A dynamical low-rank approach to solve the chemical master equation for biological reaction networks

Martina Prugger[a,b], Lukas Einkemmer[c], Carlos F. Lopez[b,d]

[a]*Department of Biochemistry, University of Innsbruck, Innsbruck, Tyrol, Austria*
[b]*Department of Biochemistry, Vanderbilt University School of Medicine, Nashville, TN, USA*
[c]*Department of Mathematics, University of Innsbruck, Innsbruck, Tyrol, Austria*
[d]*Department of Biomedical Informatics, Vanderbilt University Medical Center, Nashville, TN, USA*

Abstract

Solving the chemical master equation is an indispensable tool in understanding the behavior of biological and chemical systems. In particular, it is increasingly recognized that commonly used ODE models are not able to capture the stochastic nature of many cellular processes. Solving the chemical master equation directly, however, suffers from the curse of dimensionality. That is, both memory and computational effort scale exponentially in the number of species. In this paper we propose a dynamical low-rank approach that enables the simulation of large biological networks. The approach is guided by partitioning the network into biological relevant subsets and thus avoids the use of single species basis functions that are known to give inaccurate results for biological systems. We use the proposed method to gain insight into the nature of asynchronous vs. synchronous updating in Boolean models and successfully simulate a 41 species apoptosis model on a standard desktop workstation.

1. Introduction

Mechanistic models are commonly used to acquire insights about the biochemical reaction networks that govern cellular processes inside a cell. These models are typically obtained by a combination of expert and literature-driven knowledge as well as experimental data. Models based on ordinary differential equations (ODE) are most commonly used [12]. However, it is

also well known that in a number of problems the ODE formulation is insufficient to describe important features of the biological system [48, 25, 36, 38]. Often this inability comes from the fact that ODE models replace stochastic dynamics by some average. In contrast, such stochastic effects are taken into account by models based on the chemical master equation (CME).

The primary challenge associated with models based on the CME is the large computational cost. For a direct solver, the computational effort and memory requirements increase exponentially with the number of species, limiting such simulations to relatively small systems. This is a reason why the less computationally expensive ODE models are popular. In this paper, we employ a dynamical low-rank approximation to drastically reduce the computational and memory costs and therefore make the solution of large, complex systems tractable. In our approach, the network is divided into subsets that are tightly coupled, while only averaged information is exchanged between them. This enables simulations with a network size that would be prohibitive for direct solvers. Although low-rank approximations have been used in the past — particularly in the quantum physics literature — our approach departs from traditional applications by exploiting the underlying structure of the network rather than reducing the equations to single species basis functions (orbitals in the language of quantum dynamics). This represents a significant shift in how these methods should be applied in order to obtain accurate results with drastically reduced computational effort.

We note that in the literature the large computational cost of solving the CME has been addressed in a variety of ways. If a system is very sparse (i.e. only a few chemical states are populated) finite state projection [35] is a viable way to reduce computational complexity. However, especially for Boolean models (for details see below) this essentially assumes that a large number of species is either on or off with probability 1, a very unrealistic assumption. The other commonly used technique is Monte-Carlo simulation or SSA [22, 26]. In this approach trajectories in the space of possible configurations are sampled and a statistical representation is obtained by repeated simulations. Thus, to obtain, e.g., low abundant phenotypes requires the computation of an excessively large number (depending on the problem potentially tens to hundreds of thousands) of trajectories [23].

The idea of dynamical low-rank approximation is to use lower dimensional basis functions in order to approximate the high dimensional probability distribution. If a relatively small number of such basis functions is sufficient to describe the relevant dynamics, this results in a drastic reduction of the

required computational resources. In Jahnke et al. [27], such an approach is presented for a relatively low-dimensional CME. In this work, following the quantum physics literature, each species is considered its own subspace (i.e. the basis functions only depend on a single species). The proposed method works relatively well for the given examples ($\approx$ five species and $\approx$ ten reactions), but the authors are somewhat discouraging regarding the wider use of the method. The reason for this is that in biological applications, it would be difficult or even impossible to consider each species as independent, particularly given the intricate structures of complex biological networks [3]. Therefore, approximating the probability distribution of a system of reactions using single species basis function could significantly degrade the accuracy of the method. Our approach avoids this by using insight into the structure of the reaction network when choosing the dependence of the basis functions. This results in a dynamical low-rank algorithm that can accurately simulate many real world reaction networks at low computational cost.

Our proposed dynamical low-rank approximation formalism for biological networks leverages prior work. In particular, the projector splitting based dynamical low-rank approach introduced by Lubich et al. [29] and its later extension to hierarchical tensors [30]. This was the first robust low-rank approach able to handle dynamical systems [34]. More recently, this approach found success in solving particle kinetic equations such as the Vlasov equation [18, 19, 17] and radiative transfer problems [40, 15, 14, 16]. Our dynamical low-rank approach is fully deterministic (i.e. noise free) and provides the entire probability distribution with a single simulation. This enables us to get an accurate picture of the network dynamics, while still being relatively cheap from a computational point of view. In addition, the proposed approach does not show unfavorable scaling with the number of reactions (as does SSA), but only with the dependent species for each reaction. The approximation error in the dynamical low-rank approach is controlled by how the network is partitioned, thus allowing us to make use of expert knowledge to refine the result.

