Hierarchical Optimization of Biochemical Networks

Nisha Ann Viswan, Alexandre Tribut, Manvel Gasparyan, Ovidiu Radulescu, Upinder S Bhalla

Hierarchical Optimization of Biochemical Networks

Nisha Ann Viswan[1,3], Alexandre Tribut[2,4], Manvel Gasparyan[2],
Ovidiu Radulescu[2,‡], Upinder S. Bhalla[1,‡]

[1] National Centre for Biological Sciences, Tata Institute of Fundamental Research, Bangalore, India,
[2] LPHI, University of Montpellier, CNRS, and INSERM, Montpellier, France,
[3] The University of Trans-Disciplinary Health Sciences and Technology, Bangalore, India,
[4] Ecole Centrale de Nantes, Nantes, France.
[‡] To whom correspondence should be addressed.

Abstract

Biological signalling systems are complex, and efforts to build mechanistic models must confront a huge parameter space, indirect and incomplete data, and frequently encounter multiscale and multiphysics phenomena. We present HOSS, a framework for Hierarchical Optimization of Systems Simulations, to address such problems. HOSS operates by breaking down extensive systems models into individual pathway blocks organized in a nested hierarchy. At the first level, dependencies are solely on signaling inputs, and subsequent levels rely only on the preceding ones. We demonstrate that each independent pathway in every level can be efficiently optimized. Once optimized, its parameters are held constant while the pathway serves as input for succeeding levels. We develop an algorithmic approach to identify the necessary nested hierarchies for the application of HOSS in any given biochemical network. Furthermore, we devise two parallelizable variants that generate numerous model instances using stochastic scrambling of parameters during initial and intermediate stages of optimization. Our results indicate that these variants produce superior models and offer an estimate of solution degeneracy. Additionally, we showcase the effectiveness of the optimization methods for both abstracted, event-based simulations and ODE-based models. We also present a technique leveraging abstracted modeling properties as a 'scaffold' to constrain the optimization of more detailed ODE models.

Keywords: systems biology, mechanistic models, optimization, modularity.

1 Introduction

Many large biochemical pathway models have been developed since the early days of systems biology. These models are manifold, including visual representations of data, such as protein interaction networks [1], and executable models like ordinary differential equations [2], Boolean models [3], and more recently, Hill-tau abstractions [4]. Among executable models, ODEs provide accurate representation of pathway dynamics, but incorporate many unknown parameters.

Two key advances have opened up the possibility of scaling up systems models substantially, in terms of complexity and reproducibility. First, there is now a rich ecosystem of data resources and data mining resources, both from structured databases and from the much broader but unstructured scientific literature. These approaches have already been used to scale up pathway diagrams and interaction networks [5]. Second, the advent of numerous high-throughput methods such as phosphoproteomics, imaging, and mass spectrometry promise far larger and internally consistent datasets (see [6]) than the extant patchwork of precise but once-off biochemical experiments performed by individual laboratories.

However, in spite of a few attempts [7, 8], the model development process is far from being automatic and standardized. In particular, parameter optimization methods were deployed in a piecemeal manner [9, 10, 11, 12]. This is in part due to the very wide diversity of experimental inputs

used to constrain such models, but also due to the inherent contradictions and incompleteness of the parameter constraints. For example, to compensate for the incompleteness of specific datasets and further constrain the model, it is not uncommon to amalgamate the findings from various publications. These data sources may utilize experimental preparations that differ significantly or even involve different classes of organisms [2]. However, this practice introduces inconsistent experimental inputs into the model. Consequently, this process is highly idiosyncratic, and different modelers may arrive at quite distinct models or parameter sets despite drawing on similar data sources.

There have been previous ambitious efforts to systematically funnel many experimental inputs into detailed and biologically driven models [13]. Such efforts require the integration of large-scale systematic data gathering with data management and modeling (e.g, SPEDRE [14]). The current paper focuses on standardizing the calibration and optimization stages of model development, given a large but incomplete set of experimental data. We build on our recently developed framework (FindSim [15]) for curating a very wide range of biochemical and physiological experiments, representing it in a consistent format, and using such curated experiment definitions to drive multiscale models. In principle, each new experiment should improve our understanding of biological systems, and thus help us to refine models of these systems. This amounts to a multi-parameter optimization problem. Its result should be a model that fits experiments as well as possible within the limitations of the model, while incorporating expert evaluation of the relative reliability of different experiments.

We introduce a novel methodology for solving the multi-parameter optimization problem. The methodology is based on the property of biochemical networks being modular, i.e., having groups of species and biochemical reactions that function autonomously given their input. Network modularity allows us to employ hierarchical optimization strategies. Inspired by game theory and now with multiple applications in science and engineering, hierarchical optimization decomposes a complex optimization problem into several coupled simpler problems [16, 17]. Although NP-hard in general, hierarchical optimization becomes easier for nested hierarchies, where lower levels depend on fewer parameters than the upper levels. We refer to such a method as nested hierarchical optimization and provide algorithmic solutions for implementing it in pathways.

We also report the development of an optimization pipeline, HOSS, implementing nested hierarchical optimization. We illustrate its use on an extant database of over 100 experiment definitions in the domain of synaptic signalling to improve the parameterization of a set of models of major signalling pathways involved in synaptic signalling and cell proliferation. HOSS utilizes FindSim in order to consistently evaluate models based on a specified set of experiments.

We show how our hierarchical approach addresses many of the challenges of parametric optimization problems, and outperforms a 'flat' optimization approach in efficiency, structure, and accuracy.

2 Methods

2.1 Mathematical formalism

2.1.1 Objective function

A popular choice of objective function is the log likelihood. For time series with normally distributed deviations, it reads

$$J(\theta, s, \sigma) = \frac{1}{2} \sum_{i,k} \left[\log\left(2\pi\sigma_i^2\right) + \left(\frac{y_{ik} - s_i x_i(t_k, \theta)}{\sigma_i} \right)^2 \right] \tag{1}$$

where y_{ik} and $x_i(t_k, \theta)$ are observed and predicted concentrations of the i^{th} observed species at the time t_k, respectively. θ are kinetic parameters. Parameters s_i are scaling parameters, accounting for the fact that the measurements are not absolute and σ_i are standard deviation parameters (see [18]).

Another popular choice is the least squares objective function:

$$LS(\theta, s) = \sum_{i,k} (y_{ik} - s_i x_i(t_k, \theta))^2 \tag{2}$$

Dose response or bar charts data [15] are modeled as steady states, in which case the time parameters are replaced by dose or qualitative parameters in (1) and (2).

2.1.2 Flat and Hierarchical optimization

Parameter optimization involves minimization of an objective function $f : S \subset \mathbb{R}^n \to \mathbb{R}$, where S is a space of constraints, $\boldsymbol{p} \in S$ a vector of parameters. The flat method consists of solving the problem:

$$\min_{\boldsymbol{p} \in S} f(\boldsymbol{p}). \tag{3}$$

There are many methods to solve (3). In our framework we use multistart optimization, by launching local search procedures from randomly chosen starting points generated uniformly in logarithmic scale:

$$\boldsymbol{p} = \tilde{\boldsymbol{p}} \exp(\log(\boldsymbol{a}) + \log(\boldsymbol{b})\boldsymbol{U}), \tag{4}$$

where $\tilde{\boldsymbol{p}}$ is a nominal guess, $\boldsymbol{U}$ random, independent variables whose distribution is uniform over $[0, 1]$ or standard normal, $\boldsymbol{a}$ and $\boldsymbol{b}$ are scale parameters. All the vector multiplications in (4) are elementwise. We refer to this procedure as *parameter scrambling*. Despite its simplicity, multistart optimization with logarithmic sampling has proven to be effective in benchmarks of biochemical pathways [19].

In hierarchical optimization [16], K sub-problems, each one defined by an objective function

$$f_i : S \longrightarrow \mathbb{R},\ i = 1, \ldots, K$$

are solved iteratively. Parameters of the problem are grouped in K groups $\boldsymbol{p} = (p_1, \ldots, p_K)$, where $p_i \in \mathbb{R}^{n_i}$ for $1 \leq i \leq K$ and $n = n_1 + \ldots + n_K$. We look for $\boldsymbol{p}^* = (p_1^*, \ldots, p_K^*) \in S \subset \mathbb{R}^n$, solution of

$$\begin{array}{ll}
\min\limits_{p_K} f_K(p_1^*, \ldots, p_{K-1}^*, p_K) & \text{where } p_{K-1}^* \text{ solves} \\
\min\limits_{p_{K-1}} f_{K-1}(p_1^*, \ldots, p_{K-2}^*, p_{K-1}, p_K) & \text{where } p_{K-2}^* \text{ solves} \\
\vdots & \vdots \\
\min\limits_{p_2} f_2(p_1^*, p_2, \ldots, p_K) & \text{where } p_1^* \text{ solves} \\
\min\limits_{p_1} f_1(p_1, p_2, \ldots, p_K) & p \text{ subject to } S.
\end{array} \tag{5}$$

The case $K = 2$ is known as bilevel optimization [17]. In this case the optimization of f_1 is called lower-level problem, whereas the optimization of f_2 is the upper-level problem.

The problem (5) is difficult, because each individual problem has to be solved for multiple values of the remaining variables, all subjected to the constraints S. Indeed, it has been proved that even apparently simple bilevel optimization problems are NP-hard [17]. However, the solution of bilevel optimization is straightforward if the lower-level problem has unique analytic solution. In this case, the naive algorithm, utilizes the solution of the lower-level problem to eliminate p_1 and reduce the upper-level optimization to minimizing a composed function that depends on p_2 only, is effective. Bilevel optimization with analytic solution for the lower-level problem has already been used for systems biology models. In this case the lower-level parameters are the scaling and standard deviation parameters that can be optimized by analytic formulas, see [18].

Another simple hierarchical optimization case is when the functions $f_i, 1 \leq i \leq K$ depend on nested sets of parameters and the set of constraints factorizes $S = S_1 \times S_2 \times \ldots \times S_K$. Then, (5) reads

$$\begin{array}{ll}
\min\limits_{p_K \in S_K} f_K(p_1^*, \ldots, p_{K-1}^*, p_K) & \text{where } p_{K-1}^* \text{ solves} \\
\min\limits_{p_{K-1} \in S_{K-1}} f_{K-1}(p_1^*, \ldots, p_{K-2}^*, p_{K-1}) & \text{where } p_{K-2}^* \text{ solves} \\
\vdots & \vdots \\
\min\limits_{p_2 \in S_2} f_2(p_1^*, p_2) & \text{where } p_1^* \text{ solves} \\
\min\limits_{p_1 \in S_1} f_1(p_1) &
\end{array} \tag{6}$$

We call the problem (6), **nested hierarchical optimization**. Nested hierarchical optimization can be solved iteratively, starting with the last problem in (6).

2.1.3 Nested hierarchical decompositions

A biochemical model is defined by a set of reactions $\mathcal{R}$ and a set of species $\mathcal{S}$. We also define the stoichiometric matrix $\boldsymbol{S}$, whose elements S_{ij} represent the number of molecules of the species i produced (if $S_{ij} > 0$) or consumed (if $S_{ij} < 0$) by the reaction j. Furthermore, the reaction rate vector $\boldsymbol{R}(\boldsymbol{x}, \boldsymbol{p}) = (R_1(\boldsymbol{x}, \boldsymbol{p}), \ldots, R_r(\boldsymbol{x}, \boldsymbol{p}))$ is a function of species concentrations $\boldsymbol{x} = (x_1, \ldots, x_N)$ and kinetic parameters $\boldsymbol{p}$. Each reaction j is characterized by a parameter vector $\boldsymbol{p}_j$, therefore we have $\boldsymbol{p} = (\boldsymbol{p}_1, \ldots, \boldsymbol{p}_r)$.

