Article

SMGen: A generator of synthetic models of biochemical reaction networks

Simone G. Riva [1,2,3*], **Paolo Cazzaniga** [4,5,6], **Marco S. Nobile** [7,5], **Simone Spolaor** [8], **Leonardo Rundo** [9,10], **Daniela Besozzi** [8,5,6] and **Andrea Tangherloni** [4*]

1 Department of Haematology, University of Cambridge, Cambridge CB2 0AW, United Kingdom
2 Wellcome Trust Sanger Institute, Wellcome Trust Genome Campus, CB10 1HH Hinxton, United Kingdom
3 Wellcome Trust – Medical Research Council Cambridge, Stem Cell Institute, CB2 0AW Cambridge, United Kingdom
4 Department of Human and Social Sciences, University of Bergamo, 24129 Bergamo, Italy
5 Bicocca Bioinformatics, Biostatistics and Bioimaging Centre (B4), 20854, Vedano al Lambro, Italy
6 SYSBIO/ISBE.IT Centre for Systems Biology, 20126 Milan, Italy
7 Department of Industrial Engineering & Innovation Sciences, Eindhoven University of Technology, The Netherlands
8 Department of Informatics, Systems and Communication, University of Milano-Bicocca, 20126 Milan, Italy
9 Department of Radiology, University of Cambridge, CB2 0QQ Cambridge, United Kingdom
10 Cancer Research UK Cambridge Centre, University of Cambridge, CB2 0RE Cambridge, United Kingdom
* Correspondence: sgr34@cam.ac.uk (S.G.R.); andrea.tangherloni@unibg.it (A.T.)

Abstract: Several software tools for the simulation and analysis of biochemical reaction networks have been developed in the last decades; however, assessing and comparing their computational performance in executing the typical tasks of Computational Systems Biology can be limited by the lack of a standardized benchmarking approach. To overcome these limitations, we propose here a novel tool, named SMGen, designed to automatically generate synthetic models of biochemical reaction networks that, by construction, are characterized by both features (e.g. system connectivity, reaction discreteness) and emergent dynamics resembling real biological networks. The generation of synthetic models in SMGen is based on the definition of an undirected graph consisting in a single connected component, which generally results in a computationally demanding task. To avoid any burden in the execution time, SMGen exploits a Main-Worker paradigm to speed up the overall process. SMGen is also provided with a user-friendly Graphical User Interface that allows the user to easily set up all the parameters required to generate a set of synthetic models with any used-defined number of reactions and species. We analysed the computational performance of SMGen by generating batches of symmetric and asymmetric RBMs of increasing size, showing how a different number of reactions and/or species affects the generation time. Our results show that when the number of reactions is higher than the number of species, SMGen has to identify and correct high numbers of errors during the creation process of the RBMs, a circumstance that increases the overall running time. Though, SMGen can create synthetic models with 512 species and reactions in less than 7 seconds. The open-source code of SMGen is available on GitLab: https://gitlab.com/sgr34/smgen.

Keywords: Synthetic Models; Reaction-based Models; Biochemical Networks; Systems Biology

Citation: Simone G. Riva, Paolo Cazzaniga, Marco S. Nobile, Simone Spolaor, Leonardo Rundo, Daniela Besozzi and Andrea Tangherloni SMGen: A generator of synthetic models of biochemical reaction networks. *Symmetry* **2021**, *1*, 0. https://doi.org/

Received:
Accepted:
Published:

1. Introduction

Systems Biology is a multidisciplinary research field that combines mathematical, computational, and experimental expertise to understand and predict the behavior of complex biological systems [1,2]. Among the different formalisms that can be used to describe intracellular processes, Reaction-Based Models (RBMs) [3–6] are the most suitable for obtaining a detailed comprehension of the mechanisms that control the emergent behavior of the system under analysis [5]. The analysis of RBMs can be used to drive the design of focused lab experiments; to this aim, computational tasks such as parameter estimation, sensitivity analysis, and parameter sweep analysis are generally applied [1,6–8]. Unfortunately, these computational tasks require the execution of huge amounts of simulations, so that the capabilities of biochemical simulators running on Central Processing Units (CPUs) (see, e.g., [9–11]) can be easily overtaken. Thus, several simulators exploiting Graphics Processing Units (GPUs) have been lately introduced to reduce the running times (see, e.g., [12–20]).

A crucial point, whenever new simulators are designed and implemented, regards the evaluation of their computational performance and their efficiency in executing the aforementioned demanding tasks. In this context, RBMs represent a key means as they can be exploited to run both stochastic simulation algorithms and (deterministic) numerical integration methods. Though, only a limited number of RBMs is present in the literature (e.g., signal transduction pathways [21–24] or metabolic pathways [25]). The lack of detailed RBMs, especially those characterized by hundreds or thousands of reactions and molecular species, thus hampers the possibility of performing a thorough analysis of the performance of these simulators.

The computational performance of several GPU-powered tools were assessed using randomly generated synthetic RBMs [14,19,20]. However, only a few generators of biochemical models have been proposed so far, hindering the possibility of having a common and well-defined benchmarking approach. For instance, Komarov *et al.* [14,15] developed a tool to generate synthetic networks, which was then used to test the performance of their GPU-based simulators. Given the number of reactants, the type of reactions to be included in the RBM, and the total number of reactions, they generated synthetic RBM by exploiting a hash table to avoid duplicates. The tool was then modified by randomly sampling the values of the initial concentrations of the species from a uniform distribution and the kinetic constants from a logarithmic distribution [19]. Another known and established model generator is the Reaction Mechanism Generator (RMG) [26], which was specifically developed to create synthetic chemical processes. RMG exploits an extensible set of 45 reaction families to generate elementary reactions from chemical species, while the reaction rates are estimated using a database of known rate rules and reaction templates. RMG relies on graphs to represent the chemical structures, and trees to represent thermodynamic and kinetic data. Due to its peculiarities, RMG was used to, e.g., automatically create kinetic models for the conversion of bio-oil to syngas through gasification [27]. Finally, other tools, such as Moleculizer [28], were introduced for the generation of reaction systems to obtain a deeper understanding of transduction networks.

Despite the efforts done to automatically define synthetic models, all these generators share a common drawback, that is, they have a limited flexibility and can generate only a restricted set of biochemical networks and processes. Considering the impelling necessity of defining a common benchmarking approach that allows for fairly evaluating and comparing different simulation approaches [29], we propose here a novel tool, named SMGen, designed to automatically generate synthetic yet realistic biological networks codified as RBMs, whose dynamics resemble those of real biological networks. SMGen adheres to well-defined structural characteristics based on graph theory and linear algebra properties, in particular, it exploits the definition of an undirected graph with a single connected component, which makes the whole generation process a computationally demanding task. To overcome this limitation, on the one hand, SMGen internally codifies all data structures by means of sparse matrices as well as *ad-hoc* structures specifically designed to avoid worthless values, which would increase the running time required to generate RBMs. On the other hand, SMGen is able to drastically reduce the computational time by exploiting a Main-Worker paradigm used to distribute the overall generation process of RBMs onto multi-core CPUs. We show that SMGen can create, in less than 7 seconds, synthetic RBMs with hundreds of chemical species and molecular reactions that resemble the behavior of real biochemical networks.

Among the different features provided by SMGen, it allows for easily generating both symmetric and asymmetric RBMs: symmetric RBMs are composed of a number of species equal to the number of reactions, while in asymmetric RBMs the number of species can be lower than the number of reactions or vice-versa. From a computational point of view, the concept of symmetry is crucial in the analysis of complex networks to measure their information and entropy [30]. Studying the symmetries of mechanistic models, which aim at formalizing the structures and behavior of the underlying dynamics of biological systems, can allow for revealing the intrinsic properties of the system of interest [31]. Moreover, the possibility of evaluating GPU-powered simulators using symmetric and asymmetric RBMs is fundamental to understand their performance under different conditions. Indeed, a fair comparison would allow the user to select the best simulator based on characteristics of the RBM that has to be analysed.