While the dynamical low-rank approach can be applied to any model based on the CME, in this paper we consider Boolean models [54]. In this formalism, each biochemical species (e.g. genes, proteins) can exist in two states (on or off), which significantly simplifies model complexity and only requires that one knows the structure of the network (which in some situations can be inferred from data [41]). The state of the network is then determined by a binary string that encodes which species are switched on and which

are switched off. In the CME approach a probability is assigned to each such state. Boolean models have proven very useful to extract mechanistic insights from complex systems [52, 13] and a range of software packages are available for practitioners. We note that the Boolean model formalism is also used in a variety of applications outside biology including quantum physics, circuit theory, natural language processing, and search engines [37, 5]. For increasingly large and complex problems Boolean models have the added advantage that they do not require precise knowledge of a large number of kinetic parameters, which otherwise must be either measured experimentally, or inferred using statistical methods [46, 2].

2. Dynamical low-rank approximation

2.1. Setup

The chemical master equation is given by

$$\frac{\mathrm{d}P(N,t)}{\mathrm{d}t} = \sum_{r\in\mathcal{R}} a_r(N-\nu_r)P(N-\nu_r,t) - \sum_{r\in\mathcal{R}} a_r(N)P(N,t), \tag{1}$$

where $N(t) = (N_1(t), N_2(t), ..., N_n(t))$ and N_i is the number of molecules of the chemical species of type i. The number of distinct chemical species is denoted by n. $P(N,t)$ describes the state of the system determined by the set of all possible chemical reactions $\mathcal{R}$ and the reaction propensities $a_r(N)$, where ν_r represents the change of species due to reaction $r \in \mathcal{R}$.

In the case of a Boolean model, instead of number of molecules, we have states $x_i \in \{0,1\}$, where each x_i represents the state of a chemical species in the system. That is, each chemical species can only be in one of two states (often referred to as on and off, or active and inactive). The chemical master equation describes the dynamics (i.e. time evolution) of a probability distribution $P(t, x_1, \ldots, x_d)$. Boolean models are often specified by a set of (logic) rules. For an explanation of the translation of a Boolean ruleset describing a biological network into a system of ODEs, we refer the reader to Appendix A.

To apply the dynamical low-rank algorithm we partition the d nodes of the network into two groups containing $\{x_1, ..., x_m\}$ and $\{x_{m+1}, ..., x_d\}$ respectively. Ideally, the two partitions are as equal as possible (i.e., $m = d/2$). Note, that within this framework we can use an arbitrary partition of the network by simply renaming the nodes. The performance of the algorithm

is crucially dependent on how this partitioning is done. Both domain specific knowledge (i.e. knowledge on the behavior of the reaction network) and mathematical techniques (such as graph partitioning algorithms) can be used to find good partitionings. We will discuss this in more detail later in the paper.

We now write P in matrix form by linearizing the elements within each partition. That is, we linearize a state according to the function

$$\alpha(x_1, ..., x_k) = x_1 + 2 \cdot x_2 + ... + 2^{m-1} x_k.$$

We then consider the following matrix

$$P_{\alpha(x_1,...,x_m),\alpha(x_{m+1},...,x_d)} = P(x_1, ..., x_d).$$

This means, that the index of the state in the first partition determines the row index in the matrix, and the index of the state in the second partition determines the column index.

To address the curse of dimensionality, we avoid forming the full matrix P (i.e. nowhere in the algorithm will it be necessary to store the full matrix P), but instead represent it by a low-rank approximation

$$P = USV^T,$$

where $U \in \mathbb{R}^{2^m \times r}$, $S \in \mathbb{R}^{r \times r}$, and $V \in \mathbb{R}^{2^{d-m} \times r}$ and r is the rank of the approximation. These so-called low-rank factors are stored in computer memory. If r can be chosen small, storing the low-rank factors results in a drastic reduction of the memory required to represent the state of the system. Since most models of signaling networks try to model the effect of an input signal on a downstream target, there is usually a couple of pathways that are described to reach this target. Even though this can also include some feedback of the pathways to the upper species, the connections within the pathways are usually from one species to the next. This results in a fairly directed graph, where most of the species have few direct connections with other species. We, therefore, predict that most signaling networks are well represented by a low-rank approximation as relatively few cuts are needed in order to partition the network. Thus, it is possible to keep the error from the partitioning low and therefore only a low rank is necessary to achieve accurate simulations.

2.2. Time step

Knowing the low-rank factors at one point in time we now require a way to efficiently update them according to the dynamics of the chemical master equation. To do this we employ a dynamical low-rank integrator. More specifically, the projector splitting integrator proposed in [29]. This approach is outlined in Algorithm 1.

Algorithm 1: Projector splitting dynamical low-rank algorithm. In each time step the low-rank factors are updated according to the described algorithm.