Species concentrations evolve in time as a result of chemical reactions. These define a semiflow (time dependent action on the species concentrations, enabling the computation of future concentrations based on the present ones) $\phi(t, \boldsymbol{x}; \boldsymbol{p}), t \geq 0$ such that $\boldsymbol{x}(t) = \phi(t, \boldsymbol{x}_0; \boldsymbol{p})$ represents the species concentration vector starting from initial values $\boldsymbol{x}(0) = \boldsymbol{x}_0$. The semiflow results from the integration of ODEs in chemical kinetics models or from the simulation of event driven dynamics in HillTau abstractions [4].

Some species forming a subset $BS \subset \mathcal{S}$ are buffered, and their concentrations are kept constant.

Our construction relies on the following concept. We call **autonomous pair**, a pair of reaction and species subsets $(I, J), I \subset \mathcal{S}, J \subset \mathcal{R}$ that satisfy:

1. if a species is in the subset I, then all the reactions consuming or producing this species are in the corresponding reaction subset J, namely if $i \in I$ then $j \in J$ whenever $S_{ij} \neq 0$.

2. if a reaction is in the subset J, then all the species on which the reaction rate depends are in the corresponding species subset I, unless these species are buffered, i.e. if $j \in J$ then $i \in I$ whenever $\frac{\partial R_j}{\partial x_i} \neq 0$ and $i \notin BS$.

Let $\boldsymbol{x}_I$ be the concentration vector of the species in I and $\boldsymbol{p}_J$ the kinetic constants of the reactions in J. From the above definition it follows that $\boldsymbol{x}_I$ can be computed at any positive time t by a semiflow depending only on the parameters $\boldsymbol{p}_J$, namely $\boldsymbol{x}_I(t) = \phi_I(t, \boldsymbol{x}_I(0); \boldsymbol{p}_J)$. Consider the data subset D_I, consisting of observations $\boldsymbol{y}_I$ of the species $\boldsymbol{x}_I$ only. Then objective function measuring the difference between observed and predicted values of $\boldsymbol{x}_I$ depends only on $\boldsymbol{p}_J$, namely

$$f_J(\boldsymbol{p}_J) = \sum_{i \in I} \sum_k (y_{ik} - \phi_i(t_k, \boldsymbol{x}_I(0); \boldsymbol{p}_J))^2. \tag{7}$$

Suppose now that we find species and reaction subsets,

$$\begin{aligned} I_1 \subset I_2 \subset \ldots \subset I_K = \mathcal{S}, \\ J_1 \subset J_2 \subset \ldots \subset J_K = \mathcal{R}, \end{aligned} \tag{8}$$

such that (I_k, J_k) are autonomous pairs for all $1 \leq k \leq K$.

We call (8) a nested hierarchical decomposition. Optimization of the objective functions (7) can then be done hierarchically, as in (6).

2.1.4 Constructing nested hierarchical decompositions using the interaction graph

Let us define the interaction digraph as $\mathcal{I} = (V, E)$, where V is the set of vertices (all species) and E is the set of edges. A pair of species $(i, j) \in E$ defines an edge from i to j if and only if there is a reaction that consumes or produces the species j, and its rate depends on the concentration of the species i. This graph is used to define causality relations between species, namely we say that j is causal to i, $j \rightsquigarrow i$, if j is connected to i by a path in $\mathcal{I}$. All the species j causal to i are needed for computing the time evolution of x_i.

Strongly connected components (SCC) of $\mathcal{I}$ are subsets $K \subset V$ such that $j \rightsquigarrow i$ and $i \rightsquigarrow j$ for all $i, j \in K$, maximal with respect to this property. In this paper, we refer to SCCs as **blocks**. Blocks form a partition of the species set I. This partition can be used to define a SCC quotient graph as follows: blocks are vertices of the SCC quotient graph, and two blocks are connected if there is one species in one block connected in the interaction graph to a species in the other block. The SCC quotient graph is always acyclic (see Figure 1 and [20]).

The following property is important for building nested hierarchical decompositions.

Property: Any block is either disjoint from or entirely contained within any subset I of an autonomous pair (I, J).

Thus, we build a nested hierarchical decomposition by using the blocks and the quotient graph. The first level subset I_1 is the union of roots of the quotient graph. The corresponding reaction subset J_1 is made of all reactions producing or consuming species from I_1. The next level I_2 is obtained by adding to the roots all the blocks directly connected to the roots, so far and so forth.

Although (8) does not define a unique decomposition, the decomposition resulting from the quotient graph is unique and has some advantages (minimality). For instance, I_1 is the minimal subset containing the roots, such that (I_1, J_1) is autonomous. I_2 is the minimal subset containing the roots and for each root that is not a sink, at least one species influenced by the root not in the root.

The quotient graph also provides a useful data structure for parallel optimization of the parameters. Thus, each tree originating from a root corresponds to terms in the objective function that can be optimized independently of the others.

2.1.5 r-blocks and feed-back in signaling networks

In some signaling pathway models, the signal does not propagate solely in the forward direction but also in reverse. This phenomenon can occur due to mainly two factors. This phenomenon mainly occurs due to two factors. First, downstream signaling molecules regulate upstream ones through feedback. Second, enzyme sequestration creates another form of feedback, which will be discussed in the next subsection. In such cases, it is not uncommon for all species to influence each other causally, resulting in a single block where hierarchical and plain optimizations are equivalent. Nevertheless, even in such scenarios, we can delineate smaller blocks.

Feedback regulation [21] can be neglected as a first approximation by imposing an upper limit on the length of paths connecting species in the interaction graph.

A species j is r-causal to a species i, $j \overset{r}{\rightsquigarrow} i$, if j is connected to i by a path in the interaction graph $\mathcal{I}$, of length smaller than or equal to r. A r-block is a subset K of species such that $i \overset{r}{\rightsquigarrow} j$ and $j \overset{r}{\rightsquigarrow} i$ for all $i, j \in K$, and which is maximal with respect to this property.

Because r-causality is not transitive, r-blocks are not disjoint in general. We therefore define r-strongly connected components (r-SCC) the sets obtained by lumping r-blocks that have non-empty intersection. The set of r-SCC is a partition of the set of species. Like the SCC partition, the r-SCC partition defines a quotient graph as follows: the nodes of the r-SCC quotient graph are r-SCC, and two r-SCC are connected if there is a species in one connected in the interaction graph to a species in the other. Contrary to SCC quotient graphs, r-SCC quotient graphs can contain cycles.

In order to construct hierarchies from the r-SCC quotient graph, one begins with roots. However, when there are cycles, roots cannot be solely defined using graph theory. In such instances, roots are determined based on biochemical criteria: the roots of the r-SCC quotient graph consist of species within the r-SCC that receive external signals. We also suppose that any r-SCC is reachable from at least one root. With this condition, hierarchies are defined like for the SCC quotient graph. The lowest level is made by root r-SCC, the next level is made by r-SCC reachable by one step in the r-SCC quotient graph, so far and so forth.

The application of this procedure to a signaling network with feedback is illustrated in the Figure 12. The choice of the value of r is made in function of the number of blocks (r-SCCs) obtained in the decomposition. For the purposes of hierarchical optimization we want to maximize the number of blocks. In practice, the decomposition is robust. The number of blocks is constant when r ranges from one to a critical value, and then decreases to a smaller, constant value for all r larger than or equal to the critical value (Figure 12).

2.1.6 Signal back-propagation and reduced Michaelis-Menten mechanisms

Enzyme sequestration appears in mass action biochemical networks when several Michaelis-Menten enzymatic mechanisms, explicitly represented as mass action elementary steps, share a common enzyme. This leads to back-propagation of the signal [22]. The back-propagation phenomenon disappears under quasi-steady state (QSS) conditions, when the enzyme-substrate complexes have low concentrations. As signalling pathways models often assume QSS, it is useful to have a tool that

reduces mass action models by eliminating complexes. The reduced models can then be decomposed into nested autonomous sets.

The algorithm performing QSS reduction is based on the following steps:

- First, detect all Michaelis-Menten mechanisms (Algorithm 6 in SM):

$$S_i + E_i \underset{k_i^-}{\overset{k_i^+}{\rightleftharpoons}} ES_i \xrightarrow{k_{\text{cat}}^i} P_i + E_i.$$

- Then, rewrite the model:
  - Let I_i the subset of reactions using the same enzyme E_i, i.e. $E_j = E_i,\ \forall j \in I_i$.
  - $\forall j \in I_i$ the Michaelis-Menten mechanism is replaced by a single reaction $S_j \overset{V_j}{\to} P_j$, with

$$V_j = k_{\text{cat}}^j SE_j = k_{\text{cat}}^j \frac{E_j^{\text{tot}} \frac{S_j}{k_m^j}}{1 + \sum\limits_{l \in I_i} \frac{S_l}{k_m^l}}, \tag{9}$$

    where

$$k_m^j = (k_j^- + k_{\text{cat}}^j)/k_j^+. \tag{10}$$

The algorithm, implemented in Python, handles `sbml` models and uses the `libsbml` library. We use this algorithm to transform mass action pathways into pathways that have non-trivial nested hierarchies. The resulting decomposition may be useful even if QSS conditions are not rigorously satisfied. In this case, hierarchical levels are autonomous only approximately, but hierarchical optimization may still be better than the plain optimization. Quasi-equilibrium (QE) reduction can be implemented using the same algorithm where (10) is changed to $k_m^j = k_j^-/k_j^+$.

Scoring the hierarchies Hierarchies can be scored to assess the quality of the decomposition. A rough score can be based, as discussed above, on the number of blocks. We have also defined a more advanced score, computed with the formula

$$S = \frac{N_{pairs} - N_{nested} + growth + smallest + \frac{1}{0.2 + N_{species} - N_{pairs}}}{N_{species}}$$

where

- N_{pairs} is the number of disjoint minimal autonomous pairs
- $N_{species}$ is the number of species in the model
- *smallest* is a score to check if the size of the smallest block is not too small or too big compared to the mean size. Here the mean size is $N_{species}/N_{pairs}$.
- *growth* is a score to check if the growth of autonomous pairs is uniform; it is equal to zero if the same amount of species is added from one pair to the next, for all pairs.
- N_{nested} is the number of autonomous pairs that are part of a nested hierarchy.
- the last term in the numerator is minimal when the number of pairs is small compared to the number of species and maximal when $N_{pairs} = N_{species}$.

A better hierarchy means a smaller score.

2.2 The HOSS Optimization framework

The HOSS software is designed to orchestrate complex, multi-level hierarchical optimizations. To do this it deploys numerous individual optimization steps, each of which fits a subset of a model to a number of individual experiments (Figure 2 A). HOSS works on signalling and other models which are subdivided into blocks, typically individual signalling pathways in a signalling network. The blocks are organized into a hierarchy where each level depends only on signalling input coming from preceding levels, and blocks within a level are independent of each other (Figure 2 B). The mathematical basis for this is elaborated upon in Subsection 2.1. HOSS reads a configuration file in JSON format, which specifies the metadata and overall optimization parameters, such as optimization algorithm and tolerance (Figure 2 B). The configuration file further specifies a weighted set of experiments, and parameters to tweak, for each block. HOSS calls the FindSim utility [15] to set the parameter vector, and obtain scores for model fit for each experiment. The default scoring for an individual experiment is done by taking the root-mean-square difference between experimental data and simulation readout and normalizing it by the range of data. Individual experiment scores in a block are then combined using normalized root-mean square of the experiment score scaled by weight. This consolidated score is used in the inner optimization algorithm which is provided by scipy.minimize. HOSS can employ nested parallelization by simultaneously running FindSim on each experiment within a block, and independently optimizing each block on different processes. For the purposes of subsequent discussion, we refer to the inner optimization routine (provided by scipy.optimize) as the optimization *algorithm*, and the outer hierarchical program (provided by HOSS) as the HOSS *method*.

