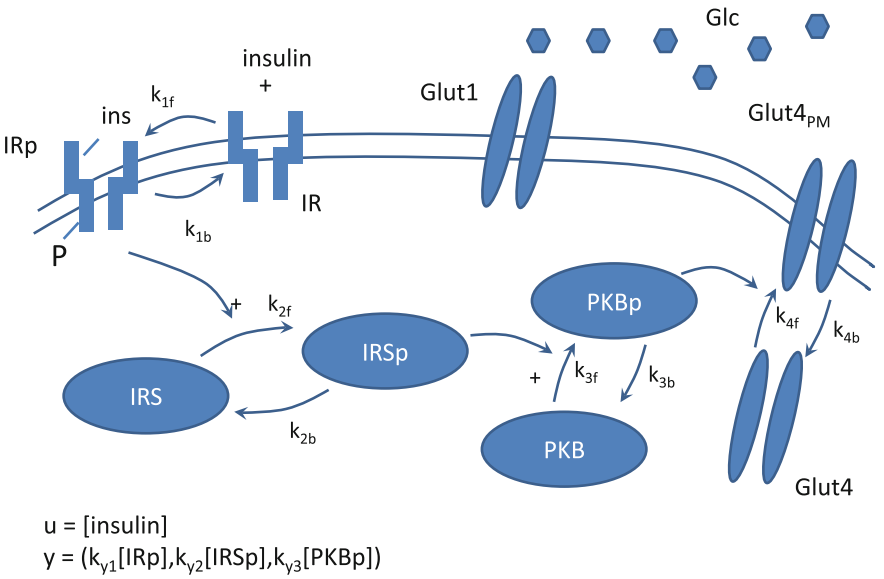

Fig. 7.2 A schematic description of an insulin-signaling network. The input and output signals are the concentrations of insulin, and the measurements of the phosphorylated proteins, *IRp*, *IRSp* etc. This schematic description is referred to as an interaction-graph, and it corresponds one-to-one with the ODEs, as is described for instance in [43], which also is the paper from which this model is taken

is given by:

$$[\acute{IRS}] = \frac{d[IRS]}{dt} = \acute{x}_{IRS} = k_{2b}[IRSp](t) - k_{2f}[IRS][IRp](t) \tag{7.3}$$

In the model specification, each parameter p_i (usually, but not always, corresponding to some rate constant k_j) is specified to have values in a specified range. This range is here defined by l_b below and u_b above

$$l_{b,i} < P_i < u_{b,i} \tag{7.4}$$

These boundaries are referred to as box-constraints. The box-constraints should include all physically possible parameter values. If the range of physiologically likely parameters is unknown, a box-constraint would still be required in order to restrict the parameter space. Many physiologically relevant models are at least bounded by 0 below, and the time-scale of the observed dynamics can often give a suggestion for an upper bound as well. The box-constrained parameter space may also be referred to as the search space, and a parameter set is a point in the (box-constrained) parameter space.

The parameter estimation problem is to find those parameter sets that both are biologically feasible and that display a satisfactory agreement with experimental observations. An experimental observation may refer to any set of values that are available from experimental measurements. One example is a so-called perturbation experiment, where a cell is perturbed from steady-state, and where the transient response of the system is observed. In the insulin example above, insulin is such a perturbation, i.e. the levels of all the states are assumed to be in steady-state before the perturbation, and one then follows how the signaling intermediates ([*IRp*], [*IRSp*], etc.) are changing over time. Such perturbation experiments can also be done in other systems, e.g. gene regulatory networks. In gene regulatory networks the availability of perturbation experiments is growing as several technological advances [39] have allowed the generalization of gene perturbation screens of the systems. One such example is RNA interference experiments, which are being used as a tool for both association discovery and for validation purposes [53].

The agreement between model and data can be of two types: qualitative and quantitative. A qualitative agreement can often be assessed by mere inspection and reasoning. For instance, the initial response to an insulin stimulation is that [*IRp*] and [*IRSp*] goes up. Another qualitative observation may be that the system oscillates, i.e. that it changes periodically up and down over time. In these two cases, the model will qualitatively agree with the data if it goes up in response to insulin, or if it oscillates, respectively. However, experimental data that have been collected often contains fundamentally more information than that: it usually contains quantitative information. Such quantitative information is typically measured as the average deviation from the data points (Eq. 7.5 below). Loosely, parameters that give an acceptable agreement with the data are called feasible, and other parameters are referred to as infeasible. Whereas the question of whether or not simulations for a specification parameter agree qualitatively with data can be assessed by mere inspection, the quantitative agreement requires a more formal treatment. This parameter characterization—which simulations agree, and which do not agree, with the data—is the task of parameter estimation.

Originally parameter estimation was done "by hand", by combining physiological knowledge with reasoning and manual tests. Such an approach is time-consuming, requires expert knowledge on the exact role of each parameter, and is therefore infeasible for most biological models. With the invention and development of computers, this estimation "by hand" could be replaced by more exhaustive searches in the parameter space, since computers allow for thousands or millions of parameter sets to be tested. This development of computers has gone hand-in-hand with the developments in the field of optimization. More specifically, the application of methods from the field of optimization has been essential for the exploration of extremely large parameter spaces. This is the case, since the task of covering large parameter spaces seldom can be solved by "brute-force" approaches, even with the rapid advances in computer power. Let us therefore now turn to a more formal treatment of these concepts, using the language of mathematical optimization.

7.3 The Optimization Problem in Parameter Estimation

In Sect. 7.2, we learned that parameter estimation is concerned with the search for feasible parameter sets in the box-constrained parameter space, and that "feasible" measures the agreement between our experimental observations and the model simulations, either qualitatively or quantitatively. Now the challenge is to transform the concepts "feasible" and "agreement" into mathematical terms.