SMGen allows also for exporting the generated RBMs into the Systems Biology Markup Language (SBML) [32], Version 4 Level 2, and into the BioSimWare standard [33], which is used by different GPU-powered simulators. Thus, we designed and developed SMGen to be a unifying, user-friendly, and standalone tool freely accessible to the Systems Biology community. The RBMs can be easily generated by using the provided user-friendly Graphical User Interface (GUI), which is designed to help the users in setting all the parameters required to generate the desired RBMs.

The manuscript is structured as follows. Section 2 describes the mathematical formalism of the RBMs, as well as the structural characteristics that must be complied to generate realistic biological networks. In addition, we provide all the algorithms and details at the basis of SMGen. Section 3 shows the experimental results achieved by SMGen. Finally, a discussion and conclusive remarks are provided in Section 4.

2. Materials and Methods

2.1. Reaction-Based Models

An RBM is defined by specifying the set $\mathcal{S} = \{S_1, \ldots, S_N\}$ of N molecular species, and the set $\mathcal{R} = \{R_1, \ldots, R_M\}$ of M biochemical reactions that describe the interactions among the species appearing in $\mathcal{S}$. Each reaction R_i, with $i = 1, \ldots, M$, is defined as:

$$R_i : \sum_{j=1}^{N} a_{ij} S_j \xrightarrow{k_i} \sum_{j=1}^{N} b_{ij} S_j, \tag{1}$$

where a_{ij} and $b_{ij} \in \mathbb{N}$ are the stoichiometric coefficients, and $k_i \in \mathbb{R}^+$ is the kinetic constant associated with R_i. The stoichiometric coefficients specify how many molecules of species S_j, with $j = 1, \ldots, N$, appear either as reactants or products in reaction R_i. Note that some species might not appear in a reaction, so that the corresponding stoichiometric coefficient will be equal to 0. The order of a reaction is equal to the total number of molecules (of the same or different species) that appear as reactants in that reaction.

Each RBM can be written in the compact matrix-vector form $\mathbf{AS} \xrightarrow{\mathbf{K}} \mathbf{BS}$, where $\mathbf{S} = [S_1 \cdots S_N]^\top$ is the N-dimensional column vector of the molecular species, $\mathbf{K} = [k_1 \cdots k_M]^\top$ is the M-dimensional column vector of the kinetic constants, while $\mathbf{A}, \mathbf{B} \in \mathbb{N}^{M \times N}$ are the stoichiometric matrices, whose non-negative elements $[A]_{i,j}$ and $[B]_{i,j}$ correspond to the stoichiometric coefficients a_{ij} and b_{ij} of the reactants and products of the reactions, respectively.

Starting from an RBM and assuming the law of mass-action [34–36], the system of coupled ODEs corresponding to the RBM can be derived as follows:

$$\frac{d\mathbf{X}}{dt} = (\mathbf{B} - \mathbf{A})^T [\mathbf{K} \circ \mathbf{X}^{\mathbf{A}}], \tag{2}$$

where each ODE describes the variation in time of a species' concentration. In Equation 2 the N-dimensional vector $\mathbf{X} = [X_1 \cdots X_N]$ represents the concentration values of species

$S_1, \ldots, S_N$, while $\mathbf{X}^{\mathbf{A}}$ is the vector-matrix exponentiation form [34]; the symbol $\circ$ denotes the entry-by-entry matrix multiplication (Hadamard product).

2.2. SMGen

In order to generate synthetic and yet realistic models of biochemical networks, SMGen complies with specific structural characteristics that the RBMs have to satisfy, that is:

- *System connectivity*: a biochemical network can be represented as an undirected graph with a single connected component, where the nodes represent the molecular species and the edges correspond to the species interactions (i.e., reactions). In order to satisfy this constraint, each species $S_j \in \mathcal{S}$, with $j = 1, \ldots, N$, must be involved in at least one reaction $R_i \in \mathcal{R}$, with $i = 1, \ldots, M$.
- *Maximum number of reactants and products*: for each reaction $R_i \in \mathcal{R}$, with $i = 1, \ldots, M$, the number of reactants and the number of products cannot be arbitrarily large, but has to be lower than or equal to a user-defined values. Stated otherwise, the maximum order of the generated reactions should be fixed, and mass balance constraints should be implicitly considered.
- *Linear independence*: to ensure that each reaction R_i, with $i = 1, \ldots, M$, resembles a plausible biochemical reaction, the vectors of the stoichiometric coefficients of the reactants and products involved in R_i must be linearly independent.
- *Reaction discreteness*: each reaction R_i, with $i = 1, \ldots, M$, must appear only once in the network, that is, duplicated reactions are not allowed.

SMGen is provided with a user-friendly GUI (see Figure 1) that allows the user to easily set up all the parameters required to generate the desired synthetic RBMs:

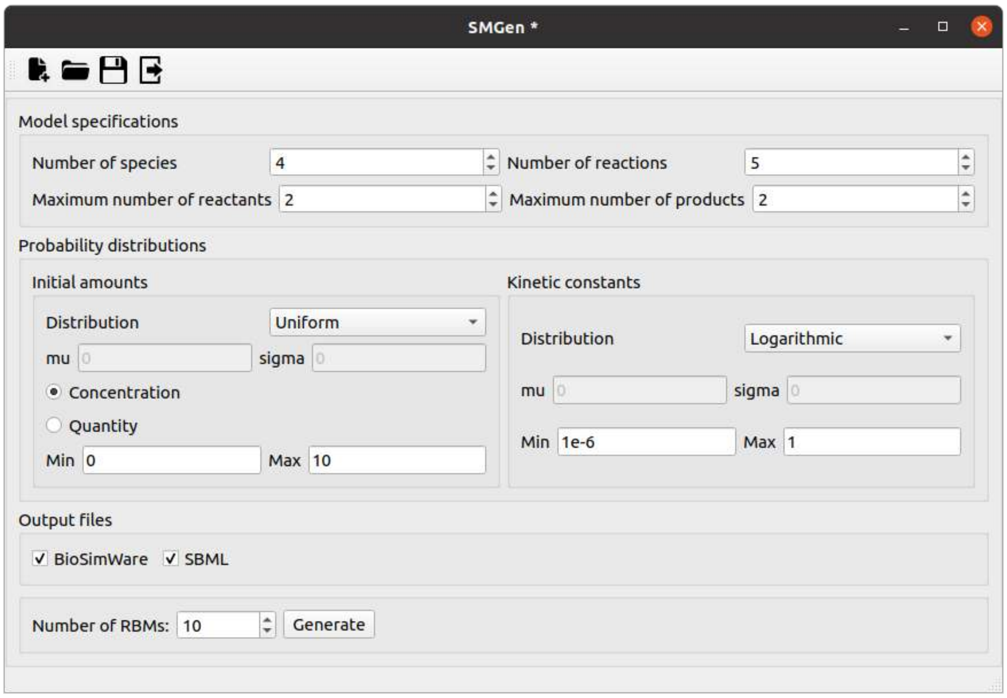
- the number of species N and the number of reactions M;
- the maximum number of reactants and products max_{num_r} and max_{num_p} that might appear in any reaction;
- the probability distribution $\mathcal{D}_s$ that is used to initialize the species amounts (to be chosen among uniform, normal, logarithmic or log-normal distributions);
- the minimum and maximum values min_s and max_s for the initial species amounts (to be specified either as number of molecules or concentrations);
- the probability distribution $\mathcal{D}_r$ that is used to set the values of the kinetic constants (to be chosen among uniform, normal, logarithmic or log-normal distributions);
- the minimum and maximum values min_r and max_r for the kinetic constants;
- the total number of RBMs that the user wants to generate;
- the output format file to export the generated RBMs (i.e., BioSimWare [33] and SBML [32]);
- the mean and standard deviation values μ_s and σ_s for the initial amounts—as well as the mean and standard deviation values μ_r and σ_r for the kinetic constants—must also be provided if the normal or log-normal distributions are selected.