FindSim is the Framework for Integration of Neuronal Data and SIgnalling Models [15]. Briefly, it does three things: 1) read a model and tweak its parameters, 2) read the definition of an experiment and run it on the model, and 3) compare the output of the model with data from an experiment (Figure 2 D, E, F). FindSim is agnostic to model definition format and simulator. It currently works with the HillTau [4] format and simulator, and with ODE and mass action models specified in SBML and other formats, and solved using the MOOSE simulator. FindSim utilizes a JSON format file in which to specify experiment inputs and readouts. Crucially, an experiment defined in FindSim format can be applied to completely different models even using different modelling formalisms, provided the input and output entities are common. We illustrate these capabilities below. In the context of HOSS, we use FindSim on four kinds of experiments applicable to cellular signalling: dose-response, time-series, bar-charts and direct parameter estimates (Figure 2 D, E, F). FindSim has additional capabilities to handle common electrophysiological experiments [23, 24] but these are not used in the current study.

It is useful to note that optimization can be parallelized at two levels: First, the objective function typically requires evaluation of multiple distinct experiments on a given subset of the model. This can be achieved by running multiple FindSim processes in parallel. Second, when a hierarchical level contains multiple blocks they can be independently optimized in parallel (Figure 8).

2.3 Large Models overview

For the purposes of this report, we model two signalling pathways in two formalisms each (Figure 3 A-D). The pathways are the beta-adrenergic receptor activation of protein kinase A (the b2AR pathway) and the epidermal growth factor activation of MAPK/ERKII (the EGFR pathway). The reaction topologies of these pathways are based on and simplified from [2]. The formalisms are HillTau [4], which is an abstracted reduced signalling model specification which maintains direct experimental mapping of selected parameters such as concentrations and rates, and well-mixed mass-action chemistry specified in the SBML format. The composition of the models is reported in Table 1.

Pathway	Formalism	# Species	# Reactions	# Parameters
EGFR-MAPK	HillTau	14	7	29
b2AR-PKA	HillTau	21	12	37
EGFR-MAPK	ODE	36	22	54
b2AR-PKA	ODE	53	40	93

Table 1: Composition of large test models used in this study.

2.4 Experimental database

We have used manual curation of the experimental literature to build up a repository of over 350 signalling experiments with a focus on synaptic signalling pathways. There are two key characteristics of this dataset, which drives several of the design choices in HOSS. First, the number of experiments pertaining to each pathway is limited, and considerably below the number of parameters even for HillTau models (Figure 4 A, B). Second, there are frequently overlapping experiments which disagree on the quantitative values of readouts (Figure 4 C, D).

3 Results

3.1 Hierarchical optimization outperforms flat optimization for a paradigmatic model with synthetic datasets

In order to illustrate and test the hierarchical optimization method we first use a paradigmatic model of the MAPK signaling cascade, introduced by Huang and Ferrel [25]. The SBML model is available in the Biomodels [26] database. The corresponding ODE system can be found in ODEbase [27] database https://www.odebase.org/detail/1330. The original SBML model consists of mass-action elementary reactions. Because of multiple Michaelis-Menten mechanisms sharing the same enzyme there is back-propagation of the signal and the application of the hierarchical decomposition algorithm to this model results in only one autonomous pair that includes the entire model.

By applying the QSS reduction transformation, the 22 ODEs in the ODEbase model are simplified to 8 differential equations. Notably, 4 species exclusively function as enzymes, and are considered buffered after the transformation (MAPKKK activator, MAPKKK inactivator, MAPKKPase, MAPKPase). As shown in Figure 5 A, the reduced MAPK model lends itself to a hierarchical cascade with 3 levels.

We have tested hierarchical optimization using time series produced in [28], consisting of 10 *in silico* experiments. Each experiment employs a different concentration of MAPKKK activator. For the flat optimization we used 12 distinct starting points log-uniformly distributed in a hypercube with edges $[10^{-10}, 10]$, and for hierarchical optimization (with parameter scrambling) we used 12 starting points per level. Figure 5 B shows that flat optimization takes longer compared to hierarchical optimization in terms of total duration. Additionally, the hierarchical optimization outperforms classical optimization significantly in terms of the objective function value (Figure 5 C).

3.2 The automatic generation of nested hierarchies scales well in extensive model repositories

To gain a better understanding of the nested hierarchy algorithm's performance and to explore the possibility of constructing hierarchies with other models, we applied the algorithm to 1058 manually curated models from the Biomodels database [26]. Most of the models have relatively good scores of the hierarchical decomposition obtained via the method described in Section **??**, or get good scores after reducing Michaelis-Menten mechanisms using the method of Section 2.1.6, Figure 6 A. The algorithm scales well for rather large models (up to 786 species) with sub-quadratic complexity in the average, Figure 6 B.

3.3 Black-box optimization methods work well for flat optimization.

To scale up our analysis to moderately large models, we utilized the HOSS pipeline on a set of four signaling pathways as described in Table 1. As a reference, we first ran the HOSS pipeline using flat (non-hierarchical) optimization on the models, employing a number of standard optimization methods in the scipy.minimize library (Figure 7 A). Our initial models were initially parameterized manually using inspection of a limited subset of experiments. Following the flat optimization, all of the algorithms produced better fitting models than the start models. This was reflected in modest improvement in model-fitting scores (Figure 7 B). We found that COBYLA (black-box, non-gradient algorithm based on linear approximation and programming) and SLSQP (iterative quasi-Newton algorithm based on quadratic programming sub-problems) were considerably faster to converge than gradient algorithms such as BFGS (Figure 7 C). COBYLA was more reliable in

producing good scores. Accordingly we used COBYLA for subsequent hierarchical optimization runs.

3.4 Hierarchical optimization is more efficient than flat optimization for biochemical models with real datasets

We next tested the HOSS pipeline for hierarchical optimization (Figure 8 A). We have shown in the methods section that nested hierarchical optimization is more efficient than flat optimization, and suggested that this efficiency may also carry over to cases which do not strictly obey the assumptions of autonomy between hierarchical levels. Here we test this for a set of substantial real-world cases, involving large models (Figure 3), large but incomplete datasets (Figure 4), and noisy and sometimes inconsistent data (Figure 4). We implemented hierarchical optimization in HOSS as per (Figure 8 A).

Additional details such as applicable experiments (FindSim files in JSON format), experiment weights, parameter lists, and parameter bounds were incorporated into the HOSS files. The signaling models from Figure 3 were manually subdivided into individual pathways, and placed in a hierarchy which reflected their position in the signaling cascade (e.g., Figure 2 C, Figure 8 B), and this layout was encoded in a HOSS configuration files for each model. We again tested three different inner algorithms for optimization: BFGS, COBYLA and SLSQP. We found that hierarchical optimization worked for all methods, though COBYLA gave better scores than BFGS and SLSQP for the b2AR models (Figure 8 B). The runtimes followed the same pattern as for flat optimization, that is, BFGS > COBYLA > SLSQP. We then compared how hierarchical optimization performed compared to flat optimization (Figure 8 D E) In the b2AR models, hierarchical optimization gave better scores. We speculate that a loop unrolling pass would improve the EGFR pathway scores, as we discuss below. Further, the runtime for hierarchical optimization was considerably faster in two cases, and similar in the other two.

3.5 Multistart methods yield better optimization score: initScram method.

As the basic HOSS algorithm may be susceptible to local minima, we implemented a version which generated a large number of initial models with parameters randomized in a log-normal distribution of width scramble Range (scramRange) (Figure 9 A, B). This is a known approach, with roots in simulated annealing methods [29, 30, 19]. We extended the HOSS framework to overlay model parameter scrambling and process farming onto the hierarchical optimization method. This is an embarrassingly parallel problem and each of the optimization processes could run in parallel. In the course of these runs we identified one necessary refinement to the algorithm. In some cases, a subset of the initial models took an enormously long time to converge. Thus we implemented a timeout for each elementary minimization run. This slightly reduces the number of completed runs, but may considerably reduce runtime. Multiple rounds of optimization tended to converge rapidly (Figure 9 C). This is similar to simulated annealing.

The optimization scores resulting from a typical run with 200 initial models fell into a distribution which depended both on model and on scramRange (Figure 9 D, E). Two effects were notable: The width of the score distribution increased with scramRange, and the peak of the distribution moved to the right (poorer scores), with increasing scramRange. The best fits were at the left of the distribution and typically were obtained with a scramRange of $\sim$2.0, that is, log-normal random scaling from 1/2 to 2-fold of each initial parameter (Figure 9 D, E). The scores for these fits were considerably better than those obtained with plain HOSS. Interestingly, the best few models (lowest scores) were not necessarily very similar in their parameters. To relate the NRMS divergence between parameters to scrambleRange, we generated a set of models at a series of scrambleRange values, and computed NRMS between each population (Figure 9 F). We did a normalized RMS comparison of parameters of the top 10 D4-b2AR models and found no obvious clusters (Figure 9 G). Using the relationship from (Figure 9 F), we observed that the NRMS range of $\sim$1.0, as seen in these best 10 models, corresponded to a scrambleRange of $\sim$2.0. As another measure of the parameter similarity of 'good' models, we plotted the distribution of (model parameter) / (mean parameter) across all parameters taken from the best 25% of models, that is, those whose scores were in the lowest quartile (Figure 9 H, I). We found that this clustered around one for scramble ranges of 2 and below, suggesting that there is indeed a common basin of attraction to which most models

converge. Note that this parameter distribution is narrower with a broad tail, as compared to the source model parameter distribution from (Figure 9 B).

3.6 Multi-stage Monte-Carlo yields further improvements of the optimization score: hossMC method.

As a final refinement of our code-base, we implemented a similar model-scrambling step within each stage of the HOSS algorithm (Figure 10 A). Thus, each subset of the model was subject to scrambling to give S variants (S$\sim$200 for a full run). These S variants were individually optimized in an elementary minimization step similar to a single stage in the original HOSS method (Figure 8 A). If there were multiple model subsets within a given level of the HOSS hierarchy, each was subject to this process to give S optimized variants. The best of each subset were then recombined so as to obtain the top N solutions for a given level. Typical values for N were $\sim$10. These top N sub-models were then used as separate starting points for further scrambled models for the next level of HOSS, such that we again had S variants to optimize. After the program ran through all levels, we had a set of the best-fitting N models obtained by the overall pipeline. This method generated excellent fits to the data (Figure 10 B) and wallclock time was similar to that of the initScram method provided there were enough CPU cores available to run all the steps in parallel (Figure 10 C). The total CPU time for both randomized methods was also quite similar (Figure 10 D).

To summarize the performance of the four methods employed here (flat, hoss, initScram and hossMC), we compared three metrics across the four optimization methods in the HOSS framework. The metrics were the final score (Figure 10 B), wallclock time (Figure 10 C), and total CPU time (Figure 10 D). As detailed above, the hossMC method is most effective but most CPU-costly, though the requirement for doing multiple iterations for the initScram method may lessen the gap. While the plain hoss method is quite fast, it is unlikely to be the method of choice because it did not come close to the best optimization score for any of our models. The conventional flat method is not a good choice for this kind of signaling model optimization.

3.7 Using reduced models to generate synthetic data improves optimization

One of the biggest challenges in our entire approach is that the models are usually underconstrained, as there are far fewer experiments than parameters (Figure 4 A, B). An outcome of this is seen in Figure 9 G, showing that optimized models with very similar and individually excellent scores may still differ over a 50% or greater range in multiple parameters. We noted that the scores for the reduced EGFR model (D3EGFR), which used HillTau formalism, were much better than for the ODE version (D4EGFR)(Figure 11 B). This led us to propose that we use synthetic experiments generated from the HillTau models to further constrain the behavior of the more complex ODE models of the same signaling pathway. In effect, we use a well-behaved reduced model as a scaffold to limit the dynamics of a closely mapped, but more complex ODE model. To accomplish this, we automatically generated a set of synthetic experiments to provide input-output mappings for each individual reduced reaction of the source HillTau model (Figure 11 A). In addition, we also generated input-output mappings where the input was the ligand at the starting level, and the readouts were output molecules at each level of the pathway (Figure 11 B). We generated two synthetic experiments for each input-output mapping: a time-series experiment in which the input molecule was elevated in a step function, and a dose-response experiment in which the input molecule was systematically varied over a range. We performed this procedure for the EGFR pathway where the ODE model had not converged well (Figure 11 B), using the D3-EGFR model as the source for generating synthetic experiments. We curated the synthetic experiment dataset to remove cases where the reduced model mechanism was inconsistent with the known chemical steps. We then redid the D4 level optimization using the synthetic experiments along with the real ones (Figure 11 C-F). We found that the addition of synthetic experiments improved scores and especially helped us ensure that signal propagation through multiple stages of a signaling pathway retained its dynamic range (Figure 11 F). Finally, we scored the resultant models on their performance against the original real experiments in the dataset (Figure 11 G). In other words, we were able to improve the optimization of a complex system through a hierarchical optimization across scales of granularity of models of

the same system.