Let us start by considering a specific set of data and simulation possibilities, again taken from insulin signaling [5]. The example is depicted in Fig. 7.3. The data reflects the amount of IRSp, and at time $t = 0$ the system (isolated fat cells) have been stimulated with 100 nM insulin. The collected data points are depicted in blue, where the error bars depict the uncertainties (top to bottom reflects 2 standard deviations, 2σ). As can be seen, these data have a big uncertainty, but one can nevertheless see certain features clearly. First, the system responds by going up, but then around one $t = 1$ min, the response decreases, and from somewhere between 5 and 10 min and onwards, the system has settled at an intermediate steady-state level. Now, when a model tries to mimic these data, there are two aspects that can be taken into account. First, one may capture the presence of the overshoot (that the response goes up and then down), and this is captured by the blue but not the green line. Second, one may capture the quantitative features of the data, e.g. the location of the final steady state. This steady-state is, conversely, captured by the green but not the blue line. These two lines correspond to simulations with the studied model, done using different parameter values, and this example illustrates that quantitative and qualitative aspects

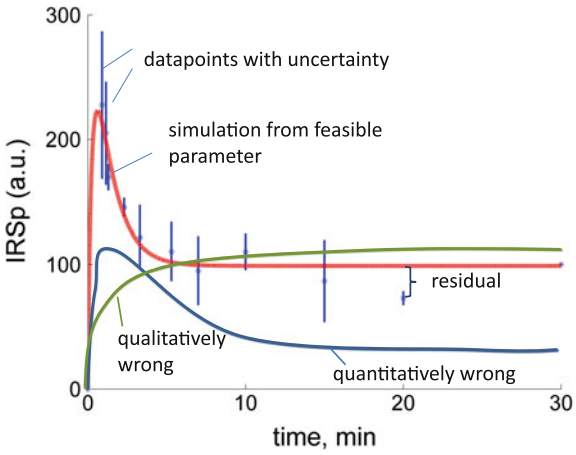

Fig. 7.3 Example of data and model simulations. This example is also from insulin signaling, as described in [5]. In *blue* you see the data points with uncertainty, and the other lines show simulations for different parameters values: one that is qualitatively wrong but quantitatively right (*green*), one with the opposite problem (*blue*), and one from a feasible parameter (*red*). A cost function is formed by summing together the normed and squared residuals, as described in Eq. (7.5)

do not always coincide. The task is therefore, to decide upon how to quantify the features (quantitative or qualitative) that are important in the data, and then optimize this quantification, to find the best parameters (red line).

In other words, we need an automatic way to evaluate how "feasible" a parameter set is. Let us start by denoting the experimental data point at time t_k with $y(t_k)$. Similarly, let $\hat{y}(t_k|P)$ denote the simulated data point, at time t_k and using the parameters P. Now, we are interested in the differences between these two values, the so called residuals $r(t_k) = y(t_k) - \hat{y}(t_k|P)$. In general, the larger the residuals, the worse the model, so we are interested in the average size of the residuals. However, we want them to be always positive (since it does not matter if the model is above or below the data). Furthermore, if the data point is highly uncertain (there is a big $\sigma(t_k)$), that residual should be less important, when taking the average. All of these requirements are met by studying the following cost function

$$V(P) = \sum_{k=1}^{N} \frac{\left(y(t_k) - \hat{y}(t_k|P)\right)^2}{\sigma(t_k)^2} \tag{7.5}$$

As can be seen the residuals have been squared (to make the residual always positive), normalized with the standard deviation of the measurement uncertainty (divided with σ), and then summed together (giving the average size, considering all residuals). This cost function is the most common choice, and it is usually denoted the chi-square cost function. Independently of which cost function, $V(P)$, that is used, the final parameters are in the end defined by

$$\hat{P} = \underset{P}{argmin}\, V(P) \tag{7.6}$$

Let us now consider a general optimization problem, and see if we have succeeded with the reformulation of the parameter estimation problem. In mathematical optimization, the problem is defined by three elements: (a) the objective function—in our case the evaluation function described in (5); (b) a set of decision variables that can vary during the search—in our case the parameters P; and (c) the constraints that limit the value of the decision variables or the relations between them—in our case the box constraints (4). The problem we have described is thus a mathematical optimization problem. More specifically, our problem is a continuous optimization problem, because the decision variables are all real numbers. A systems biology oriented description of optimization is available at [2].

Let us now consider a complication of this general scenario that often happens: additional constraints. Such constraints may stem from prior knowledge, not contained in the data. Consider for instance a system that contains the variables C and D, where expert knowledge assures us that D cannot be more than twice the value of C at any time. Then this extra information should be added to the problem setting. The information can be added to the system in two ways. The first way is to include the information as a constraint in the objective function (in the same way the boxed-parameter space defines inequalities):

$$D(p,t) < 2C(p,t) \forall t \tag{7.7}$$

The second way (which we favor here, as it is easier to integrate with the heuristic methodologies shown in the next section) is to construct a new objective function $V_{tot}(P)$ that is the weighted combination of the normal "data-based" objective function $V(P)$ (e.g. that in Eq. (7.5)) and a new one, $eval2(P)$

$$V_{tot}(P) = V(P) + w \cdot eval2(P) \tag{7.8}$$

$$eval2(p) = \sum_{k=1}^{N} dist(C, D, P, t_k) \tag{7.9}$$

$$dist(C, D, P, t) = \begin{cases} (2C(p,t) - D(p,t)) & if\ (2C(p,t) > D(p,t)) \\ 0 & otherwise \end{cases} \tag{7.10}$$

As can be seen, $eval2(P)$ will be zero if the relation based on expert knowledge between C and D is fulfilled by the simulation, and if the constrain is violated, $eval2(P)$ will grow with the size of the violation. The parameter describes the weighting between the two sources of knowledge: the experimental data ($V(P)$) and the expert knowledge ($eval2(P)$). These kind of *ad hoc* expansions of the cost function can be used to incorporate also other types of knowledge, or preferences regarding how the data should behave. In the example in Fig. 7.3, one could for instance decide that the model should have an overshoot. One could then add a similar punishment as $eval2$ in (8) to such simulations that do not produce such an overshoot.