Figure 2 shows a high-level scheme of the proposed implementation of SMGen, which exploits the Main-Worker paradigm to speed up the generation of the RBMs [37]. The user can specify the number of processes P—otherwise automatically set to the minimum value 3—which are used as follows:

- Proc_1 manages the GUI;
- Proc_2 is the Main process that orchestrates the computation;
- Proc_p, with $p = 3, \ldots, P$, are the Worker processes.

The whole functioning of SMGen can be summarized as follows:

- the user interacts with the GUI, managed by Proc_1, to fill in all the required values for the parameters necessary to create the RBMs;
- Proc_1 sends the values of all parameters to the Main process (Proc_2), which allocates the resources and distributes the work to the Workers (Proc_p, with $p = 3, \ldots, P$);



SMGen *

Model specifications

Number of species: 4
Number of reactions: 5
Maximum number of reactants: 2
Maximum number of products: 2

Probability distributions

Initial amounts
Distribution: Uniform
mu: 0 sigma: 0
(●) Concentration
( ) Quantity
Min: 0 Max: 10

Kinetic constants
Distribution: Logarithmic
mu: 0 sigma: 0
Min: 1e-6 Max: 1

Output files
☑ BioSimWare ☑ SBML

Number of RBMs: 10 Generate

Figure 1. Graphical User Interface of SMGen. The user can set all the parameters to generate the desired RBMs, i.e., number of species and reactions, maximum number of reactants and products, probability distribution for the initial amounts and kinetic constants, and the output format file (i.e., BioSimWare, SBML).

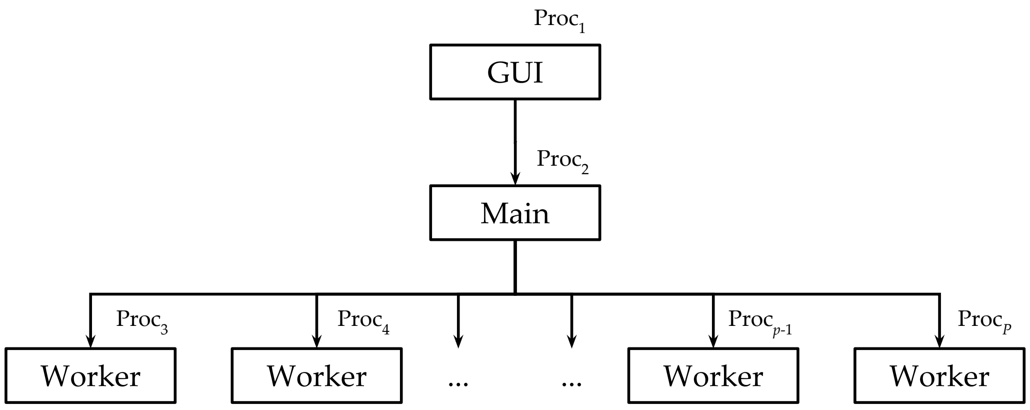


Figure 2. Scheme of the Main-Worker implementation of SMGen. The Main process (Proc$_2$) orchestrates all the available Workers (Proc$_p$, with $p = 3, \ldots, P$), which generate the RBMs in a distributed computing fashion.

- each Worker (Proc$_p$, with $p = 3, \ldots, P$) generates a RBM. As soon as a Worker terminates its execution, it communicates to the Main process that the RBM has been created. If necessary, the Main process assigns the generation of other RBMs to idle Workers. When all required RBMs are obtained, the Workers enter in the death state, while the Main process waits for further instructions from Proc$_1$.

The workflow of each Worker consists in 9 different phases, in which a specific algorithm is executed (see Figure 3).

The pseudo-code reported in Algorithm 1 briefly summarizes all the steps required to generate a single RBM; the pseudo-code of the procedures invoked within Algorithm 1 are reported in Appendix A. For the sake of clarity, Table 1 lists the symbols used in the following description and in the pseudo-codes.

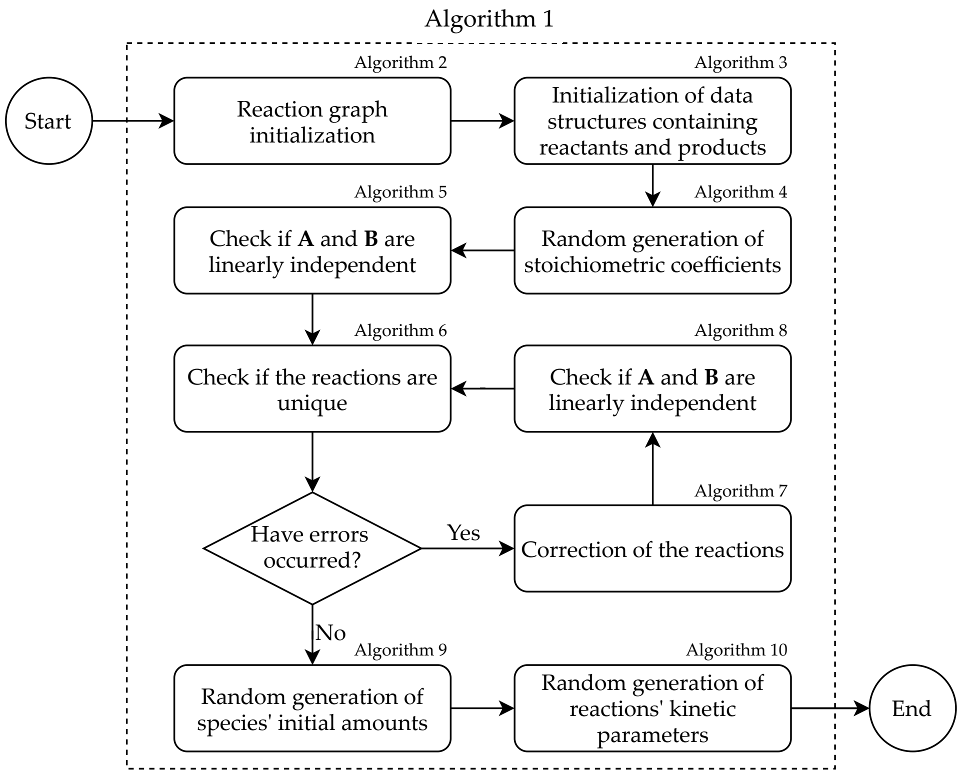

Algorithm 1
Start
Algorithm 2
Reaction graph initialization
Algorithm 3
Initialization of data structures containing reactants and products
Algorithm 4
Random generation of stoichiometric coefficients
Algorithm 5
Check if A and B are linearly independent
Algorithm 6
Check if the reactions are unique
Algorithm 8
Check if A and B are linearly independent
Have errors occurred?
Yes
Algorithm 7
Correction of the reactions
No
Algorithm 9
Random generation of species' initial amounts
Algorithm 10
Random generation of reactions' kinetic parameters
End

Figure 3. Workflow of a single Worker execution. First, the graph of the reactions is randomly initialized, and then converted into the data structures used to store the reactants and products. Second, the stoichiometric coefficients are randomly generated and the consistency of the reactants and products is verified. Third, the initial amounts of the species and the kinetic parameters of the reactions are randomly generated using the probability distributions specified by the user.