4 Discussion

We have developed a pipeline for hierarchically optimizing large signalling models with hundreds of parameters. We show that hierarchical optimization gives better model fits, and does so faster than conventional flat optimization. We extend this approach to two further methods which use Monte Carlo sampling to give still better models, and which run in nearly the same time if there are sufficient parallel threads available. We further show that a hierarchy of model granularity may improve convergence in the common case of insufficient experimental data.

4.1 Model provenance and modelling disease variants

Complex biological models, and signaling models in particular, frequently draw upon diverse sources of data. Such models are typically hand-tuned, and such tuning may be very effective because it draws upon a great deal of intuition and implicit knowledge about the behaviour of familiar pathways. This does, however, make model provenance problematic. How did the modeller end up with a particular set of parameters? The HOSS framework introduces model fitting pipelines that are efficient, scalable, repeatable and above all, transparent. The development of a well-structured optimization configuration format in HOSS ensures that all experiment choices, their weights, and all parameter selections are as clearly defined as the algorithms and the simulators. This emphasis on provenance is designed to place the HOSS framework in line with FAIR principles[31]. We highlight two use cases to illustrate how HOSS supports reuse. First, model rederivation: A different scientist may feel that some of the original experiments should be considered more authoritative than others. This can be done simply by assigning a greater numerical weight to the selected experiments, rerunning the pipeline, and seeing what changes in the resultant optimized model. Similarly, a researcher could include some new experiments into the dataset against which the model is to be optimized. This simplicity of model derivation brings a more data-driven flavor to debates over model assumptions and how well they represent the known experimental literature. As HOSS is agnostic to model formalism, it follows that these comparisons could even extend over distinct models implemented with different formalisms (e.g., HillTau vs mass action chemistry).

Second, The HOSS structure is highly effective for model specialization: A researcher may wish to make a family of models for different disease mutations, based on a dataset of readouts for experiments in a set of mutant animal or cell lines. Using the HOSS pipeline, it is straightforward to replace the original (wild-type) experiments with the respective mutant line experiments, rerun the optimization, and obtain disease-specific models. Thus the HOSS framework encourages best practice in developing complex models which can be easily reused.

4.2 Large models and large datasets

HOSS is scalable. This is in large part due to the efficiency of the hierarchical optimization core method we have described. Based on this, we have shown that even large models can be optimized quickly. Further, the approach of model decomposition into a hierarchy has informed the design of the HOSS framework. HOSS organizes models into hierarchies, within which data, parameter choices, and multiple optimization stages of a pipeline can be triggered using a single command. Thus, once it is set up, an optimization run does not require many steps of inspection and tweaking by the investigator, and is limited only by computational resources. Several tools also provide model optimization (e.g., COPASI [29]) Model building is not limited just by resources and datasets, but also by how manageable is the organization of the dataset. The traditional way to associate model parameters with experiments is to provide citations (e.g., refs: DOQCS, BioModels). This is not scalable - every iteration of the model would in principle require human intervention. Several efforts have sought to digest such experimental citations into a machine-readable form (refs), and HOSS uses the FindSim format to do so (refs). The organization of a HOSS pipeline lends itself to version control, since every component of the pipeline is a file in a standard location and standard format. We have scaled HOSS pipelines to refine a suite of models of activity-driven synaptic signalling pathways at different levels of granularity, covering over 40 pathways and 400 experiments (refs).

These take from 20 to 48 hours to optimize on a 128-core server using the initScram method. We anticipate an influx of phosphoproteomics and other large datasets (e.g, Erin Shuman's recent work), which will substantially increase the size of the experimental database. HOSS encourages the clear subdivision of models and experiments into groupings around individual signalling steps, such as the activation of a kinase by its immediate second messengers. This has implications for experimental design geared to tightly defining large signalling systems, because it lays out the kinds of experiments that are needed to achieve full coverage of all the reaction steps. Specifically, we find that two kinds of experiments can greatly tighten parameters: local experiments that probe input-output properties of a given signalling step, and readouts of all key intermediates along a receptor-driven pathway to ensure that signal propagation remains intact (Figure xxyy).

4.3 Model degeneracy, granularity, and completeness

In electrophysiological models, it is now clear that many parameter combinations may yield the same input-output properties. The origins of this degeneracy could be epistemic and due to data incompleteness [32], but also biological being a feature of neurons themselves (refs: Prinz, De Schutter, Turrigiano). The HOSS framework may provide a useful tool to study such degeneracy, especially in signalling. Most directly, the two Monte Carlo methods supported by HOSS (initScram and hossMC) generate multiple 'good' models, which can be tested for degeneracy (e.g., Figure 9). Because HOSS is agnostic to model detail, simulator, and model formalism, it also lends itself to asking how model granularity affects degeneracy. We illustrate this capability in Figure 9, where we find that for the two sample signalling systems, the more abstract model (HillTau formalism) is indeed less degenerate than the mass-action one, suggesting that the origin of parameter uncertainty is epistemic at least partially (for the same data, reduced models are more constrained than detailed models). We have previously suggested that it is useful to develop a family of models at different resolution for any given signalling system [33]. HOSS is well equipped to facilitate this, as it can use the same experimental dataset for models at different detail. Model completeness is quite difficult to ascertain in biology as it is in all scientific fields confronting theory and experiments [34]. Several methods have attempted to explore model topology space along with parameters [35, 36, 8, 7], but HOSS supports a more pragmatic interpretation: Is a model complete enough to account for a given set of observations? It does so by trying a large number of possible parameter sets and seeing whether any of these initial conditions result in well-fitting models. A failure to do so suggests that the model topology may need to be reconsidered. In our examples above, we find that our reduced kinase pathway models in HillTau formalism are able to fit the experiments almost as well as the more detailed mass-action models (Figure 8). Thus they are as complete, by this definition. We have previously illustrated the behavior of a series of models of activity-driven synaptic signalling at different levels of granularity, and show that more detailed models fit additional features of the response (ref HillTau paper [4]).

Code availability

Code locations: HOSS https://github.com/upibhalla/HOSS, FindSim https://github.com/BhallaLab/FindSim, HillTau https://github.com/BhallaLab/HillTau, MOOSE https://github.com/BhallaLab/moose-core.

Acknowledgments

This work has been supported by CEFIPRA grant 68T08-3 to OR and USB NAV has been supported by the Department of Biotechnology, Government of India, under the fellowship No. DBT/2018/NCBS/998. USB and NCBS-TIFR receive the support of the Department of Atomic Energy, Government of India, under Project Identification No. RTI 4006 Pawan Kumar for help with supervision of AT.

Author contributions

NV developed the database of experiments, and provided initial models for the b2AR and EGFR pathways. AT developed the algorithms and wrote the codes for nested hierarchical composition and QSS reduction, analysed the MAPK example and benchmarked early versions of the hierarchical decomposition on Biomodels database. MG implemented the algorithm for automatic hierarchical decomposition based on r-SCC. OR developed the mathematical framework, designed the research, and wrote the paper. USB designed the research, implemented the code for the HOSS framework, ran the simulations, and wrote the paper.

References

[1] Kanehisa, M. and Goto, S. (2000) KEGG: kyoto encyclopedia of genes and genomes. *Nucleic acids research,* **28**(1), 27–30.

[2] Bhalla, U. S. and Iyengar, R. (1999) Emergent properties of networks of biological signaling pathways. *Science,* **283**(5400), 381–387.

[3] Abou-Jaoudé, W., Traynard, P., Monteiro, P. T., Saez-Rodriguez, J., Helikar, T., Thieffry, D., and Chaouiya, C. (2016) Logical modeling and dynamical analysis of cellular networks. *Frontiers in genetics,* **7**, 94.

[4] Bhalla, U. S. (2021) HillTau: A fast, compact abstraction for model reduction in biochemical signaling networks. *PLoS Computational Biology,* **17**(11), e1009621.

[5] Gyori, B. M. and Bachman, J. A. (December, 2021) From knowledge to models: Automated modeling in systems and synthetic biology. *Current Opinion in Systems Biology,* **28**, 100362.

[6] Nyman, E., Stein, R. R., Jing, X., Wang, W., Marks, B., Zervantonakis, I. K., Korkut, A., Gauthier, N. P., and Sander, C. (2020) Perturbation biology links temporal protein changes to drug responses in a melanoma cell line. *PLoS computational biology,* **16**(7), e1007909.

[7] Büchel, F., Rodriguez, N., Swainston, N., Wrzodek, C., Czauderna, T., Keller, R., Mittag, F., Schubert, M., Glont, M., Golebiewski, M., van Iersel, M., Keating, S., Rall, M., Wybrow, M., Hermjakob, H., Hucka, M., Kell, D. B., Müller, W., Mendes, P., Zell, A., Chaouiya, C., Saez-Rodriguez, J., Schreiber, F., Laibe, C., Dräger, A., and Le Novère, N. (November, 2013) Path2Models: large-scale generation of computational models from biochemical pathway maps. *BMC Systems Biology,* **7**(1), 116.

[8] Gyori, B. M., Bachman, J. A., Subramanian, K., Muhlich, J. L., Galescu, L., and Sorger, P. K. (November, 2017) From word models to executable models of signaling networks using automated assembly. *Molecular Systems Biology,* **13**(11), 954 Publisher: John Wiley & Sons, Ltd.

[9] Banga, J. R. (2008) Optimization in computational systems biology. *BMC systems biology,* **2**(1), 1–7.

[10] Chou, I.-C. and Voit, E. O. (2009) Recent developments in parameter estimation and structure identification of biochemical and genomic systems. *Mathematical biosciences,* **219**(2), 57–83.

[11] Kravaris, C., Hahn, J., and Chu, Y. (2013) Advances and selected recent developments in state and parameter estimation. *Computers & chemical engineering,* **51**, 111–123.

[12] Loskot, P., Atitey, K., and Mihaylova, L. (2019) Comprehensive review of models and methods for inferences in bio-chemical reaction networks. *Frontiers in genetics,* p. 549.

[13] Gilman, A. G., Simon, M. I., Bourne, H. R., Harris, B. A., Long, R., Ross, E. M., Stull, J. T., Taussig, R., Arkin, A. P., Cobb, M. H., et al. (2002) Overview of the alliance for cellular signaling. *Nature,* **20**, 703–706.

[14] Nim, T. H., White, J. K., and Tucker-Kellogg, L. (2013) SPEDRE: a web server for estimating rate parameters for cell signaling dynamics in data-rich environments. *Nucleic acids research,* **41**(W1), W187–W191.

[15] Viswan, N. A., HarshaRani, G. V., Stefan, M. I., and Bhalla, U. S. (2018) FindSim: a framework for integrating neuronal data and signaling models. *Frontiers in neuroinformatics,* **12**, 38.

[16] Anandalingam, G. and Friesz, T. (December, 1992) Hierarchical optimization: An introduction. *Annals of Operations Research,* **34**(1), 1–11.

[17] Liu, R., Gao, J., Zhang, J., Meng, D., and Lin, Z. (2021) Investigating Bi-Level Optimization for Learning and Vision From a Unified Perspective: A Survey and Beyond. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* **44**, 10045–10067.

[18] Loos, C., Krause, S., and Hasenauer, J. (07, 2018) Hierarchical optimization for the efficient parametrization of ODE models. *Bioinformatics,* **34**(24), 4266–4273.