An alternative approach to a weighted combination of different elements in a cost function is to compute Pareto Optimal (PO) solutions. Considering two evaluations functions, A and B, a PO solution is such that there is no other solution with better evaluation for both A and B simultaneously. The study of Pareto Optimal is an active field in multi-objective optimization but we do not consider it further in this chapter. Relevant examples of multi-objective optimization in biology can be found in [8, 57, 58].

Finally, optimization has been used in the study of many different types of biological systems, and the details of the formulations may differ from setting to setting. Common examples include flux balance analysis [51], metabolic engineering, signaling pathways [43, 54], reverse engineering [24], and stochastic modelling [60]. However, a common feature, usually appearing in biological systems is that there are many local optima. This means that global optimization algorithms should be used to search the parameter space. Such global methods are usually described using meta-heuristics.

7.4 Meta-Heuristics: A Tool for Optimization

7.4.1 Introduction to Meta-Heuristics

The term meta-heuristics is a composition of the Greek words *meta* (beyond) and *heureskin* (to find) or *heuristic* (rules of thumb). Meta-heuristics combine basic heuristic methods in higher-level frameworks to efficiently explore the box constrained objective function to be optimized. Introductions to meta-heuristics can be found at [19].

Meta-heuristics require little or no information about the problem to be solved, and they are sometimes referred to as black-box optimization methods. Interestingly, the same meta-heuristic algorithm can usually be applied to a wide range of different problems. Meta-heuristics do not make explicit use of Hessians or gradients as opposed to gradient descent methods, and can therefore be used also on discontinuous problems. More importantly, most parameter estimation problems are multi-modal, i.e., they contain several optima; hence, traditional gradient-based local search methods usually fail at locating the global optimum in such problems. In the search for the global optimum, one is therefore directed to use global optimization methods where one alternative is to use meta-heuristic methods.

Within the area of global optimization there are two different approaches, deterministic and stochastic methods, and they differ in their ability to guarantee convergence towards the global optimum. The deterministic methods can in theory provide a level of assurance of finding the global optimum. However, the computational time often increases exponentially with problem size, which often makes it impossible to locate the global optimum in a reasonable time. The stochastic methods, including meta-heuristics, use random components to locate the vicinity of the optimum. The trade-off is that one cannot guarantee the identification of the global optimum. However, the computational cost is significantly lower for stochastic methods compared to deterministic methods and this is an important advantage in large problems.

Many different meta-heuristic methods exist and in this chapter only a few of the most common methods are described. It should also be noted that the way of implementing the algorithms can differ widely, and here, only the pseudo-code is shown in order to illustrate the basic structure of the algorithms. Let us now turn to the specific algorithms.

7.4.2 Simulated Annealing

Simulated annealing was originally developed for solving combinatorial optimization problems [35]. However, the scope of use for the algorithm has grown and it is currently used in various optimization areas for numerous types of optimization problems. The algorithm resembles the internal energy of atoms at different temperature levels. The analogy is as follows. At a high temperature, multiple states for the atoms are possible and a random state is adopted. A reduced temperature restricts

the number of possible states to ultimately converge to a single state at the absolute zero temperature. Finally, if the atoms are allowed to equilibrate at each temperature level, the final state is also the optimal state in terms of energy. In optimization, the energy function, which describes the states of the atoms, is replaced by the objective function $V(P)$ as described in [35]. Thus, a state of the atoms is replaced by a parameter vector in the search space.

In practice, the search space may be explored by a geometrical figure, e.g. a simplex. In a D-dimensional search space, a simplex is a hyper-triangle with D + 1 vertexes, where each vertex consists of a parameter vector. The simplex is programmed to update the parameter vector (vertex) with the worst objective function value with a better one. The updating procedure of the simplex is iterated, which results in a simplex that deforms and moves across the search space. At high temperatures, the simplex performs more or less a random search. However, the probability to include a parameter vector with higher cost than those already included in the simplex becomes smaller as the temperature decreases. Thus, at low temperatures, the simplex performs a local search.

To be able to guarantee convergence to the optimal parameter vector, an impractically slow cooling scheme must be used [55]. In practice, pre-defined faster cooling schemes are used, which do not guarantee global convergence. In this chapter, a slightly modified Simulated annealing procedure based on the Nelder-Mead Downhill Simplex algorithm (NMDS) is shown, for which a description can be found in [46]. The modification makes it possible to search for several optima simultaneously by using several simplexes, see "Algorithm 4" on page 27 in [44].

In this modified version, the starting point for each simplex is found by a clustering technique, which makes sure that the simplexes are started with a sufficient Euclidian distance between them. The temperature is decreased after a specified number of iterations for each one of the simplexes. The magnitude of the temperature decrease is set by a reduction factor which reduces the temperature in a step-wise manner. As the temperature reaches the defined end temperature it is set to zero, and no uphill moves are allowed. This is similar to a local search. A short pseudo-code description is given in Fig. 7.4, to illustrate the algorithm. For further information, see [44].