The steps performed by each Worker to generate a RBM are the following:

1. Given the parameters provided by the user, the graph representing the species and their interactions is randomly initialized (line 3 of Algorithm 1, see Algorithm 2).
2. The adjacency matrix of the graph generated in Step 1 is converted into the stoichiometric matrices **A** and **B** (line 5 of Algorithm 1, see Algorithm 3). Note that the instructions in lines 6–17 of Algorithm 1 are required to build the data structure of the initial graph, which is then modified).
3. The stoichiometric coefficients are randomly generated (line 19 of Algorithm 1, see Algorithm 4).
4. For each reaction R_i, with $i = 1, \ldots, M$, the linear independence between the reactants and products is verified (line 21 of Algorithm 1, see Algorithm 5).
5. The uniqueness of each reaction in the RBM is verified (line 23 of Algorithm 1, see Algorithm 6).
6. Any error in the RBM identified in the previous steps is corrected (line 27 of Algorithm 1, see Algorithm 7); the linear independence and the uniqueness of the reactions in the modified RBM are iteratively verified (lines 29 and 31 of Algorithm 1, see Algorithms 8 and 6, respectively).
7. The initial amounts of the species are generated according to the chosen probability distribution (line 33 of Algorithm 1, see Algorithm 9). If a species appears only as a reactant in the whole RBM, its amount is set to remain unaltered. The rationale behind this is double: on the one hand, we avoid the possibility of creating reactions that could be applied at most once, which is a highly improbable situation in biological systems; on the other hand, we mimic the non-limiting availability of some biochemical resources, for instance, it might be used to reproduce the

Algorithm 1 SMGen: workflow of a single Worker execution.

```
1:  function GENERATOR($M, N, max_{num_p}, max_{num_r}, \mathcal{D}_s, \mathcal{D}_r, min_s, max_s, min_r, max_r, \mu_s, \sigma_s, \mu_r, \sigma_r$)
2:      ## Algorithm 2
3:      $\mathbf{G} \leftarrow$ GRAPH_GEN($N$)
4:      ## Algorithm 3
5:      $\mathbf{A}, \mathbf{B} \leftarrow$ STOICH_MATRICES_GEN($M, N, \mathbf{G}$)
6:      $AIJ[\cdot], BIJ[\cdot] \leftarrow [\,]$                                   ▷
7:      for $i = 1$ to $M$ do                                                  ▷
8:          for $j = 1$ to $N$ do                                              ▷
9:              if $\mathbf{A}[i, j] == 1$ then                                ▷
10:                 $AIJ \leftarrow AIJ \odot \langle i, j \rangle$            ▷
11:             if $\mathbf{B}[i, j] == 1$ then                                ▷
12:                 $BIJ \leftarrow BIJ \odot \langle i, j \rangle$            ▷
13:     ## Algorithm 4
14:     $\mathbf{A}, \mathbf{B} \leftarrow$ STOICH_COEFFICIENTS_GEN($\mathbf{A}, \mathbf{B}, M, N, max_{num_r}, max_{num_p}, AIJ, BIJ$)
15:     ## Algorithm 5
16:     $err_{LinDep} \leftarrow$LINEAR_INDEPENDENCE1($\mathbf{A}, \mathbf{B}, M$)
17:     ## Algorithm 6
18:     $err_{Repeat} \leftarrow$UNIQUE_REACTIONS($\mathbf{A}, \mathbf{B}, M$)
19:     while $err_{LinDep} \wedge err_{Repeat}$ are not empty do
20:         $rows_{Err} \leftarrow unique(err_{LinDep} \odot err_{Repeat})$
21:         ## Algorithm 7
22:         $\mathbf{A}, \mathbf{B} \leftarrow$ CORRECTION_REACTIONS($\mathbf{A}, \mathbf{B}, rows_{Err}, AIJ, BIJ, max_{num_r}, max_{num_p}$)
23:         ## Algorithm 8
24:         $err_{LinDep} \leftarrow$ LINEAR_INDEPENDENCE2($\mathbf{A}, \mathbf{B}, rows_{Err}$)
25:         ## Algorithm 6
26:         $err_{Repeat} \leftarrow$UNIQUE_REACTIONS($\mathbf{A}, \mathbf{B}, M$)
27:     ## Algorithm 9
28:     $\mathbf{M_0} \leftarrow$ AMOUNTS_GEN($N, \mathcal{D}_s, min_s, max_s, \mu_s, \sigma_s$)
29:     ## Algorithm 10
30:     $\mathbf{K} \leftarrow$ KINETIC_CONSTANTS_GEN($M, \mathcal{D}_r, min_r, max_r, \mu_r, \sigma_r$)
```

▷ → the instructions shown in lines 6–12 are required to build the structure of the initial graph of the reactions.

execution of *in vitro* experiments where some species are continually introduced in the systems to keep their amount constant [38].

8. The kinetic constants of the reactions are generated according to the chosen probability distribution (line 35 of Algorithm 1, see Algorithm 10).

SMGen was developed using the Python programming language and exploiting `mpi4py` [39], which provides bindings of the Message Passing Interface (MPI) specifications for Python to leverage multi-core CPUs [40]. The open-source code of SMGen is available on GitLab (https://gitlab.com/sgr34/smgen) under the GPL-3 license.

3. Results

We analysed the performance of SMGen regarding both its capability of creating RBMs resembling the dynamics of real biochemical networks, and the computational time required to generate sets of RBMs of increasing size. All tests were executed on a workstation equipped with an Intel Core i7-8750H CPU (clock 4.1 GHz), 16 GB of RAM and a Samsung 970 EVO solid-state drive NVMe PCIe (up to 3400 MB/s and 1500 MB/s read and write speed, respectively), running Ubuntu 20.04 LTS.

As a first batch of tests, we generated 100 synthetic RBMs characterized by a limited number of reactions and species (4 and 5, respectively), and we analysed their characteristics and dynamics. We set to 3 both the maximum number of reactants

Symbol	Description
M	Number of reactions composing the RBM
N	Number of species involved in the RBM
max_{num_r}	Maximum number of the reactants
max_{num_p}	Maximum number of the products
$\mathbf{M_0}$	Array of the initial amounts
$\mathbf{K}$	Array of the kinetic constants
$\mathbf{A}$	Stoichiometric matrix of the reagents
$\mathbf{B}$	Stoichiometric matrix of the products
$\mathbf{G}$	Adjacency matrix of the graph of the reactions
$\mathcal{D}_s$	Probability distribution for the initial amounts
min_s	Minimum value of the initial amounts
max_s	Maximum value of the initial amounts
μ_s	Mean of the normal and log-normal distributions for the initial amounts
σ_s	Standard deviation of the normal and log-normal distributions for the initial amounts
$\mathcal{D}_r$	Probability distribution for the kinetic constants
min_r	Minimum value of the kinetic constants
max_r	Maximum value of the kinetic constants
μ_r	Mean of the normal and log-normal distributions for the initial amounts
σ_r	Standard deviation of the normal and log-normal distributions for the kinetic constants amounts
$\odot$	The concatenation operator

Table 1: List of symbols used in the pseudo-code of algorithms at the basis of SMGen.

max_{num_r} and products max_{num_p}. We sampled the initial amounts of species from a normal distribution with mean $\mu_s = 5$ and standard deviation $\sigma_s = 5$, considering a minimum value $min_s = 0$ and maximum value $max_s = 10$. The kinetic constants were instead sampled from a logarithm distribution with minimum value $min_r = 10^{-16}$ and maximum value $max_r = 10$.

Table 2 shows the list of reactions along with the kinetic constants of one of these 100 synthetic RBMs. Since the species X_0 appears only as a reactant, its amount will be kept constant during the simulation. The initial molecular amounts of all species—given as number of molecules—are listed in Table 3. This small RMB includes the basic “cascade of reactions” structure typically observed in signaling pathways, starting from the source represented by species X_0 and X_4, toward species X_2 and X_3.

We simulated the dynamics of this RBM for 50 time steps (arbitrary unit), and the achieved dynamics are shown in Figure 4. These plots evidence that, although the RBM was randomly generated by SMGen, it produces a realistic behavior. It is worth mentioning that obtaining realistic RBMs exhibiting non trivial dynamics is fundamental to perform in-depth computational analyses and comparisons among the existing and the novel simulators. Indeed, in the case of stable or flat dynamics, or when the overall behavior of the network is extremely fast and instantly exhausts all the reactants, the most advanced integration algorithms are able to simulate the emergent dynamics in just one computation step [20]. In such a case, the computational performance of the simulation tools is only partially assessed, thus hindering a fair comparison among the tools.

As a second batch of tests, we evaluated the computational performance of SMGen exploiting the Main-Worker paradigm running on 4 distinct cores of the CPU. First, we considered the generation of symmetric RBMs with an increasing number of species and reactions (i.e., $M = N = 2^x$, with $x = 2, \dots, 9$). The initial amounts and kinetic

No.	Reagents	Products	Constant
R_1	$X_0 + X_4$	X_3	$4.295 \cdot 10^{-5}$
R_2	X_4	$X_1 + 2X_2$	$2.207 \cdot 10^{-2}$
R_3	X_4	$X_2 + X_4$	$7.070 \cdot 10^{-4}$
R_4	$X_1 + X_4$	$X_2 + X_3$	$4.613 \cdot 10^{-2}$

Table 2: List of the reactions of a RBM with by 4 reactions and 5 species generated by SMGen.