Input: U_0, S_0, V_0
Output: U_1, S_1, V_1

1. $K_1 = U_0 S_0 + (\Delta P) V_0$;
2. QR decomposition to get $U_1 \hat{S}_1 = K_1$;
3. $\tilde{S}_0 = \tilde{S}_1 - U_1^T (\Delta P) V_0$;
4. $L_1 = V_0 \tilde{S}_0^T + (U_1^T \Delta P)^T$;
5. QR decomposition to get $V_1 S_1^T = L_1$

Since the chemical master equation is linear, we can write it abstractly as

$$\frac{\mathrm{d}P(x,t)}{\mathrm{d}t} = AP(x,t).$$

In each time step we update P according to Algorithm 1 where (in the simplest case of explicit Euler) we have

$$\Delta P = P^1 - P^0 = \tau AP,$$

where τ is the time step size. The complexity of the QR decomposition is $\mathcal{O}(r^2 2^m)$ and $\mathcal{O}(r^2 2^{d-m})$, respectively. The crucial step of the algorithm is how to compute $(\Delta P)V_0, U_1^T(\Delta P)$, and $U_1^T(\Delta P)V_0$ (the orange parts in the algorithm). Simply forming ΔP and multiplying it with V_0, e.g., would be extremely expensive. That is, we have to use the structure of the chemical master equation in order to efficiently compute these three terms. This is the primary challenge that the application of the dynamical low-rank algorithm to any problem faces and we will thus address it in the following for the case of Boolean reaction networks.

2.3. Compute $(\Delta P)V_0$

We first write $\Delta P = \tau AP = \tau A_1 P + ... + \tau A_d P$. A rule (for more details see Appendix A) has the form $x_{p0} = F(x_{p_1}, ..., x_{p_s})$, with s usually small and F a Boolean expression. Each rule can be written in matrix form (i.e. as a linear operator) as follows

$$\begin{aligned}
(A_k P)(x) &= \delta_{\{x_p\}} P(T_{p_0} x) - \overline{\gamma_{\{x_p\}}} P(x) \\
\gamma_{\{x_p\}} &= \delta_{x_{p_0}, F(x_{p_1}, ... x_{p_s})} \\
\delta_{x_p} &= \delta_{x_{p_0}, F(x_{p_1}, \overline{x}_{p_0}, ... x_{p_s})} \\
T_{p_0}(x_1, ..., x_{p_0-1}, x_{p_0}, x_{p_0+1}, ..., x_d) &= T_{p_0}(x_1, ..., x_{p_0-1}, \overline{x_{p_0}}, x_{p_0+1}, ...x_d),
\end{aligned}$$

where negation is denoted by a line above the expression. We will use the convention that if T_{p_0} is applied to an index that does not contain p_0 it is just the identity. For simplicity, we will also write $x_{G1} = (x_1, ..., x_m)$, $x_{G2} = (x_{m+1}, ..., x_d)$, and $x_R = (x_{p_0}, ..., x_{p_1})$.

Since there are at most d rules it is sufficient to consider them one by one. Then we have

$$\begin{aligned}
(A_k P V_0)(x_{G1}, l) &= \sum_{x_{m+1} ... x_d} [\gamma_{x_R} P(T_{p_0} x) - \overline{\gamma_{x_R}} P(x)] V_0(x_{G2}, l) \\
&= \sum_{x_{m+1} ... x_d} \gamma_{x_R} P(T_{p_0} x) V_0(x_{G2}, l) - \sum_{x_{m+1} ... x_d} \overline{\gamma_{x_R}} P(x) V_0(x_{G2}, l) \\
&= \sum_{x_{m+1} ... x_d} \gamma_{x_R} \left[\sum_{ij} U(T_{p_0} x_{G1}, i) S_{ij} V(T_{p_0} x_{G2}, j) \right] V_0(x_{G2}, l) \\
&\quad - \sum_{x_{m+1} ... x_d} \overline{\gamma_{x_R}} \left[\sum_{ij} U(x_{G1}, i) S_{ij} V_j(x_{G2}, j) \right] V_0(x_{G2}, l) \\
&= \sum_{ij} S_{ij} U(T_{p_0} x_{G1}, i) \sum_{x_{m+1} ... x_d} \gamma_{x_R} V(T_{p_0} x_{G2}, j) V_0(x_{G2}, l) \\
&\quad - \sum_{ij} S_{ij} U_i(x_{G1}) \sum_{x_{m+1} ... x_d} \overline{\gamma_{x_R}} V(x_{G2}, j) V_0(x_{G2}, l) \\
&=: \sum_{ij} S_{ij} U(T_{p_0} x_{G1}, i) \Gamma^1_{jl}(x_R \cap x_{G1}) - \sum_i S_{ij} U_i(x_{G1}, i) \Gamma^2_{jl}(x_R \cap x_{G1}),
\end{aligned}$$

where

$$\Gamma^1_{jl}(x_R \cap x_{G1}) := \sum_{x_{m+1} ... x_d} \gamma_{x_R} V(T_{p_0} x_{G2}, j) V_0(x_{G2}, l)$$

and

$$\Gamma^2_{jl}(x_R \cap x_{G1}) := \sum_{x_{m+1} ... x_d} \overline{\gamma_{x_R}} V(x_{G2}, j) V_0(x_{G2}, l).$$

Algorithm 2:

1 For each $x_R \cap x_{G1}, l, j$ compute $\Gamma^1_{lj}(x_R \cap x_{G1})$ and $\Gamma^2_{lj}(x_R \cap x_{G1})$;
2 For each (x_{G1}, l) compute $(A_k P)V_0$;
3 Compute $(\Delta P)V_0 = \tau \sum_k (A_k P)V_0$;

We then proceed as stated in Algorithm 2.