7.4.3 Scatter Search

The meta-heuristic method Scatter search was originally introduced by Glover in the late 1970s [18]. Scatter search involves strategic combinations of parameter vectors to generate improved parameter vectors.

As originally proposed in [18], the core of Scatter search is the Reference Set (RefSet) which contains a limited number of promising parameter vectors. The number of parameter vectors, denoted R_s, in the RefSet is an option specified by the user. Different heuristic methods interact with the RefSet during the search. Most scatter search implementations follow *the five step template,* originally described by Glover. These five steps, or sub-methods, are: *Diversification Generation Method*,

Pseudo-code Simulated Annealing

Requirements: Start temperature T_s, end temperature T_e, reduction factor α, start guess P_0, number of iterations I at each temperature, number of iterations at absolute zero temperature I_z, and maximum number of simplexes S.

```
Initiate parameters and set k = 1
Set T_k = T_s
while T_k > T_e do
        Perform I iterations of Nelder-Mead Downhill Simplex
        algorithm for the S simplexes
        Set T_{k+1} = αT_k
        Calculate R new restart points with single-linkage-clustering
        where R ≤ S
        Set k = k + 1
end while
Set T_k = 0
Perform I_z iterations of Nelder-Mead Downhill Simplex algorithm at
the absolute zero temperature
return Best parameter vectors found during the last temperature
iteration
```

Fig. 7.4 Pseudo-code: simulated annealing

Improvement Method, Reference Set Update Method, Subset Generation Method, and a *Solution Combination Method*. The methods will be briefly explained in the way they are implemented in SSm (Scatter Search for Matlab). For a detailed description, the reader is referred to the original description of SSm [15].

- *Diversification Generation Method* generates a set d consisting of D_p diverse trial parameter vectors. The parameter values for the D_p parameter vectors with D parameters are selected from ND cells, where the rows are N intervals for each parameter and the D columns are the different parameters. The intervals for each parameter are defined by dividing the region between the lower and upper boundaries in N intervals. Thus, all parameters have N intervals regardless of range between lower and upper boundary. Since the D_p parameters in d are selected from all of the ND cells, one can be certain that the trial parameter vectors are diverse. The probability $prob_{i,n}^{p+1}$ of choosing a new parameter vector $p + 1$ with one of its parameters in one of the N intervals is inversely proportional to the number of previous parameters in the specific interval, i.e.,

$$prob_{i,n}^{p+1} = \frac{\frac{1}{f_i}}{\sum_{k=1}^{N} \frac{1}{f_k}} \tag{7.16}$$

where f_i is the number of previous values of parameter i in the interval n. When the boundaries of one or several parameters are of different orders of magnitude, the intervals can be defined according to a logarithmic distribution providing a spread of the D_p parameter vectors across the entire search space.

- *Improvement Method* is optional to include in a Scatter Search design but is normally required to obtain high quality parameter vectors (in terms of cost function value). One option is to use the local search method *Dynamic hill climbing* [41] as an improvement method. The improvement method can be restricted by activating a merit filter which ensures that no local search is performed from a worse initial point than previously found.

- *Reference Set Update Method* updates the R_s parameter vectors according to quality and diversity. The first half of the RefSet is updated with high quality parameter vectors (low cost function value) which are selected from the Dp diverse parameter vectors such that

$$V\left(P_j\right) \leq V\left(P_{j+1}\right) \leq, \ldots, \leq V\left(P_{R_s}\right) \tag{7.17}$$

  where $V(P_j)$ is the cost function value for one of the R_s parameter vectors j. In order to obtain a RefSet with both parameter vectors with low cost and parameter vectors which are spread in the search space, the second half of the RefSet contains spread or *diverse* parameter vectors. Diversity is measured either by the spread in the parameter vectors in terms of Euclidian distance, or by the spread in the parameter vectors in terms of direction. For further information, see pages 57–60 in [14].

- *Subset Generation Method* creates a combination of parameter vectors and in this case, pairs of all $P_1, P_2, \ldots, P_{R_s}$ parameter vectors.
- *Solution Combination Method* explores the distance between all paired parameter vectors to find new parameter vectors. Different approaches exist on how to do the search, but in this implementation only linear or hyper-rectangle searches are used [14]

How the different methods are combined in a basic search is shown in the pseudocode, in Fig. 7.5. It should be pointed out that it is the way of implementing the five methods that decides the sophistication of the algorithm rather than the specific methods [14].

7.4.4 Genetic Algorithms

While many bio-researchers rely on computational methodologies for the analysis of large data-sets, it is interesting to observe how many methodologies have been designed in the recent decades by considering the principles observed in biological systems. For instance, machine-learning applications have used neurons as a reference to develop neural network classifiers [4, 30]. A second example, Genetic

Pseudo-code Scatter Search

Requirements: R_s (size of *RefSet*) and number of D_p diverse parameter vectors in the set d. Set remaining options to default.