Species	Initial amount
X_0	4
X_1	8
X_2	7
X_3	8
X_4	1

Table 3: Initial molecular amounts of the RBM generated by SMGen shown in Table 2.

constants were randomly sampled from a uniform distribution with minimum values $min_s = min_r = 0$ and maximum values $max_s = max_r = 10$. We also varied the maximum numbers of reactants and products considering the set of values $\{2, 3, 4\}$ and setting $max_{num_r} = max_{num_p}$. For each of the resulting 24 parameters combinations, we created 100 RBMs to collect statistically sound results about the performance of SMGen. As described in Section 2, two kinds of error can occur during the generation of a RBM: a linear dependence between reactants and products, and duplicated reactions. Since the correction of these errors is one of the most time-consuming phases of SMGen, we separately measured the generation time, which indicates the running time spent by SMGen to generate a RBM, and the saving time, which refers to the writing operations on the solid-state drive. Figure 5 shows the average running time required by SMGen to generate and save a RBM. As expected, both the generation and saving time increase along with the number of species and reactions of the RBM. Moreover, we observe that the maximum number of reactants and products have a slight impact on both the generation and the saving time; in most of the cases, increasing these values results in a higher running time.

Finally, we exploited SMGen for the creation of asymmetric RBMs, to evaluate how a different number of species and reactions affects the running time. As in the case of symmetric RBMs, we measured both the generation time and the saving time. The asymmetric RBMs were created as follows:

- we set the number of species $N \in \{4, 8, 16, 32, 64\}$, and then we varied the number of reactions $M \in \{2N, 4N, 8N\}$;
- we set the number of reactions $M \in \{4, 8, 16, 32, 64\}$, and then we varied the number of species $N \in \{2M, 4M, 8M\}$;
- we varied both the maximum numbers of reactants max_{num_r} and products max_{num_p} in $\{2, 3, 4\}$.

In such a way, we obtained a total of 90 different combinations of the parameters (i.e., number of species, number of reactions, and maximum number of reactants and products) to be tested; as in the previous tests, for each combination we generated 100 RBMs to collect statistically sound results. Figure 6 shows the average running time required to create RBMs with dimensions $N \times M$, highlighting once again that both the generation time and the saving time increase along with the size of the RBMs. As in the case of symmetric RBMs, we observed the same effect due to the maximum number of reactants and products allowed in the reactions. As expected, when there are more reactions than species (bottom panel in Figure 6) the generation times are higher than

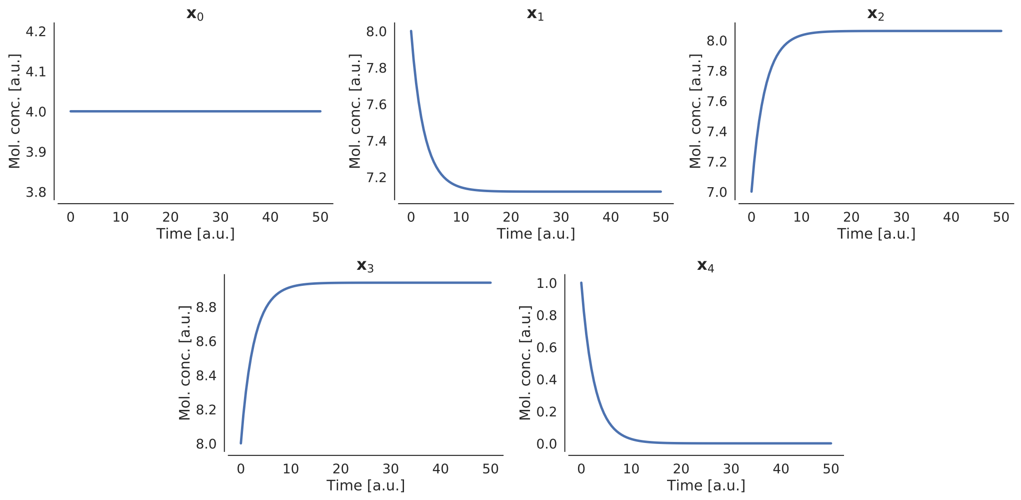


Figure 4. Dynamics of the species of the synthetic RBM generated by SMGen shown in Table 2.

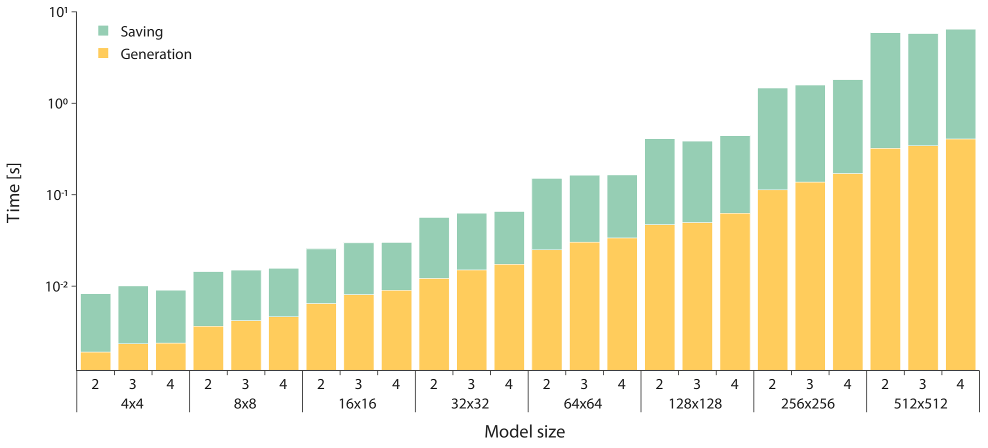
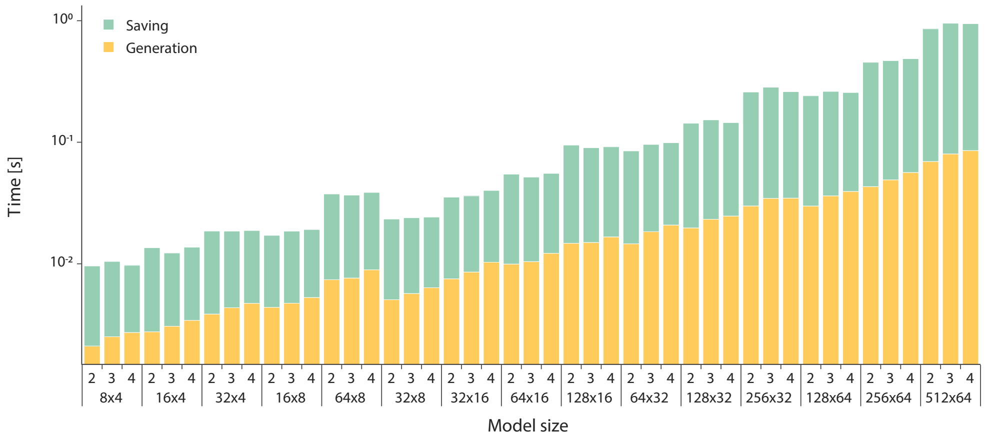


Figure 5. Stacked bar plot showing the average generation time (yellow bars) and the average saving time (green bars) required by SMGen to generate a symmetric RBM. Note that the *y*-axis is in logarithmic scale.

the opposite situation (top panel in Figure 6). This circumstance is due to the potential higher number of errors that SMGen has to identify and correct. Indeed, when $M \gg N$, the probability that repeated reactions are randomly generated is higher than the case when $N \gg M$, because the number of admissible reactions strictly depends on the number of species.