The first step is at most $\mathcal{O}(2^{\#(x_r \cap x_{G1})} r^2 2^{d-m})$. The second step is at most $\mathcal{O}(r^3 2^m)$. The third step is $\mathcal{O}(k2^m)$. This means that the efficiency is dramatically improved as those estimates do not depend on 2^d.

All of this estimates are rather pessimistic. A particular interesting case for optimization of the code is where a rule is completely contained within one partition. In this case there are two possibilities:

Case 1: $x_R \subset x_{G1}$: We have

$$\begin{aligned}
(A_k P V_0)(x_{G1}, l) &= \sum_{ij} S_{ij} U(T_{p_0}(x_{G1}), i) \gamma_{x_R} \sum_{x_{m+1} \ldots x_d} V(T_{p_0} x_{G2}, j) V_0(x_{G2}, l) \\
&\quad - \sum_{ij} S_{ij} U(x_{G1}, i) \overline{\gamma_{x_R}} \sum_{x_{m+1} \ldots x_d} V(x_{G2}, j) V_0(x_{G2}, l) \\
&= \sum_{ij} S_{ij} U(T_{p_0} x_{G1}, i) \gamma_{x_R} \delta_{jl} - \sum_{ij} S_{ij} U_i(x_{G1}) \overline{\gamma_{x_R}} \delta_{jl} \\
&= \sum_i \gamma_{x_R} U(T_{p_0} x_{G1}, i) S_{il} - \sum_i \overline{\gamma_{x_R}} U_i(x_{G1}) S_{il}.
\end{aligned}$$

The second line follows from the orthogonality condition. This is just the direct application of the rule to U (i.e. no error is made). The first step in Algorithm 2 can then be omitted. The second step is only $\mathcal{O}(r^2 2^m)$.

Case 2: $x_R \subset x_{G2}$: In this case we have $x_R \cap x_{G1} = \emptyset$ and thus the first step is only $\mathcal{O}(r^{2^{d-m}})$.

2.4. Compute $U_1^T \Delta P$

We have

$$\begin{aligned}
U_1^T A_k(x_{G2}, l) &= \sum_{x_1 \ldots x_m} U_1(x_{G1}, l) \left[\gamma_{x_R} P(T_{p_0} x) - \overline{\gamma_{x_R}} P(x) \right] \\
&= \sum_{ij} S_{ij} V(T_{P_0} x_{G2}, j) \sum_{x_1 \ldots x_m} \gamma_{x_R} U(T_{p_0} x_{G1}, i) U_1(x_{G1}, l) \\
&\quad - \sum_{ij} S_{ij} V(x_{G2}, j) \sum_{x_1 \ldots x_m} \overline{\gamma_{x_R}} U(x_{G1,i}) U_1(x_G 1, l) \\
&:= \sum_{ij} S_{ij} V(T_{p_0} x_{G2}, j) \Gamma^3_{il}(x_R \cap x_{G2}) - \sum_j S_{ij} V(x_{G2}, j) \Gamma^4_{il}(x_R \cap x_{G2}).
\end{aligned}$$

The algorithm proceeds exactly as in the previous case and we can once again make significant simplifications for the case where x_R is contained within a partition. For $x_R \subset x_{G2}$ we have

$$\begin{aligned} U_1^T A_k(x_{G2}, l) &= \sum_{ij} S_{ij} V(T_{P_0} x_{G2}, j) \sum_{x_1 \ldots x_m} \gamma_{x_R} U_1(x_{G1}, l) U(T_{p_0} x_{G1}, i) \\ &\quad - \sum_{ij} S_{ij} V(x_{G2}, j) \sum_{x_1 \ldots x_m} \overline{\gamma_{x_R}} U_1(x_{G1}, l) U_i(x_{G1}) \\ &= \sum_{ij} S_{ij} V(T_{P_0} x_{G2}, j) \gamma_{x_R} \delta_{il} - \sum_{ij} S_{ij} V(x_{G2}, j) \overline{\gamma_{x_R}} \delta_{il} \\ &= \sum_{j} S_{lj} V(T_{P_0} x_{G2}, j) \gamma_{x_R} - \sum_{j} S_{lj} V(x_{G2}, j) \overline{\gamma_{x_R}}. \end{aligned}$$

2.5. Compute $U_1^T \Delta P V_0$

This is now easy since we already have computed $\Delta P V_0$ (in Subsection 2.3). So this is just a matrix-matrix product.

Before we proceed, let us make a few remarks.

Remark. *The algorithm becomes more efficient if as many rules as possible are contained in one partition. This is also advantageous from an accuracy standpoint, since rules contained in one partition do not incur any low-rank error.*

Remark. *The notation for writing down the algorithm is somewhat complicated. However, the interpretation is very easy: if a rule is contained in one partition just apply the rule directly in the low-rank format. If not, then the states in the partition not containing x_p are averaged, and the rule is applied with respect to that averaged state.*