Start with $d = \emptyset$
Repeat
    Use the *Diversification generation method* to construct a parameter vector and apply the *Improvement Method Local solver*
    Let P be the resulting parameter vector
    if $P \in d$ **then**
        $d = d \cup \{P\}$
    Else
        Discard P
    end if
until $|d| = D_p$
Use the *Reference set update method* to build $Refset = \{P_1, \ldots, P_{R_s}\}$ with the best $\frac{R_s}{2}$ quality parameter vectors and $\frac{R_s}{2}$ diverse parameter vectors in d
Sort the parameter vectors in *RefSet* according to their objective function value such that P_1 is the best parameter vector and P_{R_s} is the worst
Set $NewSolutions = TRUE$
while $NewSolutions$ **do**
    Generate $NewSubsets$ with the *Subset Generation Method*
    Set $NewSolutions = FALSE$
    while $NewSubsets = \emptyset$ **do**
        Select the next subset s in $NewSubsets$
        Apply the *Solution Combination Method* to s to obtain new trial parameter vectors
        Apply the *Improvement Method* to the s trial parameter vectors
        Apply the *RefSet Update Method* to the $R_s \cup s$ parameter vectors
        if *RefSet* has changed **then**
            Set $NewSolutions = TRUE$
        end if
        Delete s from $NewSubsets$
    end while
end while

Fig. 7.5 Pseudo-code: scatter search

Algorithms (GA), arguably the most well-known optimization procedure inspired by biology, is based in the studies of DNA sequence evolution [20] (pseudo-code in Fig. 7.6). GA mimics the process of natural selection in order to find quality solutions in optimization problems. Briefly, GA mimics the natural selection process by considering an initial population of solutions (that can be generated randomly)

Pseudo-code Genetic Algorithm

Requirements: maximum number of iterations (It_{max}), number of elements in a population (*pop*), maximum CPU, Mutation, Crossover and Selection.

```
Initiate randomly the solution set: pool.
Set k = 1
while It_max > k do
        for in 1:pop
          Select randomly two solutions a,b.
          Crossover(a,b): generates newsol.
          Mutation(newsol)
          Add newsol to pool
        end for
        Selection(pool)
        Set k = k + 1
end while
return Best solution found during the process
```

Fig. 7.6 Pseudo-code: genetic algorithm

and iterating over the population in a survival-of-the-fittest approach. The iterations make use of the following three elements:

Selection Criteria periodically the population of solutions is evaluated, and those that score poorly are discarded. A selection criterion, based on the evaluation function, is used to make the decision. Random discards or weight-based discards are also considered as options.

Genetic Operators solutions in a population will be used to generate new solutions that will be added to the population pool. The new solutions are generated through "cross-over" (using two solutions and generating a novel, "child" solution from them) and "mutation" (modifications of a solution). There are many ways to combine cross-over and mutations to generate new solutions. Once a new solution is generated it will also be evaluated and added to the pool of solutions.

Termination Criteria it is necessary to define when the algorithm will stop searching for new solutions. Termination criteria usually considered are (i) number n of iterations without improvement of a solution, (ii) finding a solution that reaches a lower boundary, (iii) maximum number of iterations, (iv) maximum running time or (v) a combination of any of the criteria (i)–(iv).

GA, as many other meta-heuristic algorithms, faces the challenge of setting a good trade-off between exploration and local optimization. Exploring is necessary when finding optimal solutions and it can be stressed by (a) including selection criteria that do not consider uniquely "best evaluated" solutions but enforce a heterogeneity in the solution pool; a second option, among others, is to (b) consider mutations that may lower or increase the evaluation score of a solution. The burden with high

heterogeneity is that the computational time required in the search will increase dramatically; for this reason it is important to limit exploration, and for this reason local optimization may help guide the search more rapidly. Importantly, "naïve GA" has been shown to have a premature convergence that prevents a global optimization [28]. Interestingly, there are hybrid methodologies combining GA with local optimum search exploiting the properties of GA without limiting its exploring ability [11, 25, 40].

It is worth considering that GA was originally conceived to solve discrete (or integer-based) optimization problems [28]. In the context of parameter estimation, binary representations of parameter values may solve the problem of representing parameter sets containing real numbers, but this option will also increase the number of solutions to investigate. Finally, GAs have developed into a wider area of optimization algorithms known as *Evolutionary computation* (EC) that are able to deal with continuous values [12].

7.4.5 Particle Swarm Optimization

EC algorithms find solutions through continuous optimization of a population of solutions. Within this setting Particle Swarm Optimization (PSO) is amongst the most well-known algorithms [32, 33] other relevant representatives are Ant Colony Optimization [13] or Genetic Programming [34].

Given a real parameter space, PSO first places a set of particles (agents) randomly in the parameter space, assigning them a random initial velocity. Every particle receives a fitness-score (the cost, V(P), delivered by the objective function) associated to the parameter set visited; those values are stored. Additionally the parameter set with the globally best fitness-score is also stored in **GBest**. Particle positions are updated iteratively based on a (randomly) weighted combination of: (1) the inertia of the particle, (2) a vector pointing from the present particle position to the best known position and (3) a vector pointing from the present particle position towards **GBest**. In this way particles share information of the best areas to search (social knowledge) but they also keep their best-visited solutions (cognitive knowledge). The iterative movement of particle *i* is explained by the following two equations:

$$vel\,(i) = \sigma\, vel\,(i) + r_1 * c_1\,(pos\,(i) - best\,(i)) + r_2 * c_2\,(pos\,(i) - GBest) \tag{7.18}$$

$$pos\,(i) = pos\,(i) + \chi\, vel\,(i) \tag{7.19}$$

pos(i) and *vel*(i) respectively denote the current position and the current velocity of particle i. The symbols c_1 and c_2 are integer non-negative values, named cognitive and social respectively, r_1, r_2 are real values drawn from [0,1], sigma and χ are non-negative real values, named inertia weight and constriction factor, respectively.