4. Conclusions

In this work we presented SMGen, a generator of synthetic reaction-based models displaying the characteristics of real biochemical networks, which can be exploited to create benchmarks for the evaluation of novel and existing simulators. SMGen is particularly suitable to create the RBMs necessary to assess the performance of GPU-based simulators. As a matter of fact, the performance of GPU-powered simulators can drastically change with the number of chemical species and reactions composing an RBM. Considering that each RBM can be converted into the corresponding system of coupled Ordinary Differential Equations (ODEs), the resolution of this system of ODEs can be performed in a parallel fashion, where each ODE is resolved by a thread. Since each ODE corresponds to a specific chemical species, a higher number of species



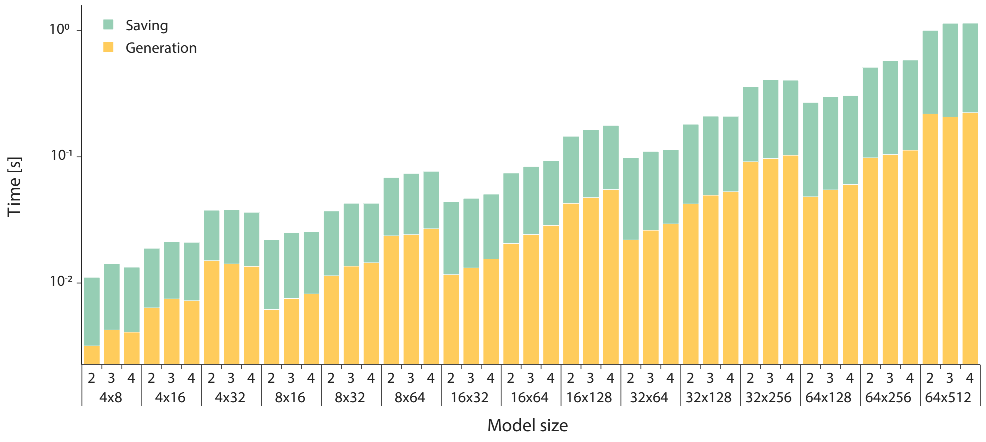


Figure 6. Stacked bar plots showing the average generation time (yellow bars) and the average saving time (green bars) required by SMGen to generate an asymmetric RBM with more species than reactions (top) and with more reactions than species (bottom). Note that the y-axes are in logarithmic scale.

generally lead to a higher parallelization, increasing the computational performance of the simulator. On the contrary, considering that the number of the reactions composing the biological system is roughly related to the length of each ODE, in terms of the mathematical complexity, the higher the number of reactions the higher the number of operations that must be performed by each thread, leading to a higher running time [19,20].

SMGen was developed in Python and was designed to be a unifying, user-friendly, and standalone tool. In addition, SMGen exploits the Main-Worker paradigm to speed up the generation of RBMs; this was implemented using the `mpi4py` [39] library, where the first process manages the GUI, the second one is the Main process, and all the other processes are the Workers that generate the RBMs in a distributed computing fashion. Thanks to the GUI of SMGen, the user can easily set up all the parameters characterizing the required RBMs, e.g., the number of species and reactions, the maximum number of reactants and products per reaction, the probability distributions (uniform, normal, logarithmic, log-normal) to generate the initial amounts of the species and the values of the kinetic constants associated with the reactions, the output file format to save the RBMs.

We assessed the capabilities of SMGen for the creation of RBMs characterized by a non trivial behavior, and we presented an example of a synthetic and yet realistic RBM, together with the simulated dynamics. We also tested the computational performance of SMGen by generating batches of symmetric and asymmetric RBMs of increasing size, showing the impact of the number of reactions and species, and of the number of reactants and products per reaction, on the generation times. We observed that when the number of reactions is higher than the number of species, SMGen generally identifies and corrects high numbers of errors during the creation process of the RBMs, a circumstance that inevitably increases the overall running time.

As a future extension of this work, we plan to develop an Application Programming Interface (API), so that SMGen can be seamlessly integrated into other processing pipelines, tools and simulators. We will also introduce a new feature specifically developed to generate feedback loops in synthetic RBMs, exploiting the theory of Petri nets [41,42]. Feedback loops are fundamental elements of biological processes that lead to the establishment of oscillatory regimes and non-linear dynamics [22]. Finally, we plan to include an initial check of the parameter values set by the user, based on some heuristics, to verify whether the RBMs can be actually generated as requested. This initial step will allow for avoiding worthless calculations and to suggest useful modifications of parameters to the user.

Author Contributions: S.G.R., P.C., M.S.N., and A.T. conceived and designed the tool. S.G.R. developed the tool. S.G.R., P.C., and A.T. conceived and designed the analyses. S.G.R. performed the analyses. P.C., D.B., and A.T. wrote the paper. S.G.R., M.S.N., S.S., and L.R. reviewed the paper. D.B., P.C., M.S.N., and A.T. supervised the whole work. All authors have read and agreed to published this version of the manuscript.

Funding: This research received no external funding.

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

Appendix A. Algorithms

We report here all the algorithms referring to the functions called by Algorithm 1, which represents the workflow of each Worker process.

Algorithm 2 Random initialization of the graph of reactions

```
1:  function GRAPH_GEN($N$)
2:      $\mathbf{G}[\cdot,\cdot], v[\cdot] \leftarrow 0$
3:      for $j = 1$ to $N$ do
4:          $v[j] \leftarrow j$
5:      $ind_1 \leftarrow random(1, len(v))$
6:      $ind_2 \leftarrow random(1, len(v))$
7:      $n_1, n_2 \leftarrow v[ind_1], v[ind_2]$
8:      $v \leftarrow delete(v[ind_1])$
9:      $v \leftarrow delete(v[ind_2])$
10:     $\mathbf{G}[n_1, n_2] \leftarrow 1$
11:     while $v$ is not empty do
12:         if $random \in \{0,1\} == 0$ then
13:             $ind \leftarrow random(1, len(v))$
14:             $k \leftarrow v[ind]$
15:             $v \leftarrow delete(v[ind])$
16:             $\mathbf{G}[n_1, k] \leftarrow 1$
17:             $n_2 \leftarrow k$
18:         else
19:             $ind \leftarrow random(1, len(v))$
20:             $k \leftarrow v[ind]$
21:             $v \leftarrow delete(v[ind])$
22:             $\mathbf{G}[k, n_2] \leftarrow 1$
23:             $n_1 \leftarrow k$
24:     return $\mathbf{G}$
```

Algorithm 3 Conversion of the adjacency matrix **G** into the stoichiometric matrices **A** and **B**

```
1:  function STOICH_MATRICES_GEN($M, N, \mathbf{G}$)
2:      $i \leftarrow 1$
3:      for $n_1 = 1$ to $N$ do
4:          for $n_2 = 1$ to $N$ do
5:              if $\mathbf{G}[n_1, n_2] == 1$ then
6:                  $\mathbf{A}[i, n_1] \leftarrow 1$
7:                  $\mathbf{B}[i, n_2] \leftarrow 1$
8:                  if $i == M$ then
9:                      $i \leftarrow 1$
10:                 else
11:                     $i \leftarrow i + 1$
12:     return $\mathbf{A}, \mathbf{B}$
```

Algorithm 4 Generation of the random stoichiometric coefficients

```
1: function STOICH_COEFFICIENTS_GEN($\mathbf{A}, \mathbf{B}, M, N, max_{num_r}, max_{num_p}, AIJ, BIJ$)
2:     for $i = 1$ to $M$ do
3:         for $k = 0$ to $max_{num_r}$ do
4:             $coef \leftarrow random(0, max_{num_r})$
5:             $j \leftarrow random(1, N)$
6:             if $coef \neq 0$ & $A[i, \cdot] + coef - A[i, j] \leq max_{num_r}$ then
7:                 $\mathbf{A}[i, j] \leftarrow coef$
8:             else if $coef == 0$ & $\langle i, j \rangle \notin AIJ$ then
9:                 $\mathbf{A}[i, j] \leftarrow coef$
10:    for $i = 1$ to $M$ do
11:        for $k = 0$ to $max_{num_p}$ do
12:            $coef \leftarrow random(0, max_{num_p})$
13:            $j \leftarrow random(1, N)$
14:            if $coef \neq 0$ & $B[i, \cdot] + coef - B[i, j] \leq max_{num_p}$ then
15:                $\mathbf{B}[i, j] \leftarrow coef$
16:            else if $coef == 0$ & $\langle i, j \rangle \notin BIJ$ then
17:                $\mathbf{B}[i, j] \leftarrow coef$
18:    return $\mathbf{A}, \mathbf{B}$
```