3. An illustrative example: a Boolean model of the mTOR pathway

To illustrate the use and efficiency of the low-rank approximation we consider a simple example; namely, the mTOR pathway model introduced by Benso et al. [6]. The model includes 22 species and its network diagram is

given in Figure 1. The full Boolean ruleset defining the network is as follows

$$
\begin{aligned}
ERK &= ERK \\
P27361 &= ERK \\
RSK &= RSK \\
Q15418 &= RSK \\
mir-1976 &= RSK \\
MLL &= MLL \\
Q03164 &= MLL \wedge \overline{mir-1976} \\
HOAX9 &= Q03164 \\
mir-196b &= HOAX9 \\
TSC1/2 &= \overline{P27361 \wedge Q15418} \wedge \overline{P31749} \\
P31749 &= AKT \\
PDK1 &= PDK1 \\
Q15118 &= PDK1 \\
AKT &= Q15118 \wedge Q6R327 \wedge P42345 \wedge Q9BVC4 \\
RHEB &= \overline{TSC1/2} \\
Q15382 &= RHEB \\
GBL &= Q15382 \\
Q9BVC4 &= GBL \\
mTOR &= Q15382 \\
P42345 &= mTOR \\
RICTOR &= Q15382 \\
Q6R327 &= RICTOR \wedge \overline{mir-196bs}.
\end{aligned}
\tag{2}
$$

3.1. Network partitioning

The first step in our dynamical low-rank approach requires the partitioning of the network into subnetworks while leveraging the network structure. For this step, we require that the partitioning of the network yields subnetworks with an approximately equal number of species. Every interaction that remains within its partition (e.g. the rule for *TSC1/2* in the blue partitioning in Figure 1) is treated exactly by the algorithm. Therefore, we want to minimize cross-partition dependencies (indicated by arrows in the figures) as these interactions will be subject to low-rank approximations (e.g. the pathway that connects *TSC1/2* to *RHEB* in the blue partitioning). In Figure 1, we introduce five different partitionings that we will use in the subsequent analysis. The blue, green, purple, and red networks are chosen following this general principle of minimizing connections between the subnetworks. To

demonstrate the consequences of a bad partitioning, we also included the yellow partition, where the species are randomly assigned to a partition.

The most important parameter of the low-rank approximation is the rank. A smaller rank r leads to faster computation times and less memory requirements for the simulation. However, decreasing rank can also lead to larger error. The maximum rank of a simulation is given by $2^{d/2}$, where d is the number of chemical species. In our example, d=22 and thus the maximum possible rank would be 2048. At the highest rank, the simulation is exact but this does not result in any computational improvements in compute time or memory use relative to a direct solver. In Figure 2, we compare the errors of five different network partitions in relation to the exact solution for different ranks (namely, $r = 2$, $r = 4$, $r = 8$, and $r = 16$). It is assumed that initially all states of the system are equally likely. For the blue (reference partition scheme) we can see that in order to obtain an error on the order of 1% (0.01) only rank 4 is required. To reduce this error to 10^{-3} the rank has to be increased to approximately 16. We also clearly see from Figure 2, by comparing e.g. the blue with the yellow partitioning, that choosing an appropriate partitioning can significantly reduce the rank and thus computational effort. Since we chose the yellow network randomly, we do not expect the partitioning to be competitive. In fact, as we can see, even increasing the rank to 16 does not decrease the error below its initial threshold, while the other partitionings see a significant reduction in the error. We also note that minimizing the connections between two partitions is not enough information in order to obtain an ideal partitioning. This can be seen in the behavior of the red and the green network. Both networks have five edges that separate the two partitions, but their accuracy is very different. The reason for this is that both the green and the red network disturb the important pathway from *Q15382* to *AKT* in a significant way. In this case insight about the problem under consideration can help to choose an optimal partitioning and thus reduce the numerical effort.

3.2. Numerical results for the steady states and dynamics of mTOR

The proposed dynamical low-rank integrator is able to obtain all the statistical information in a single simulation because it is directly solving the chemical master equation. The results of the corresponding simulation are shown in in Figure 3. We display only the steady states that have a probability that exceeds 10^{-3} (20 in total for this model). For very small ranks the approximation becomes inaccurate and SS9 and SS10 are lost.