[19] Villaverde, A. F., Fröhlich, F., Weindl, D., Hasenauer, J., and Banga, J. R. (2019) Benchmarking optimization methods for parameter estimation in large kinetic models. *Bioinformatics,* **35**(5), 830–838.

[20] Bloem, R., Gabow, H. N., and Somenzi, F. (2000) An algorithm for strongly connected component analysis in n log n symbolic steps. In *International Conference on Formal Methods in Computer-Aided Design* Springer pp. 56–73.

[21] Bhalla, U. S., Ram, P. T., and Iyengar, R. (2002) MAP kinase phosphatase as a locus of flexibility in a mitogen-activated protein kinase signaling network. *Science,* **297**(5583), 1018–1023.

[22] Ventura, A. C., Sepulchre, J.-A., and Merajver, S. D. (2008) A hidden feedback in signaling cascades is revealed. *PLoS computational biology,* **4**(3), e1000041.

[23] Bhalla, U. S. and Bower, J. M. (1993) Exploring parameter space in detailed single neuron models: simulations of the mitral and granule cells of the olfactory bulb. *Journal of neurophysiology,* **69**(6), 1948–1965.

[24] Brown, S.-A., Moraru, I. I., Schaff, J. C., and Loew, L. M. (2011) Virtual NEURON: a strategy for merged biochemical and electrophysiological modeling. *Journal of computational neuroscience,* **31**, 385–400.

[25] Huang, C. Y. and Ferrell, J. E. (September, 1996) Ultrasensitivity in the mitogen-activated protein kinase cascade. *Proceedings of the National Academy of Sciences,* **93**(19), 10078–10083 Publisher: National Academy of Sciences Section: Research Article.

[26] Malik-Sheriff, R. S., Glont, M., Nguyen, T. V. N., Tiwari, K., Roberts, M. G., Xavier, A., Vu, M. T., Men, J., Maire, M., Kananathan, S., Fairbanks, E. L., Meyer, J. P., Arankalle, C., Varusai, T. M., Knight-Schrijver, V., Li, L., Dueñas-Roca, C., Dass, G., Keating, S. M., Park, Y. M., Buso, N., Rodriguez, N., Hucka, M., and Hermjakob, H. (1, 2020) BioModels — 15 years of sharing computational models in life science. *Nucleic Acids Research,* **48**(D1), D407–D415 gkz1055.

[27] Lüders, C., Sturm, T., and Radulescu, O. (April, 2022) ODEbase: A Repository of ODE Systems for Systems Biology. *Bioinformatics Advances,* **2**(1) vbac027.

[28] Henriques, D., Villaverde, A. F., Rocha, M., Saez-Rodriguez, J., and Banga, J. R. (02, 2017) Data-driven reverse engineering of signaling pathways using ensembles of dynamic models. *PLOS Computational Biology,* **13**(2), 1–25.

[29] Elsts, A., Pentjuss, A., and Stalidzans, E. (2017) SpaceScanner: COPASI wrapper for automated management of global stochastic optimization experiments. *Bioinformatics,* **33**(18), 2966–2967.

[30] Dai, Z. and Lai, L. (2014) Differential simulated annealing: a robust and efficient global optimization algorithm for parameter estimation of biological networks. *Molecular BioSystems,* **10**(6), 1385–1392.

[31] Eriksson, O., Bhalla, U. S., Blackwell, K. T., Crook, S. M., Keller, D., Kramer, A., Linne, M.-L., Saudargienė, A., Wade, R. C., and Hellgren Kotaleski, J. (2022) Combining hypothesis-and data-driven neuroscience modeling in FAIR workflows. *Elife,* **11**, e69013.

[32] Hüllermeier, E. and Waegeman, W. (2021) Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine Learning,* **110**, 457–506.

[33] Viswan, N. A. and Bhalla, U. S. (2023) Understanding molecular signaling cascades in neural disease using multi-resolution models. *Current Opinion in Neurobiology,* **83**, 102808.

[34] Hofman, J. M., Watts, D. J., Athey, S., Garip, F., Griffiths, T. L., Kleinberg, J., Margetts, H., Mullainathan, S., Salganik, M. J., Vazire, S., et al. (2021) Integrating explanation and prediction in computational social science. *Nature,* **595**(7866), 181–188.

[35] Flöttmann, M., Schaber, J., Hoops, S., Klipp, E., and Mendes, P. (2008) ModelMage: a tool for automatic model generation, selection and management. *Genome Informatics,* **20**, 52–63.

[36] Fröhlich, F., Loos, C., and Hasenauer, J. (2019) Scalable inference of ordinary differential equation models of biochemical processes. *Gene regulatory networks: methods and protocols,* pp. 385–422.

[37] Mukhin, Y. V., Garnovsky, E. A., Ullian, M. E., and Garnovskaya, M. N. (2003) Bradykinin B2 receptor activates extracellular signal-regulated protein kinase in mIMCD-3 cells via epidermal growth factor receptor transactivation. *Journal of Pharmacology and Experimental Therapeutics,* **304**(3), 968–977.

[38] Saito, T., Okada, S., Ohshima, K., Yamada, E., Sato, M., Uehara, Y., Shimizu, H., Pessin, J. E., and Mori, M. (2004) Differential activation of epidermal growth factor (EGF) receptor downstream signaling pathways by betacellulin and EGF. *Endocrinology,* **145**(9), 4232–4243.

[39] Kholodenko, B. N., Demin, O. V., Moehren, G., and Hoek, J. B. (1999) Quantification of short term signaling by the epidermal growth factor receptor. *Journal of Biological Chemistry,* **274**(42), 30169–30181.

[40] Solberg, R., Taskén, K., Wen, W., Coghlan, V. M., Meinkoth, J. L., Scott, J. D., Jahnsen, T., and Taylor, S. S. (1994) Human regulatory subunit RIβ of cAMP-dependent protein kinases: expression, holoenzyme formation and microinjection into living cells. *Experimental cell research,* **214**(2), 595–605.

[41] Wolter, S., Golombek, M., and Seifert, R. (2011) Differential activation of cAMP-and cGMP-dependent protein kinases by cyclic purine and pyrimidine nucleotides. *Biochemical and biophysical research communications,* **415**(4), 563–566.

[42] Hasler, P., Moore, J. J., and Kammer, G. M. (1992) Human T lymphocyte cAMP-dependent protein kinase: subcellular distributions and activity ranges of type I and type II isozymes. *The FASEB journal,* **6**(9), 2735–2741.

Figures

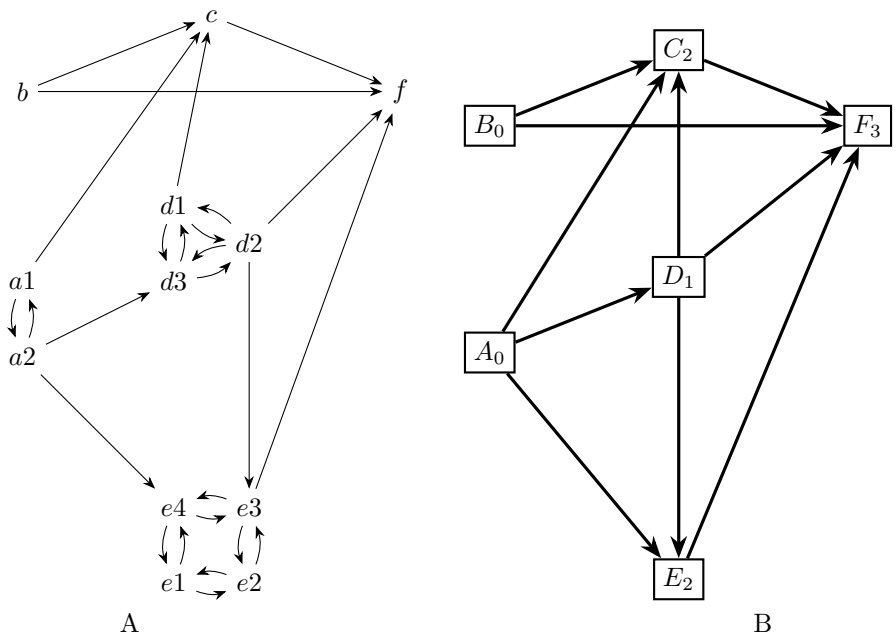

Figure 1: Interaction and block graphs define the nested hierarchical decomposition. A) Interaction graph, whose vertices are biochemical species. One source species acts on a target species if there is a reaction consuming or producing the target, whose rate depends on the concentration of the source. The blocks are strongly connected components of the interaction graph. B) The block graph is an acyclic directed graph, whose vertices are the blocks. The hierarchy level of a block is the length of the longest path on the block graph, starting from the roots. In this example, blocks A, B are roots and have level 0, block D has level 1, blocks C, E have level 2 and F have level 3.

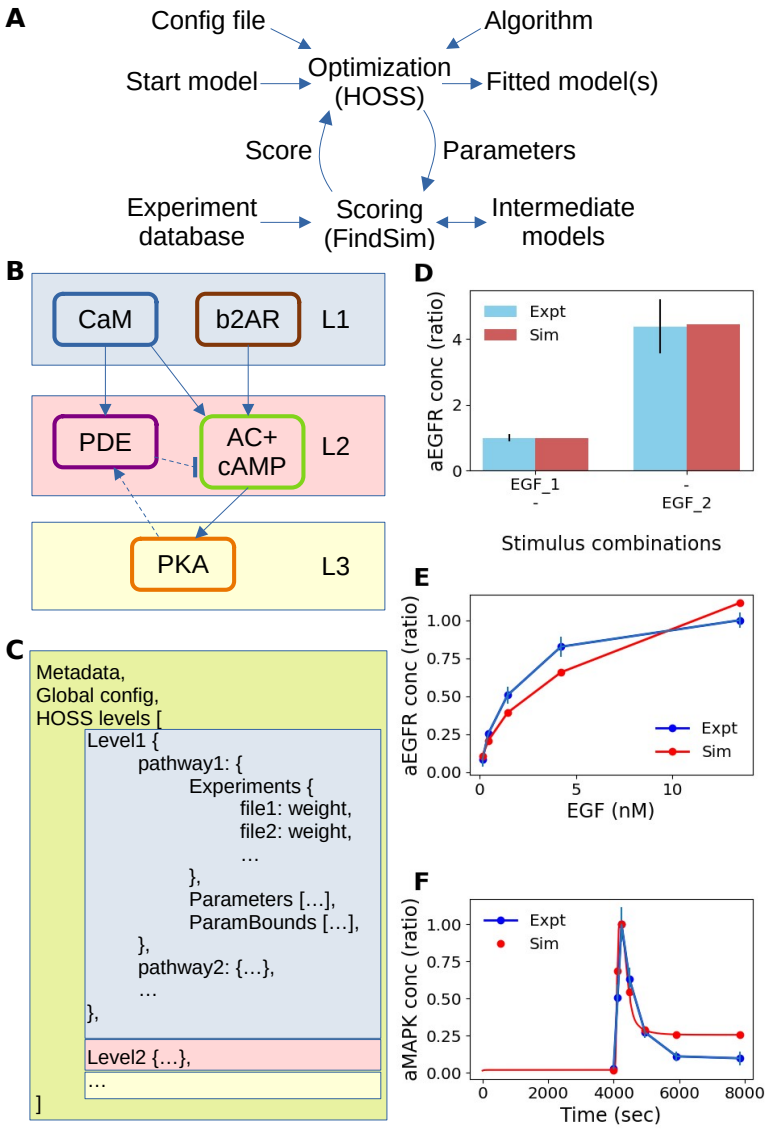

Figure 2: Optimization framework.
A: Schematic of optimization pipeline. HOSS follows a pipeline defined in a JSON configuration file to apply the specified algorithm at each stage of the optimization. HOSS orchestrates the operations of FindSim which takes a model, modifies its parameters, runs the model against specified experiments, and returns a score. This score is used by the algorithm. B: Typical model decomposition into levels. L1 depends only on inputs, L2 depends only on L1 and inputs, and L3 depends on all upstream pathways. Within a level we can have multiple signalling events provided they do not depend on each other. However, we may have cross-interactions or feedback (arrows with dashed lines), which may require the pipeline to repeat one or more levels. C: pseudocode for definition of the HOSS pipeline. Within each level we can have multiple pathways, each of which needs a list of experiments, parameters and optionally parameter bounds. Colors map to corresponding levels of the model from panel B. D, E, F: Typical examples of experiments defined in FindSim format and run using FindSim to obtain scores for goodness of fit between experiment and simulation. In all these cases EGF is used as an input to the EGFR pathway. D: Bar chart. Here EGF is provided at baseline level (0.1 nM, named EGF_1) and at stimulus level (1.5625 nM, named EGF_2), and the resultant level of activated EGF receptor (aEGFR) is found. E: Dose-response. Here EGF is provided at a series of fixed input levels, and the steady-state levels of aEGFR are measured F: Time-series. Here a 7.8125 nM step stimulus of EGF is applied at t=4000s, and the level of activated MAPK is read out.