Algorithm 5 Checking the linear independence between $\mathbf{A}[i, \cdot]$ and $\mathbf{B}[i, \cdot]$

```
1: function LINEAR_INDEPENDENCE1($\mathbf{A}, \mathbf{B}, M$)
2:     $err_{LinDep} \leftarrow [\,]$
3:     for $i = 1$ to $M$ do
4:         if $\mathbf{A}[i, \cdot] \wedge \mathbf{B}[i, \cdot]$ are linearly dependent then
5:             $err_{LinDep} \leftarrow err_{LinDep} \odot i$
6:     return $err_{LinDep}$
```

Algorithm 6 Checking if the generated reactions are unique

```
1: function UNIQUE_REACTIONS($\mathbf{A}, \mathbf{B}, M$)
2:     $\mathbf{AB}, \mathbf{ABs} \leftarrow [\,]$
3:     $err_{Repeat} \leftarrow [\,]$
4:     for $i = 1$ to $M$ do
5:         $\mathbf{AB}[i] \leftarrow \mathbf{A}[i, \cdot] \odot \mathbf{B}[i, \cdot]$
6:     for $i = 1$ to $M$ do
7:         if $\mathbf{AB}[i]$ is in $\mathbf{ABs}$ then
8:             $err_{Repeat} \leftarrow err_{Repeat} \odot i$
9:         else
10:            $\mathbf{ABs} \leftarrow \mathbf{ABs} \odot \mathbf{AB}[i]$
11:    return $err_{Repeat}$
```

Algorithm 7 Random correction of the repeated reactions

```
function CORRECTION_REACTIONS($\mathbf{A}, \mathbf{B}, rows_{Err}, AIJ, BIJ, max_{num_r}, max_{num_p}$)
    for $i = 1$ to $len(rows_{Err})$ do
        $\mathbf{A}[rows_{Err}[i], \cdot] \leftarrow 0$
        $\mathbf{B}[rows_{Err}[i], \cdot] \leftarrow 0$
        if $rows_{Err}[i] \in AIJ[\cdot, 1]$ then
            for $k = 1$ to $len(AIJ)$ do
                if $AIJ[k, 1] == rows_{Err}[i]$ then
                    $\mathbf{A}[AIJ[k, 1], AIJ[k, 2]] \leftarrow 1$
            for $c = 0$ to $max_{num_r}$ do
                $coef \leftarrow random(0, max_{num_r})$
                $col \leftarrow random(1, N)$
                if $coef \neq 0$ & $A[rows_{Err}[i], \cdot] + coef - A[rows_{Err}[i], col] \leq max_{num_r}$ then
                    $\mathbf{A}[rows_{Err}[i], col] \leftarrow coef$
                else if $coef == 0$ & $\langle rows_{Err}[i], col\rangle \notin AIJ$ then
                    $\mathbf{A}[rows_{Err}[i], col] \leftarrow coef$
        if $rows_{Err}[i] \in BIJ[\cdot, 1]$ then
            for $k = 1$ to $len(BIJ)$ do
                if $BIJ[k, 1] == rows_{Err}[i]$ then
                    $\mathbf{B}[BIJ[k, 1], BIJ[k, 2]] \leftarrow 1$
            for $c = 0$ to $max_{num_p}$ do
                $coef \leftarrow random(0, max_{num_p})$
                $col \leftarrow random(1, N)$
                if $coef \neq 0$ & $B[rows_{Err}[i], \cdot] + coef - B[rows_{Err}[i], col] \leq max_{num_p}$ then
                    $\mathbf{B}[rows_{Err}[i], col] \leftarrow coef$
                else if $coef == 0$ & $\langle rows_{Err}[i], col\rangle \notin BIJ$ then
                    $\mathbf{B}[rows_{Err}[i], col] \leftarrow coef$
    return $\mathbf{A}, \mathbf{B}$
```

Algorithm 8 Checking of the linear independence between $\mathbf{A}[i, \cdot]$ and $\mathbf{B}[i, \cdot]$

```
function LINEAR_INDEPENDENCE2($\mathbf{A}, \mathbf{B}, rows_{Err}$)
    $err_{LinDep} \leftarrow [\,]$
    for $i = 1$ to $len(rows_{Err})$ do
        if $\mathbf{A}[rows_{Err}[i], \cdot] \wedge \mathbf{B}[rows_{Err}[i], \cdot]$ are linearly dependent then
            $err_{LinDep} \leftarrow err_{LinDep} \odot rows_{Err}[i]$
    return $err_{LinDep}$
```

Algorithm 9 Random initialization of the amounts/concentrations of the species

```
function AMOUNTS_GEN($N, \mathcal{D}_s, min_s, max_s, \mu_s, \sigma_s$)
    $M_0[\cdot] \leftarrow 0$
    if $dist$ is Uniform or Logarithmic then
        for $j = 1$ to $N$ do
            $\mathbf{M_0}[j] \leftarrow random(\mathcal{D}_s, min_s, max_s)$
    else
        for $j = 1$ to $N$ do
            $\mathbf{M_0}[j] \leftarrow random(\mathcal{D}_s, min_s, max_s, \mu_s, \sigma_s)$
    return $\mathbf{M_0}$
```

Algorithm 10 Random generation of kinetic constants of the reactions

```
1: function KINETIC_CONSTANTS_GEN($M, \mathcal{D}_r, min_r, max_r, \mu_r, \sigma_r$)
2:     $\mathbf{K}[\cdot] \leftarrow 0$
3:     if $dist$ is Uniform or Logarithmic then
4:         for $i = 1$ to $M$ do
5:             $\mathbf{K}[i] \leftarrow random(\mathcal{D}_r, min_r, max_r)$
6:     else
7:         for $i = 1$ to $M$ do
8:             $\mathbf{K}[i] \leftarrow random(\mathcal{D}_r, min_r, max_r, \mu_r, \sigma_r)$
9:     return K
```