Pseudo-code Particle Swarm Optimization

Requirements: maximum number of iterations (It_{max}), number of particles (*nparticles*), maximum CPU, cognitive weight (*c*), social weight (*s*), inertia weight (*in*), constriction factor (*con*).

```
Initiate set of particles: randomly assign positions.
for i in particle_set do
    evaluate(i)
    best(i)=position(i)
end for
Set k = 1
GBest: position(i), that evaluate(i)=min(evaluate(i), i ∈ particle_set).
while It_max > k do
        for i in 1: nparticles do
          calculate velocity(i)
          update_position(i,GBest, best(i), velocity(i), c, s, in, con).
          if evaluate(i)<evaluate(best(i)) then
            best(i)= position(i)
          end if
          if evaluate(i)<evaluate(GBest) then
            GBest = position(i)
          end if
        end for
        k=k+1
end while
return GBest, evaluate(GBest)
```

Fig. 7.7 Pseudo-code: particle Swarm optimization

PSO has become a widely used methodology mainly because it is easy to implement, it is flexible and can be applied to most continuous-based problems [45] and requires little fine-tuning (few parameters to adjust) [33]. The caveat of PSO is that it is mainly limited to continuous problems (despite discrete to continuous mapping approaches [47]). The pseudo-code for PSO is found in Fig. 7.7.

7.5 Performance of Meta-Heuristics

Defining the difference between good and bad performance is essential when comparing algorithms. In accordance to the introduction, several parameter sets can give the same model output. Depending on the search purpose, different criteria for the performance exist. Denoting by a *run* the "*completion of the selected optimization*

algorithm", four possible criteria (among many others) to evaluate meta-heuristics are:

(1) **BS**: Given n runs, with limited time x, what is the quality of the best solution found in all the runs, i.e. what is the best cost found? BS provides a measure of how good is an algorithm searching for good solutions; however if the best solution is found once in 100 runs, that would require running the algorithm many times in order to find such a good solution.
(2) **BSa**: Given n runs, with limited time x, what is the average quality (average cost) of the solutions found, when pooling the best solution per run. BSa is useful when we may require algorithms that do not provide the best solutions (or lower BS) but on average perform better.
(3) **BSff**: similar to *BS* but instead of fixing to a limited time x, the run is fixed to a maximum number of objective-function evaluations.
(4) **BSaff**: similar to *BSa* but instead of fixing to a limited time x, the run is fixed to a maximum number of objective-function evaluations.

Several studies have approached the comparison of methodologies in different optimization problems. Unfortunately, there is no perfect optimization algorithm; given a performance evaluation, for each optimization methodology there is always a class of problems where the methodology will be worse than for other methodologies. That is, there is no method that will be optimal for all problems. This is a direct consequence of the No-Free-Lunch theorems presented in [59]. However, for making a selection among possible optimization algorithms, we recommend to follow two pieces of advice:

- *Hybrid version* while there is no "best-ever" algorithm for all types of optimization problems, it has been observed that hybrid methodologies, those combining ideas/methods from several optimization methodologies, tend to work robustly in all optimization problems. We recommend the implementation of hybrid versions.
- *Widely used* We can observe that there exist several optimization methods that are continuously implemented in parameter-fitting tools. For instance **Neurofitter** [56], developed to find parameter sets that able to reproduce experimental data in neuronal models, includes Random search, Particle Swarm Optimization, Evolution strategies, Multi-Start Local Optimization and combinations of them. Similarly, **COPASI** [29], an open source package that allows the generation and simulation of biological processes, includes Evolutionary Strategy, Genetic Algorithm(s), Particle Swarm Optimization, Random Search, and Simulated Annealing. This may reflect (1) the flexibility of those methods, (2) the low complexity to implement them, but also (3) that there are many successful examples of their applications.

7.6 Conclusions

The present chapter has provided a brief introduction to the field of optimization and its applications in the challenge of parameter estimation in biology. We have provided introductory explanations to some of the most widely used concepts and optimization algorithms. The focus of this chapter has been on meta-heuristic methods, within the field of stochastic global optimization. The benefit of these methods is that they can work on any optimization problem, and that they can deal with multi-modal systems: systems with multiple local optima. An outstanding question remains what algorithms are best in a specific situation. Herein, we have not really taken a stand on this issue, but nevertheless provide some tools and concepts that may be useful when comparing methods in different situations.

There are many implementations of these methods, and there are also many ways to implement models. Some of these model-formulation alternatives are standardized markup languages, such as SBML [9] and CellML [16], and some of these also have resources for parameter estimation. For instance, fitMatlabCellML has been developed for CellML. Similarly, sloppyCell [42], COPASI, and SBtoolbox [49] have been developed for SBML. However, for independent development and testing of these methods, it will always remain a tractable alternative to work with your own implementations of these algorithms.

In the following chapters, optimization tools are used to find parameter sets [38], and modified optimization algorithms are also used to generate a pool of "good-quality parameter sets", used for prediction uncertainty analysis [7, 21]. These slightly different settings will put new demands on future developments of optimization algorithms. All in all, it is therefore plausible that the development and evaluation of optimization algorithms will remain an important part of modelling in biology for the foreseeable future.

References

1. Allman, E.S., Rhodes, J.A.: Mathematical Models in Biology. Cambridge University Press, Cambridge (2003)
2. Banga, J.R.: Optimization in computational systems biology. BMC Syst. Biol. **2**, 47 (2008). doi:10.1186/1752-0509-2-47
3. Berg, J.M., Tymoczko, J.L., Stryer, L.: Biochemistry, 5th edn. W. H. Freeman, New York (2002)
4. Bishop, C.M.: Neural Networks for Pattern Recognition, 1st edn. Oxford University Press, New York (1996)
5. Brännmark, C., Palmér, R., Glad, S.T., Cedersund, G., Strålfors, P.: Mass and information feedbacks through receptor endocytosis govern insulin signaling as revealed using a parameter-free modeling framework. J. Biol. Chem. **285**(26), 20171–20179 (2010)
6. Britton, N.F.: Essential Mathematical Biology, p. 335. Springer, New York (2005)
7. Cedersund, G.: Prediction uncertainty estimation despite unidentifiability: an overview of recent developments. In: Uncertainty in Biology, A Computational Modeling Approach. Springer, Cham (2016, this volume)