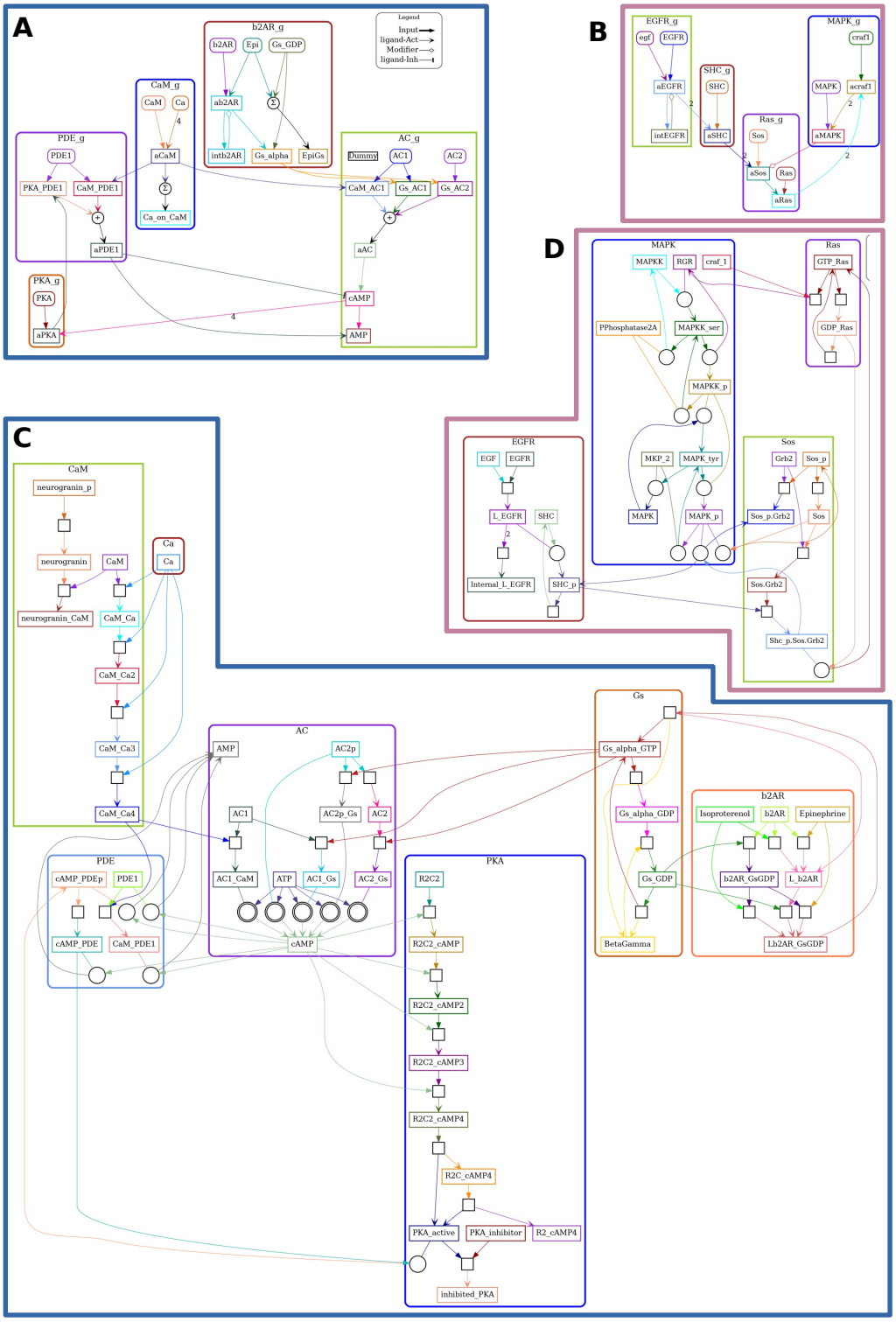

Figure 3: Models used in current study. A: Beta-2 adrenergic receptor pathway leading to Protein Kinase A activation (b2AR pathway), implemented in HillTau format. B: Epidermal growth factor receptor pathway leading to Mitogen-Activated Protein Kinase activation (EGFR pathway), implemented in HillTau format. C: b2AR pathway implemented in ODE format compatible with SBML. D: EGFR pathway implemented in ODE format. Note that the ODE format implementations are more chemically detailed, but do retain overlap with HillTau implementations for several key readouts.

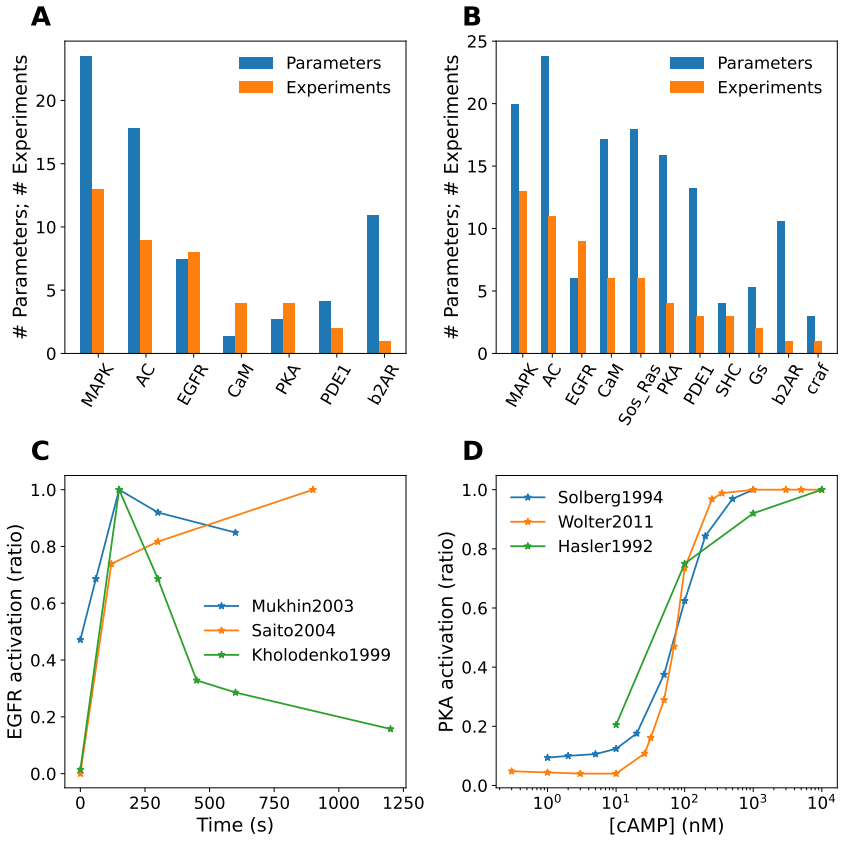

Figure 4: Features of experimental database. A, B: Number of parameters (blue) and number of experiments (orange) to constrain them, for different blocks in the model, sorted in order of decreasing number of experiments. In almost all cases the number of experiments falls well short of the number of parameters, that is, the model is underconstrained. A: Reduced (HillTau) models. B: ODE (SBML) models. C, D: Experiments may be inconsistent. C: Three time-series experiments for EGFR activation following a pulse of EGF, normalized to maximal response. These experiments were performed on different cell lines and not surprisingly, the time-courses differ [37, 38, 39]. D: Three dose-response experiments for PKA activation by cAMP. These experiments use purified preparations and despite somewhat different conditions the Kd is quite similar [40, 41, 42].

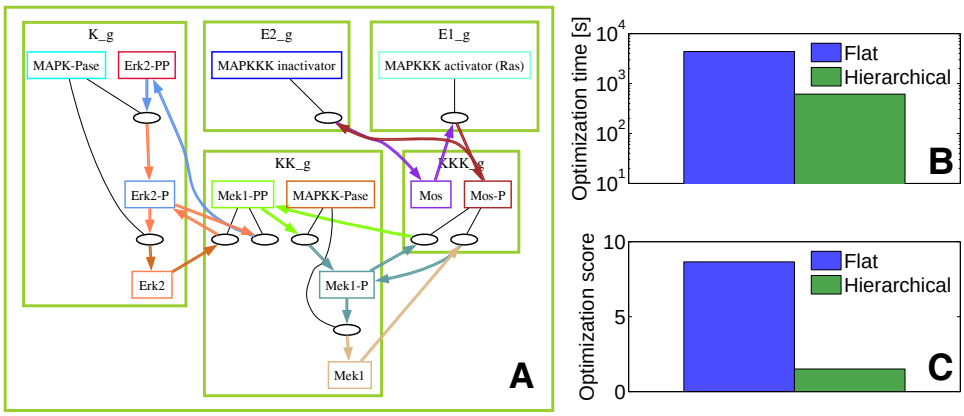

Figure 5: Optimization of the reduced MAPK model. A) Blocks in the reduced model; a three level nested hierarchy is defined as follows: level 1 (E1_g, E2_g, KKK_g), level2 (E1_g, E2_g, KKK_g, KK_g), level3 (E1_g, E2_g, KKK_g, KK_g, K_g). B,C) Performance of flat and hierarchical optimization.

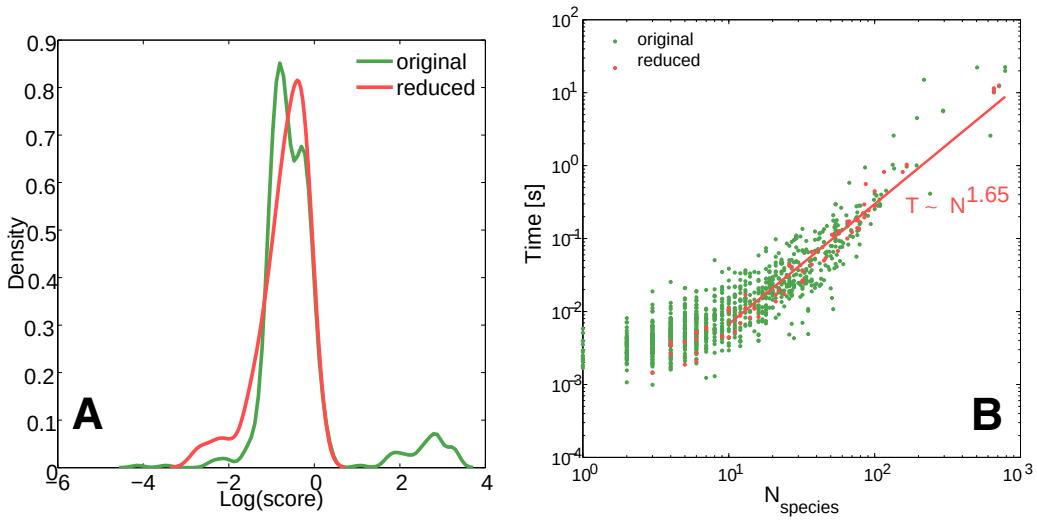

Figure 6: Benchmarking the hierarchical decomposition algorithm using 1058 models from the Biomodels repository [26]. A) The probability density of the decomposition score shows that QSS reduction of models with enzymes shared between multiple substrates improves the score. B) The algorithm has sub-quadratic complexity in the average.