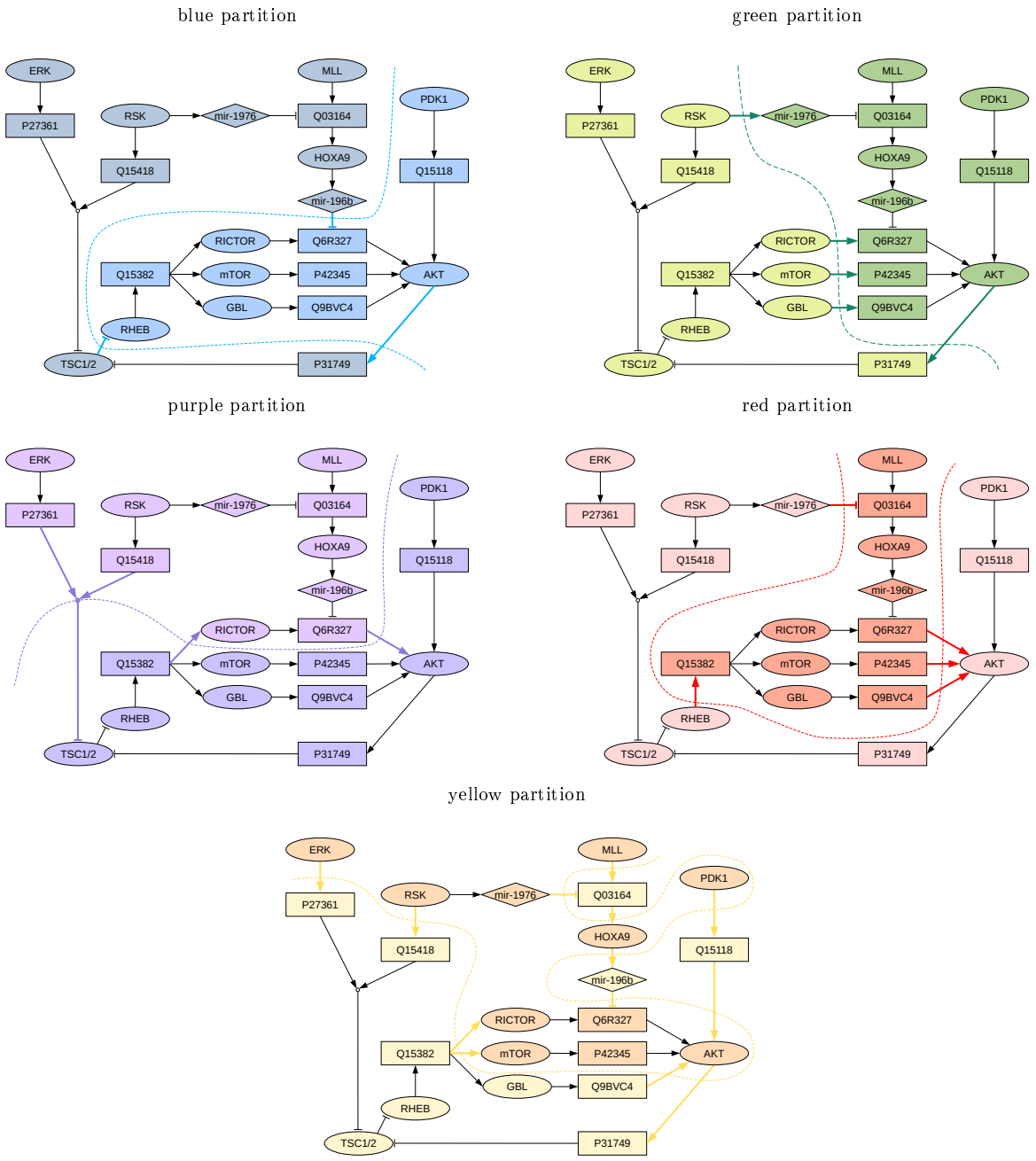

Figure 1: Boolean networks for the mTOR pathway model. We consider five different partitionings for the low-rank algorithms (blue, green, purple, red, and yellow). How the partitioning is done influences how small the rank (which is proportional to computational cost) can be chosen for a given accuracy. The blue and purple partitioning perform best as they treat the important pathway from *Q15382* to *AKT* with minimal error.

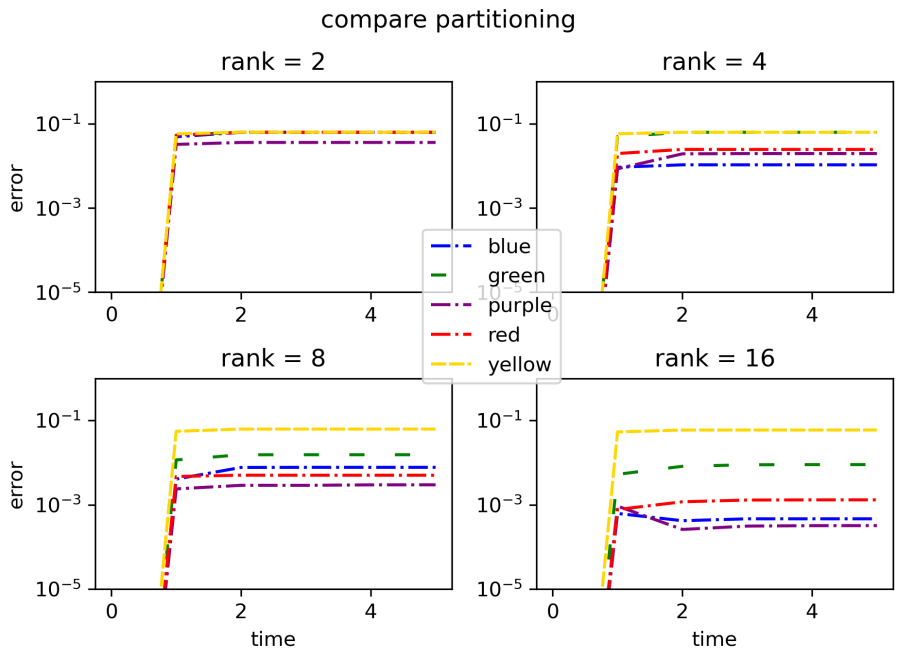

Figure 2: The error of the dynamical low-rank approximation for the five different partitionings of the mTOR network is shown for ranks 2, 4, 8, and 16. The error is the maximum of the difference in probability between the low-rank approximation and the exact solution at a given time (i.e. the error in P measured in the L^2 norm).