8. Cedersund, G.: Conclusions via unique predictions obtained despite unidentifiability-new definitions and a general method. FEBS J. **279**(18), 3513–3527 (2012). doi:10.1111/j.1742-4658.2012.08725.x
9. Chaouiya, C., Berenguier, D., Keating, S.M., Naldi, A., van Iersel, M.P., Rodriguez, N., Dräger, A., Büchel, F., Cokelaer, T., Kowal, B., Wicks, B., Gonçalves, E., Dorier, J., Page, M., Monteiro, P.T., von Kamp, A., Xenarios, L., de Jong, H., Hucka, M., Klamt, S., Thieffry, D., Le Novère, N., Saez-Rodriguez, J., Helikar, T.: SBML Qualitative Models: a model representation format and infrastructure to foster interactions between qualitative modelling formalisms and tools BMC Syst. Biol. **7**(135), (2013)
10. De Jong, H.: Modeling and simulation of genetic regulatory systems: a literature review. J. Comput. Biol. **9**(1), 67–103 (2002)
11. De Jong K.: Genetic algorithms: a 30 year perspective. In: Booker, L., Forrest, S., Mitchell, M., Riolo, R. (eds) Perspectives on Adaptation in Natural and Artificial Systems, Oxford University Press, New York (2005)
12. De Jong, K.A.: Evolutionary computation: a unified approach. MIT Press, Cambridge (2006)
13. Dorigo, M., Birattari, M., Stützle, T.: Ant colony optimization–artificial ants as a computational intelligence technique. IEEE Comput. intell. Mag. **1**, 28–39 (2006)
14. Egea, J.: New heuristics for global optimization of complex bioprocesses. Dissertation, University de Vigo (2008)
15. Egea, J., Rodriguez-Fernandez, M., Banga, J., Martí, R.: Scatter search for chemical and bioprocess optimization. J. Glob. Optim. **37**(3), 481–503 (2007)
16. Garny, A., Nickerson, D., Cooper, J., Weber dos Santos, R., Miller, A.K., McKeever, S., Nielsen, P., Hunter, P.J.: CellML and associated tools and techniques. Philos. Trans. R. Soc. A **366**(1878), 3017–3043 (2008)
17. Gerstner, W., Sprekeler, H., Deco, G.: Theory and simulation in neuroscience. Science (New York, N.Y.) **338**(6103), 60–65 (2012). doi:10.1126/science.1227356
18. Glover, F.: Heuristics for integer programming using surrogate constraints. Decis. Sci. **8**(1), 156–166 (1977)
19. Glover, F., Kochenberger, G.A.: Handbook of metaheuristics 57. International Series in Operations Research & Management Science, Springer, New York (2003)
20. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning, 1st edn. Addison-Wesley Longman Publishing Co., Inc, Boston (1989)
21. Gomez-Cabrero, D., Ardid, S., Cano-Colino, M., Tegnér, J., Compte, A.: Neuroswarm: a methodology to explore the constraints that function imposes on simulation parameters in large-scale networks of biological neurons. In: Uncertainty in Biology, A Computational Modeling Approach. Springer, Cham (2016, this volume)
22. Gomez-Cabrero, D., Compte, A., Tegnér, J.: Workflow for generating competing hypothesis from models with parameter uncertainty. Interface Focus **1**(3), 438–449 (2011). doi:10.1098/rsfs.2011.0015
23. Greenberg, H.J., Hart, W.E., Lancia, G.: Opportunities for combinatorial optimization in computational biology. INFORMS J. Comput. **16**(3), 211–231 (2004)
24. Gustafsson, M., Hörnquist, M., Lundström, J., Björkegren, J., Tegnér, J.: Reverse engineering of gene networks with LASSO and nonlinear basis functions. Ann. N.Y. Acad. Sci. **1158**, 265–275 (2009). doi:10.1111/j.1749-6632.2008.03764.x
25. Hart, W.E.: Adaptive global optimization with local search. Doctoral Dissertation, University of California, San Diego (1994)
26. Hodgkin, A.L.: Chance and Design in Electrophysiology: An informal account of certain experiments on nerve carried out between 1934 and 1952. J. Physiol. **263**(I), 1–21 (1976)
27. Hodgkin, A.L., Huxley, A.F.: Currents carried by sodium and potassium ions through the membrane of the giant axon of Loligo This information is current as of January 29, This is the final published version of this article; it is available at: this version of the article may not be. J. Physiol. (Paris) **116**, 449–472 (1952)
28. Holland, J.H.: Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. University of Michigan Press, Ann Arbor (1975)