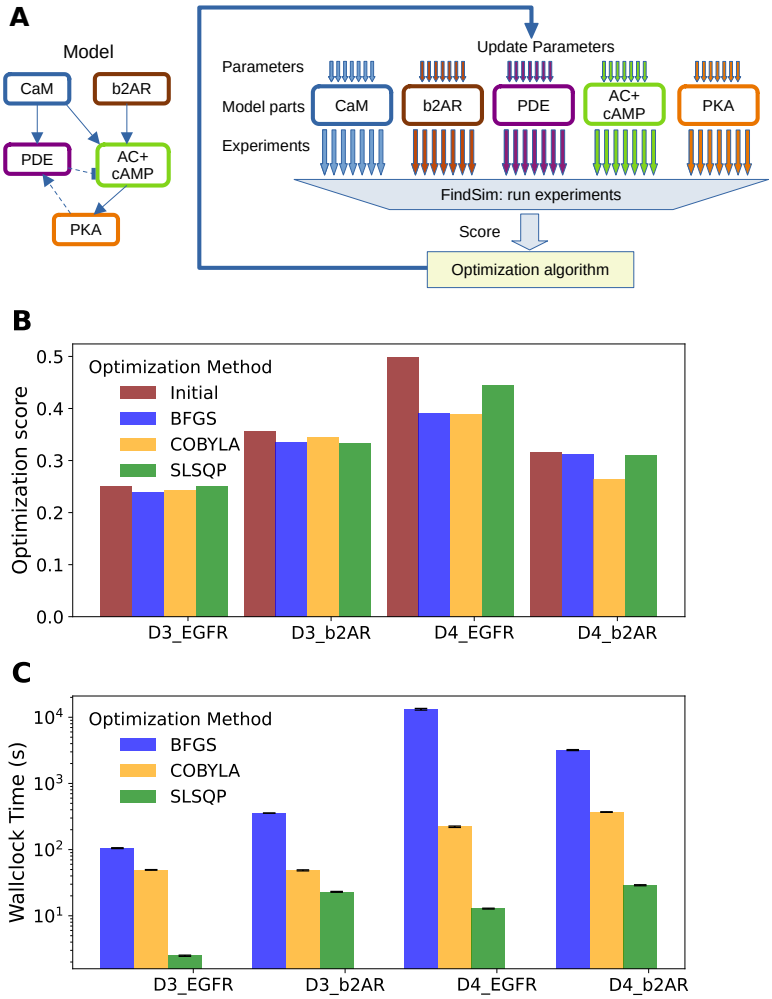

Figure 7: Flat optimization method. A: Schematic of flat optimization: Left: Model. Right: Algorithim. All model subsets and all experiments are run in parallel. The resulting scores are combined into a single value used by the optimizer. The optimizer adjusts all parameters across model subsets, for each iteration. B: Barchart of scores for the four different models, comparing the initial score with the final score obtained using three different algorithms (BFGS, COBYLA, SLSQP) from the scipy.minimize library. BFGS is a gradient descent method. Note that SLSQP sometimes does not converge to a good score. C: Barchart of runtimes for the different methods. BFGS is always slower. Although SLSQP is typically the fastest method, it sometimes produces poor scores as seen in panel B.

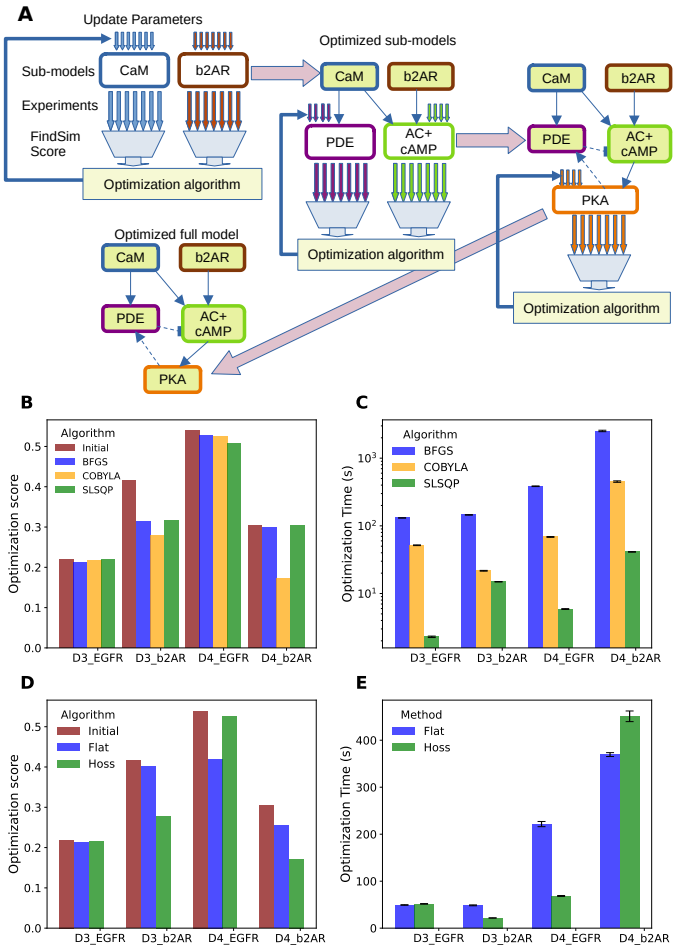

Figure 8: Hierarchical Optimization on large models. A: Schematic of hierarchical optimization as implemented in HOSS. First, the upper level of the model hierarchy is optimized, in this case the CaM and b2AR sub-models. Each is individually optimized, and any of the standard algorithms such as BFGS or COBYLA may be employed. Experiments specific to each sub-model are used to compute individual scores and independently update the sub-model parameters. Then, these sub-models are held fixed and the next level of the hierarchy is optimized (PDE and AC+cAMP sub-models). Finally, the lowest level of hierarchy (PKA) is optimized. With this the entire optimization is complete. B: Barchart of scores for the four different models, comparing the initial score with the final score obtained using three different algorithms from the scipy.minimize library. Note that SLSQP sometimes does not converge to a good score. C: Barchart of runtimes for the different algorithms. As in the flat method, SLSQP is the fastest. D: Hierarchical optimization vs flat scores using COBYLA. With a single exception, Hoss gives better scores. This exception is likely due to relaxation of hierarchy assumptions due to feedback. E: Hierarchical optimization vs flat timing using COBYLA. Except for the D4-b2AR model, Hoss is as fast or faster.

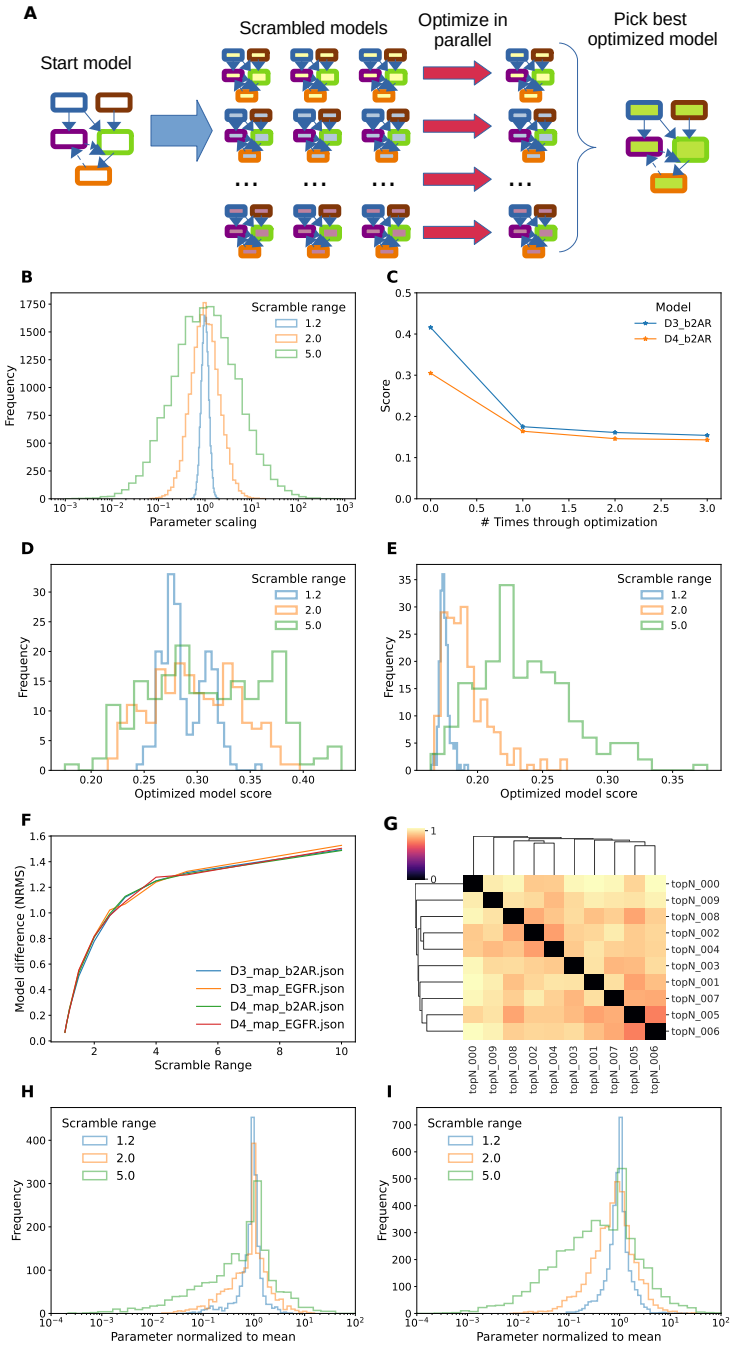

Figure 9: InitScram method. A. Schematic of method. B. log-normal distribution of parameter scaling from reference values for scrambleRange (SR) of 1.2, 2.0, and 5.0. C. Improvement of fit over successive optimizations. A second optimization produces a small improvement, and little improvement results from a third round. D. Score distributions for three values of SR, D3 b2AR model. Note that the peaks are similar but the widths greater for larger SR, hence there are more good scores (left tail of distribution) for large SR. E. Score distributions for 3 values of SR, D4 b2AR model. Here the peaks of the score distribution moves to the right with higher SR, thus the best scores are similar. F. Mapping between parameter scrambling range and NRMS metric for similarity of models. This is independent of model. G. Model similarity score cluster-map for top 10 optimized D3 b2AR models. H: Distribution of parameter scaling for optimized D3-b2AR models, normalized to mean of respective parameter for the best 10 models from that run. The optimized parameters converge very closely to the best 10 means for SR = 1.2 and 2.0. I: Distribution of parameter scaling for optimized D4-b2AR models. Here the tails of the distributions are somewhat wider, but there is still a narrow peak around 1.0 for SR = 1.2 and 2.0. The SR of 5.0 does not