References

1. Aldridge, B.B.; Burke, J.M.; Lauffenburger, D.A.; Sorger, P.K. Physicochemical modelling of cell signalling pathways. *Nat. Cell Biol.* **2006**, *8*, 1195–1203.
2. Szallasi, Z.; Stelling, J.; Periwal, V. *System Modeling in Cellular Biology: From Concepts to Nuts and Bolts*; The MIT Press, 2006.
3. Besozzi, D. Reaction-Based Models of Biochemical Networks. Pursuit of the Universal. 12th Conference on Computability in Europe, CiE 2016, Proceedings; Beckmann, A.; Bienvenu, L.; Jonoska, N., Eds.; Springer International Publishing: Switzerland, 2016; Vol. 9709, *LNCS*, pp. 24–34.
4. Nobile, M.S.; Tangherloni, A.; Rundo, L.; Spolaor, S.; Besozzi, D.; Mauri, G.; Cazzaniga, P. Computational Intelligence for Parameter Estimation of Biochemical Systems. Proc. Congress on Evolutionary Computation (CEC). IEEE, 2018, pp. 1–8.
5. Munsky, B.; Hlavacek, W.S.; Tsimring, L.S. *Quantitative biology: theory, computational methods, and models*; MIT Press, 2018.
6. Tangherloni, A.; Spolaor, S.; Cazzaniga, P.; Besozzi, D.; Rundo, L.; Mauri, G.; Nobile, M.S. Biochemical parameter estimation vs. benchmark functions: a comparative study of optimization performance and representation design. *Appl. Soft Comput.* **2019**, *81*, 105494.
7. Kitano, H. Systems biology: a brief overview. *Science* **2002**, *295*, 1662–1664.
8. Chou, I.C.; Voit, E.O. Recent developments in parameter estimation and structure identification of biochemical and genomic systems. *Math. Biosci.* **2009**, *219*, 57–83.
9. Somogyi, E.T.; Bouteiller, J.M.; Glazier, J.A.; König, M.; Medley, J.K.; Swat, M.H.; Sauro, H.M. libRoadRunner: a high performance SBML simulation and analysis library. *Bioinformatics* **2015**, *31*, 3315–3321.
10. 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* **2006**, *22*, 3067–3074.
11. Moraru, I.I.; Schaff, J.C.; Slepchenko, B.M.; Blinov, M.; Morgan, F.; Lakshminarayana, A.; Gao, F.; Li, Y.; Loew, L.M. Virtual Cell modelling and simulation software environment. *IET Syst. Biol.* **2008**, *2*, 352–362.
12. Li, H.; Petzold, L. Efficient parallelization of the stochastic simulation algorithm for chemically reacting systems on the graphics processing unit. *Int. J. High Perform. Comput. Appl.* **2010**, *24*, 107–116.
13. Zhou, Y.; Liepe, J.; Sheng, X.; Stumpf, M.P.; Barnes, C. GPU accelerated biochemical network simulation. *Bioinformatics* **2011**, *27*, 874–876.
14. Komarov, I.; D'Souza, R.M.; Tapia, J. Accelerating the Gillespie τ-leaping method using graphics processing units. *PLoS ONE* **2012**, *7*, e37370.
15. Komarov, I.; D'Souza, R.M. Accelerating the Gillespie exact stochastic simulation algorithm using hybrid parallel execution on graphics processing units. *PLoS ONE* **2012**, *7*, e46693.
16. Nobile, M.S.; Cazzaniga, P.; Besozzi, D.; Mauri, G. GPU-accelerated simulations of mass-action kinetics models with cupSODA. *J. Supercomput.* **2014**, *69*, 17–24.
17. Nobile, M.S.; Cazzaniga, P.; Besozzi, D.; Pescini, D.; Mauri, G. cuTauLeaping: A GPU-powered tau-leaping stochastic simulator for massive parallel analyses of biological systems. *PLoS ONE* **2014**, *9*, e91963.
18. Sumiyoshi, K.; Hirata, K.; Hiroi, N.; Funahashi, A. Acceleration of discrete stochastic biochemical simulation using GPGPU. *Front. Physiol.* **2015**, *6*, 42.
19. Tangherloni, A.; Nobile, M.; Besozzi, D.; Mauri, G.; Cazzaniga, P. LASSIE: simulating large-scale models of biochemical systems on GPUs. *BMC Bioinform.* **2017**, *18*, 246.
20. Tangherloni, A.; Nobile, M.S.; Cazzaniga, P.; Capitoli, G.; Spolaor, S.; Rundo, L.; Mauri, G.; Besozzi, D. FiCoS: a fine-and coarse-grained GPU-powered deterministic simulator for biochemical networks. *bioRxiv* **2021**.
21. Amara, F.; Colombo, R.; Cazzaniga, P.; Pescini, D.; Csikász-Nagy, A.; Muzi Falconi, M.; Besozzi, D.; Plevani, P. *In vivo* and *in silico* analysis of PCNA ubiquitylation in the activation of the Post Replication Repair pathway in *S. cerevisiae*. *BMC Syst. Biol.* **2013**, *7*, 24.
22. Besozzi, D.; Cazzaniga, P.; Pescini, D.; Mauri, G.; Colombo, S.; Martegani, E. The role of feedback control mechanisms on the establishment of oscillatory regimes in the Ras/cAMP/PKA pathway in *S. cerevisiae*. *EURASIP J. Bioinform. Syst. Biol.* **2012**, *2012*.

23. Cazzaniga, P.; Nobile, M.S.; Besozzi, D.; Bellini, M.; Mauri, G. Massive exploration of perturbed conditions of the blood coagulation cascade through GPU parallelization. *BioMed Res. Int.* **2014**, *2014*. Article ID 863298.
24. Pescini, D.; Cazzaniga, P.; Besozzi, D.; Mauri, G.; Amigoni, L.; Colombo, S.; Martegani, E. Simulation of the Ras/cAMP/PKA pathway in budding yeast highlights the establishment of stable oscillatory states. *Biotechnol. Adv.* **2012**, *30*, 99–107.
25. Renz, A.; Widerspick, L.; Dräger, A. Genome-Scale Metabolic Model of Infection with SARS-CoV-2 Mutants Confirms Guanylate Kinase as Robust Potential Antiviral Target. *Genes* **2021**, *12*, 796.
26. Gao, C.W.; Allen, J.W.; Green, W.H.; West, R.H. Reaction Mechanism Generator: Automatic construction of chemical kinetic mechanisms. *Comput. Phys. Commun.* **2016**, *203*, 212–225.
27. Khanshan, F.S.; West, R.H. Developing detailed kinetic models of syngas production from bio-oil gasification using Reaction Mechanism Generator (RMG). *Fuel* **2016**, *163*, 25–33.
28. Lok, L.; Brent, R. Automatic generation of cellular reaction networks with Moleculizer 1.0. *Nature Biotechnol.* **2005**, *23*, 131–136.
29. Städter, P.; Schälte, Y.; Schmiester, L.; Hasenauer, J.; Stapor, P.L. Benchmarking of numerical integration methods for ODE models of biological systems. *Sci. Rep.* **2021**, *11*, 1–11.
30. Garrido, A. Symmetry in complex networks. *Symmetry* **2011**, *3*, 1–15.
31. Ohlsson, F.; Borgqvist, J.; Cvijovic, M. Symmetry structures in dynamic models of biochemical systems. *J. R. Soc. Interface* **2020**, *17*, 20200204.
32. Keating, S.M.; Waltemath, D.; König, M.; Zhang, F.; Dräger, A.; Chaouiya, C.; Bergmann, F.T.; Finney, A.; Gillespie, C.S.; Helikar, T.; others. SBML Level 3: an extensible format for the exchange and reuse of biological models. *Mol. Syst. Biol.* **2020**, *16*, e9110.
33. Besozzi, D.; Cazzaniga, P.; Mauri, G.; Pescini, D. BioSimWare: a software for the modeling, simulation and analysis of biological systems. International Conference on Membrane Computing. Springer, 2010, pp. 119–143.
34. Chellaboina, V.; Bhat, S.P.; Haddad, W.M.; Bernstein, D.S. Modeling and analysis of mass-action kinetics. *IEEE Control Syst.* **2009**, *29*, 60–78.
35. Voit, E.O.; Martens, H.A.; W., O.S. 150 years of the mass action law. *PLoS Comput Biol* **2015**, *11*, e1004012.
36. Nelson, D.L.; Lehninger, A.L.; Cox, M.M. *Lehninger principles of biochemistry*, 5 ed.; Macmillan: London, UK, 2008.
37. Tangherloni, A.; Spolaor, S.; Rundo, L.; Nobile, M.S.; Cazzaniga, P.; Mauri, G.; Liò, P.; Merelli, I.; Besozzi, D. GenHap: a novel computational method based on genetic algorithms for haplotype assembly. *BMC Bioinform.* **2019**, *20*, 172.
38. Cho, Y.J.; Ramakrishnan, N.; Cao, Y. Reconstructing chemical reaction networks: data mining meets system identification. Proc. 14th ACM International Conference on Knowledge Discovery and Data Mining. ACM, 2008, pp. 142–150.
39. Dalcín, L.; Paz, R.; Storti, M. MPI for Python. *J. Parallel Distrib. Comput.* **2005**, *65*, 1108–1115.
40. Gropp, W.D.; Gropp, W.; Lusk, E.; Skjellum, A. *Using MPI: Portable Parallel Programming With the Message-Passing Interface*; Vol. 1, MIT press: Cambridge, MA, USA, 1999.
41. Chaouiya, C.; Remy, E.; Thieffry, D. Petri net modelling of biological regulatory networks. *J. Discr. Alg.* **2008**, *6*, 165–177.
42. Davidrajuh, R. Detecting Existence of Cycles in Petri Nets. Proc. of International Joint Conference SOCO'16-CISIS'16-ICEUTE'16. Springer, 2016, pp. 376–385.