29. Hoops, S., Sahle, S., Gauges, R., Lee, C., Pahle, J., Simus, N., Singhal, M., Xu, L., Mendes, P., Kummer, U.: COPASI-a COmplex PAthway SImulator. Bioinformatics **22**(24), 3067–3074 (2006)
30. Hopfield, J.J.: Neural networks and physical systems with emergent collective computational abilities. Proc. Natl. Acad. Sci. U.S.A. **79**, 2554–2558 (1982)
31. Hübner, K., Sahle, S., Kummer, U.: Applications and trends in systems biology in biochemistry. FEBS J. **278**(16), 2767–2857 (2011)
32. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: Proceedings IEEE international conference on neural networks, Piscataway, pp. 1942–1948 (1995)
33. Kennedy J, Eberhart RC.: (2001) Swarm Intelligence. Morgan Kaufmann Publishers, Massachusetts (2001)
34. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge (1992)
35. Kirkpatrick, S., Gelatt Jr, C.D., Vecchi, M.: Optimization by simulated annealing. Science **220**(4598), 671–680 (1983)
36. Larranaga, P.: Machine learning in bioinformatics. Briefings Bioinf. **7**(1), 86–112 (2006). doi:10.1093/bib/bbk007
37. Li, Z., Li, P., Krishnan, A., Liu, J.: Large-scale dynamic gene regulatory network inference combining differential equation models with local dynamic Bayesian network analysis. Bioinformatics (Oxford, England) **27**(19), 2686–2691 (2011). doi:10.1093/bioinformatics/btr454
38. Mannakee, B.K., Ragsdale, A.P., Transtrum, M.K., Gutenkunst, R.N.: Sloppiness and the geometry of parameter space. In: Uncertainty in Biology, A Computational Modeling Approach. Springer, Cham (2016, this volume)
39. Markowetz, F.: How to understand the cell by breaking it: network analysis of gene perturbation screens. PLoS Comput. Biol. **6**(2), e1000655 (2010). doi:10.1371/journal.pcbi.1000655
40. Martinez-Estudillo, A., Hervas-Martinez, C., Martinez-Estudillo, F., Garca-Pedrajas, N.: Hybrid method based on clustering for evolutionary algorithms with local search. IEEE Trans. Syst. Man Cybernetics **34**(1) (2004)
41. Maza, M., Yuret, D.: Dynamic hill climbing: overcoming the limitations of optimization techniques. In: Proceedings of the 2nd Turkish Symposium on Artificial Inteligence and ANN, 1993
42. Myers, C.R., Gutenkunst, R.N., Sethna, J.P.: Python unleashed on systems biology. Comput. Sci. Eng. **9**, 34 (2007)
43. Nyman, E., Brännmark, C., Palmér, R., Brugård, J., Nyström, F.H., Strålfors, P., Cedersund, G.: A hierarchical whole-body modeling approach elucidates the link between in Vitro insulin signaling and in Vivo glucose homeostasis. J. Biol. Chem. **286**(29), 26028–26041 (2011)
44. Pettersson, T.: Global optimization methods for estimation of descriptive models. Master's thesis, Linköping University, Sweden (2008)
45. Poli, R.: Analysis of the publications on the applications of particle swarm optimisation. J. Artif. Evol. Appl. **2008**, 1–10 (2008)
46. Press, W., Teukolsky, S., Vetterling, W., Flannery, B.: Numerical Recipes in C, 2nd edn. Cambridge University Press, Cambridge (1992)
47. Roy, R., Dehuri, S., Cho, S.B.: A novel particle Swarm optimization algorithm for multi-objective combinatorial optimization problem. Int. J. Appl. Metaheuristic Comput. (IJAMC) **2**(4), 41–57 (2012)
48. Schlitt, T., Brazma, A.: Current approaches to gene regulatory network modelling. BMC Bioinform. **8**(Suppl 6), S9 (2007). doi:10.1186/1471-2105-8-S6-S9
49. Schmidt, H., Jirstrand, M.: Systems biology toolbox for MATLAB: a computational platform for research in systems biology. Bioinformatics **22**(4), 514–515 (2006)
50. Scholma, J., Schivo, S., Urquidi Camacho, R.,a, van de Pol, J., Karperien, M., Post, J.N.: Biological networks 101: computational modeling for molecular biologists. Gene **533**(1), 379–384 (2014). doi:10.1016/j.gene.2013.10.010
51. Segrè, D., Zucker, J., Katz, J., Lin, X., Haeseleer, P.D., Rindone, W.P., Church,G.M.: From Annotated Genomes to Metabolic Flux Models and Kinetic Parameter Fitting. OMICS J. Integr. Biol. **7**(3), 301–316 (2003)

52. Sejnowski, T.J., Koch, C., Churchland, P.S.: Computational Neuroscience. Science **241**, 1299–1306 (1987)
53. Skogsberg, J., Lundström, J., Kovacs, A., Nilsson, R., Noori, P., Maleki, S., Björkegren,J.: Transcriptional profiling uncovers a network of cholesterol-responsive atherosclerosis target genes. PLoS Genet. **4**(3), e1000036 (2008). doi:10.1371/journal.pgen.1000036
54. Swameye, I., Muller, T.G., Timmer, J., Sandra, O., Klingmuller, U.: Identification of nucleocytoplasmic cycling as a remote sensor in cellular signaling by databased modeling. Proc. Natl. Acad. Sci. **100**, 1028–1033 (2003)
55. Triki, E., Collette, Y., Siarry, P.A.: Theoretical study on the behavior of simulated annealing leading to a new cooling schedule. Eur. J. Oper. Res. **166**(1 SPEC. ISS.), 77–92 (2005)
56. Van Geit, W., Achard, P., De Schutter, E.: Neurofitter: a parameter tuning package for a wide range of electrophysiological neuron models. Front. Neuroinformatics **1**, 1 (2007). doi:10.3389/neuro.11.001.2007
57. Van Geit, W., De Schutter, E., Achard, P.: Automated neuron model optimization techniques: a review. Biol. Cybern. **99**(4–5), 241–251 (2008). doi:10.1007/s00422-008-0257-6
58. Van Riel, NW.: Dynamic modelling and analysis of biochemical networks: mechanism-based models and model-based experiments. Briefings Bioinf. **7**(4), 364–374 (2006). doi:10.1093/bib/bbl040
59. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Trans. Evol. Comput. **1**, 67–82 (1997)
60. Zimmer, D.C.: Parameter estimation for stochastic models of biochemical reactions. J. Comput. Sci. Syst. Biol. **6**, 011–021 (2012). doi:10.4172/jcsb.1000095