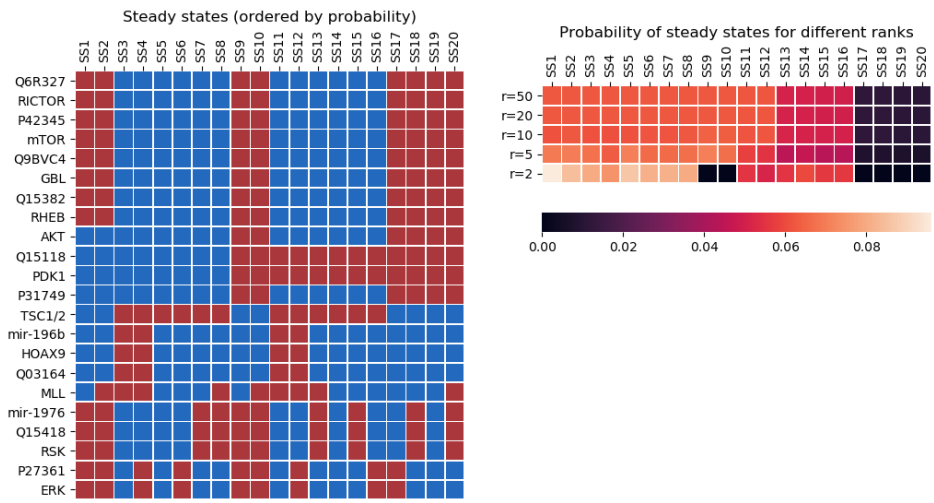

Figure 3: The steady states with probability larger than 10^{-3} are shown, sorted by their respective probability (red indicates on, blue indicates off). The blue partitioning and an initial value in which all states are equally likely are used.

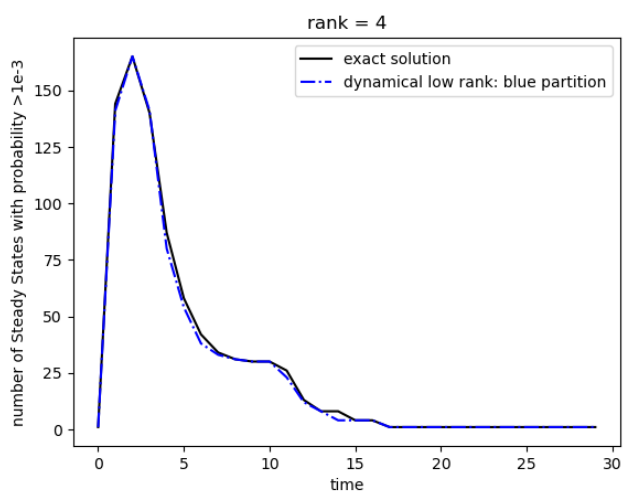

Figure 4: The number of states with probability larger than 10^{-3} for the mTOR network is shown as a function of time. Initially we start with a deterministic initial value (which is rank 1) and ultimately we end up in a single steady state with high probability (again rank 1). However, a large number of trajectories is generated intermittently in this dynamics.

However, even for $r = 5$ all of the relevant steady states are captured and starting at rank $r = 10$ the probabilities match exactly those of the reference solution.