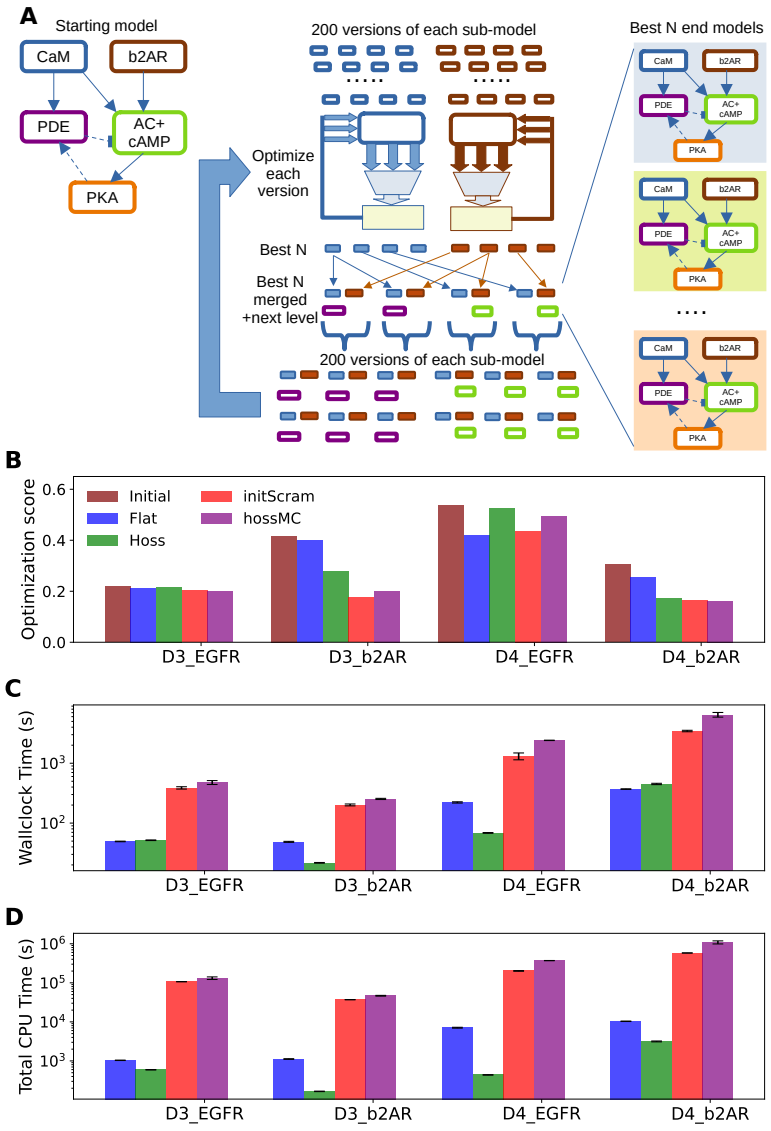

Figure 10: hossMC method A. Schematic of method. The model subsets in the first hierarchical level of the model are each scrambled 200 times, and each such starting point is optimized. The best N models (N=5) are taken from each sub-model and recombined to obtain the overall best N models for the first level. These are then merged with a sub-model from the next level, and these N models are then used as starting points for another round of model scrambling. The 200 scrambled models are again individually optimized as before, and the cycle repeats till we have optimized all levels. The best N merged models are provided as solutions. B, C, D: Comparing the 4 methods (flat, hoss, initScram and hossMC). B: Scores, including the initial score for reference. The InitScram and hossMC methods worked the best, except for the D4-EGFR model. C. Wallclock time. The plain hoss method was usually fastest, or nearly as fast as the flat method. The two randomized methods initScram and hossMC were run on 24 processes, but still were much slower because they performed 200 repeats of all optimizations. D. Total CPU time. Here we factor in the number of processes and the inner parallelization of experiment score estimation. By this metric, the hoss method is substantially better than any other, and the two randomized methods are much more computationally costly.

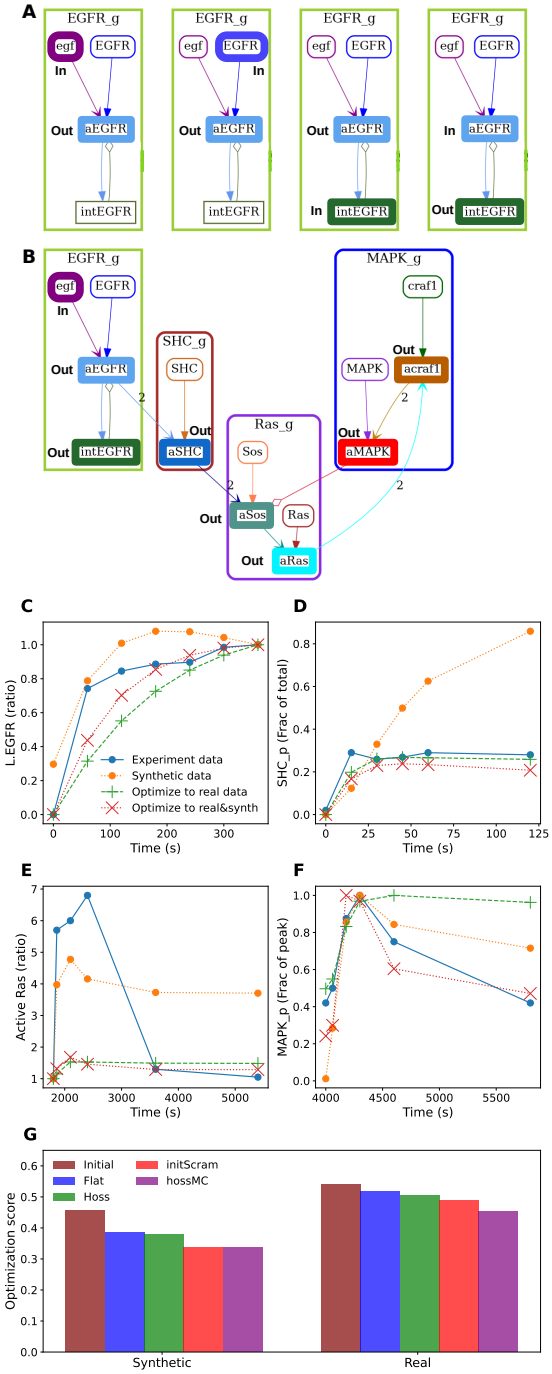

Figure 11: Optimization across hierarchy of detail using synthetic experiments. A: Four input-output relationships are converted to synthetic experiments for the receptor-ligand block of the EGFR pathway, to model each of the potential input-output relationships. B: Seven input-output relationships are used to describe ligand activation of the pathway, with readouts at each step. C-F: Sample input-output mappings of EGF activation of the pathway, to show that adding synthetic data improves model fit. In each case we plot an example of experimental data, of synthetic data from the HillTau EGFR model, of the model optimized only to real data, and of the model optimized to a combined set of synthetic and real data. Note that for some readouts from panel B there were no experimental datasets, and conversely for other readouts there were multiple datasets which typically differed from each other. Note also that the synthetic experiments included both time-series (shown here) and dose-response curves (not shown). C: Activation of EGFR by ligand binding. D: Activation of SHC. E: Activation of Ras. F: Activation of MAPK, the final stage of the pathway. G: Optimization scores for models obtained using different methods using Synthetic+Real data. Left bars score the models against the Synthetic+Real dataset upon which they were optimized, and right bars score the models against only the Real data.

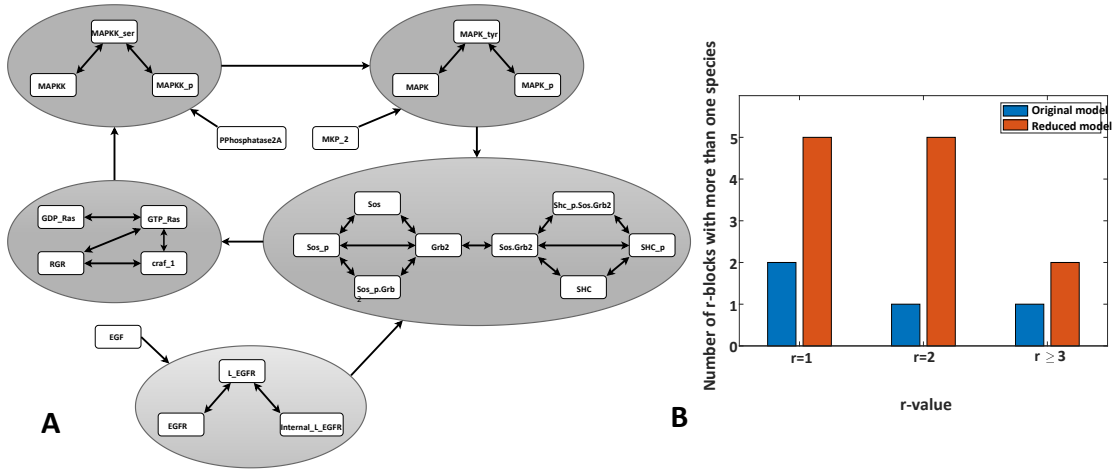

Figure 12: Automatic hierarchical decomposition. A: r-blocks for the EGFR model, for $r \leq 2$, after Michaelis-Menten type reduction. The links between blocks define the quotient graph. B: the number of r-blocks for different values of r. Without Michaelis-Menten reduction, there is only one block for $r > 1$. The reduction increases the number of blocks.

Supplementary Material

Algorithm 1 Create_Hierarchies

Input: *model*
Output: *listIJ* = list of autonomous pairs (I,J)

classified_species = $\emptyset$
listIJ = $\emptyset$
reactionsBySpecies = getReactionsBySpecies(*model*)

for *species* $\in$ *model's list of species* **do**
    `// Initialize_Hierarchy returns (subset of species` I`, subset of reactions` J`)`
    (I, J) = **Initialize_Hierarchy**(species, model, *reactionsBySpecies*)
    if *sorted(I)* $\notin$ *classified_species* **then**
        `// (we use it to avoid redundancy in the list of autonomous pairs)`
        append (I, J) to listIJ
        append sorted(I) to classified_species
    end
end
return *listIJ*

Algorithm 2 getReactionsBySpecies

Input: model
Output: a dictionary *reactionsBySpecies* with all the species as keys, and list of reactions where the key appears as the values

reactionsBySpecies = empty dictionary
for *reaction in model's list of reactions* **do**
    for *reactant in reaction's list of reactants* **do**
        append reaction to reactionsBySpecies[key = reactant]
    end
    for *product in reaction's list of products* **do**
        append reaction to reactionsBySpecies[key = product]
    end
end
return *reactionsBySpecies*

Algorithm 3 Initialize_Hierarchy

Input: *species*, *model*, *reactionsBySpecies*
Output: (I, J) = autonomous pairs (subset of species I, subset of reactions J)

I = [*species*]
J = empty list
add_subset_reactions($species, I, J, model, reactionsBySpecies$)

Algorithm 4 add_subset_reactions

Input: list of species *lsp*, subset of species I, subset of reactions J, *model*, dictionary *reactionsBySpecies*

Output: Void (only update the subset I and the subset J)

$lr = \emptyset$
for *specie in lsp* **do**
  for *reaction in reactionsBySpecies[species]* **do**
    if *reaction* ***neither*** *in J* ***nor*** *in lr* **then**
      append reaction to J
      append reaction to lr
    end
  end
end
if $lr \neq \emptyset$ **then**
  add_subset_species(lr, I, J, model, *reactionsBySpecies*)
end

Algorithm 5 add_subset_species

Input: list of reactions lr, subset of species I, subset of reactions J, *model*, dictionary *reactionsBySpecies*

Output: Void (only update the subset I and the subset J)

$lsp = \emptyset$
for *reaction in lr* **do**
  for *element in elements of reaction's kinetic law* **do**
    if *element is a species in model* ***and*** *element is not in I* **then**
      append element to I
      append element to lsp
    end
  end
end
if $lsp \neq \emptyset$ **then**
  add_subset_reactions(lsp, I, J, model, *reactionsBySpecies*)
end

Algorithm 6 findMMschemes

Input: model
Output: list of schemes

```
reactionsBySpecies = getReactionsBySpecies(model)  // (cf alg2)
listScheme = ∅

for reactionA in model's list of reactions do
    if reactionA is reversible then
        if reactionA has 2 reactants and 1 product then
            scheme.complexSE = reactionA's product
            reactionB = reactionsBySpecies[scheme.complexSE] - reactionA

            if length(reactionB) = 1 and reactionB isn't reversible and reactionB has 1 reactant and
              2 products then
                SandE = reactionA's reactants  // # substrat and enzyme
                PandE = reactionB's products  // # product and enzyme
                enzyme = SandE ∩ PandE

                if len(enzyme) = 1 then
                    scheme.substrate = SandE - enzyme
                    scheme.product =PandE - enzyme
                    scheme.enzyme = enzyme
                    scheme.reactionA = reactionA
                    scheme.reactionB = reactionB

                    append scheme to listScheme
                end
            end
        end
    end
end
return listScheme
```