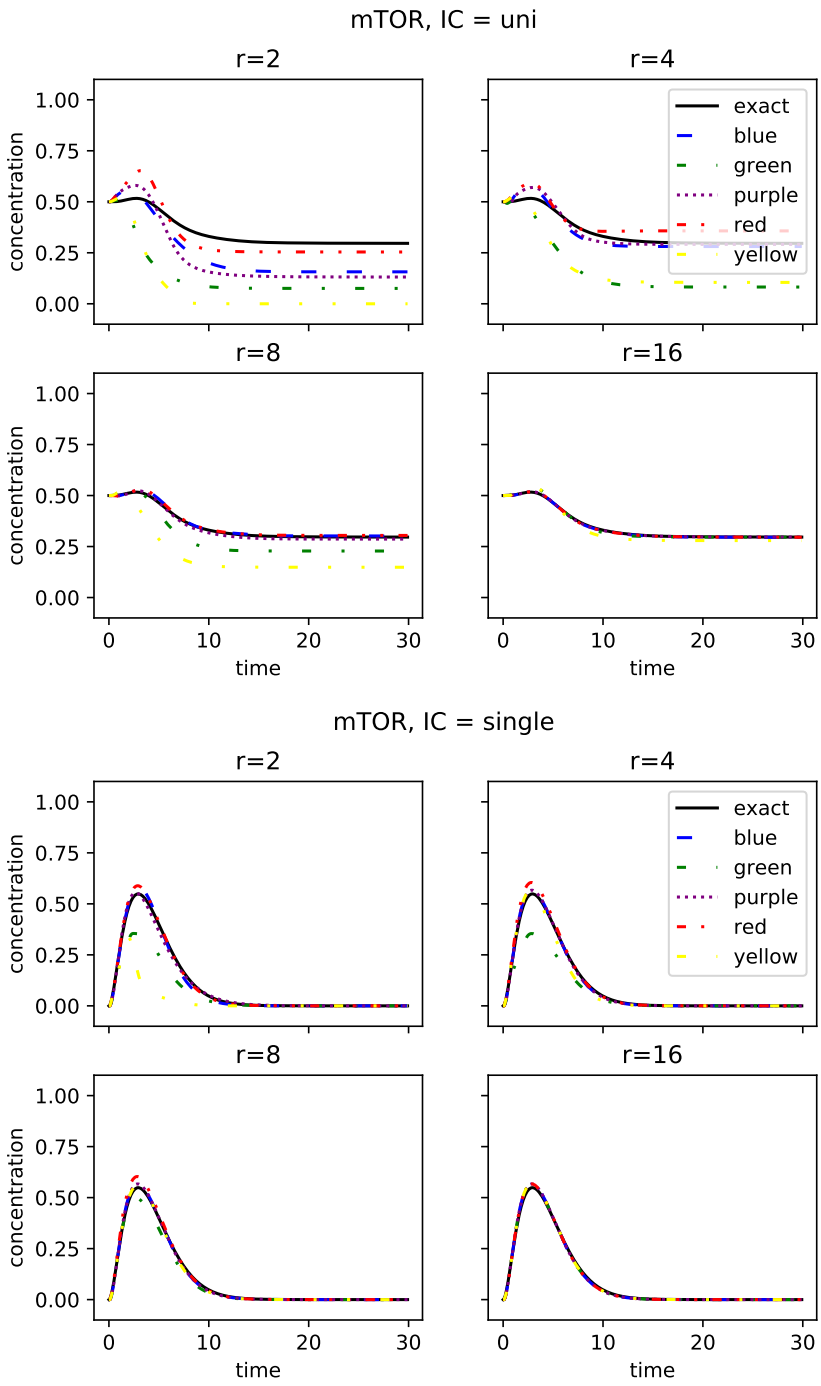

Figure 5: The time evolution of the probability that $mTOR$ is switched on is shown. The configuration on the top starts with an initial configuration where $mTOR$ initially is equally likely to be in an on or an off state. The configuration on the bottom has $mTOR$ initially switched off and after a transient phase that is mediated by a significant concentration of $mTOR$ results in a steady state where $mTOR$ is switched off again (with high probability).

In addition to the steady states, the dynamical low-rank algorithm also provides information about which trajectories lead to a given steady states as well as their probability. In order to study such time-dependent effects the dynamical low-rank approach is particularly useful. E.g. Consider the dynamics in Figure 4, where initially only a few states have a non-zero probability. Moreover, after some time the system settles in a small number of steady states. Both of these would be well represented by finite state projection. However, in the dynamics that the system follows to reach these few steady states we see an explosion in the number of states with non-zero probability. The dynamical low-rank approach, in contrast to finite state projection (which here again would suffer from the curse of dimensionality), is able to resolve these intermediate states without any increase in computational cost as we can see from Figure 4.

Finally, we look at the dynamics predicted by the dynamical low-rank algorithm. In Figure 5 we plot the time evolution of $mTOR$ for the different partition schemes and different ranks for the uniform initial condition (left) and the single initial condition (right). We observe that the blue and purple partitions in particular give results in agreement with the exact solution, even for rank 4. For rank 16 all partition schemes display the correct behavior.

We conclude that for appropriately chosen partitionings even a small rank (i.e. low computational cost) is sufficient to obtain accurate results (both in terms of steady states and dynamics). If, for some reason, a good partitioning is not available, the rank needs to be increased which adds to the computational and memory cost.

In our code, we do not only provide the projector-splitting integrator, but also more recently developed integrators, i.e., the BUG-integrator [10], as well as the augmented BUG-integrator [9]. In Table 1, we compare the different performances of the integrators for solving the CME. As we can see from the table, the error of the BUG integrator is quite high. However, once we perform the augmentation, the error is almost identical to the projector splitting approach. We also note that both the BUG integrator (due to the need to recompute the coefficients for the S step) as well as the augmented BUG integrator (since the rank for the S step is increased) are somewhat more expensive compared to the projector splitting integrator.

method	projector splitting	BUG	augmented BUG
rank	L2-error [time/step]	L2-error [time/step]	L2-error [time/step]
r2	0.5993 [0.16967 s]	1.09644 [0.23094 s]	0.60158 [0.25043 s]
r4	0.0701 [0.19262 s]	0.58711 [0.27606 s]	0.06674 [0.33073 s]
r8	0.05625 [0.33002 s]	0.13605 [0.4693 s]	0.05166 [0.58129 s]
r16	0.00537 [0.46486 s]	0.04531 [0.68406 s]	0.00714 [0.98606 s]

Table 1: Comparison of the performance of the three different integrators that we provide in our implementation. The error is measured in the L^2 norm at time point 20.

4. Computational efficiency

The mTOR network described in some detail in the previous section with its 22 species is still a relatively small model. Ultimately, we are interested in using the dynamical low-rank approach for larger problems. In the subsequent sections we will discuss two examples. A model of pancreatic cancer with 34 species and a more detailed model of apoptosis with 41 species. To illustrate the dramatic growth of memory requirements that results from increasing the number of species let us assume that a direct solver would only need to store a single input and output vector (a highly questionable assumption as a real implementation would most likely significantly exceed this theoretical value). In this case, the pancreatic cancer model would require 275 GB of main memory (RAM). This renders it beyond the scope of a desktop workstations and would require parallelization to a distributed memory supercomputer. For the apoptosis model considered the requirements would be even more severe. In fact, 35 TB of RAM would be required in this case, which is clearly not feasible.

In Table 2 (top) we show that our C++ implementation of the low-rank algorithm drastically reduces these requirements. For the pancreatic cancer problem running simulations with $r = 10$, which gives very accurate results (as we will see), only 421 MB are required which is a reduction by a factor of 650. For the apoptosis model this is even more pronounced, as we would expect. In this case 12 GB are required for $r = 10$, a reduction by a factor of approximately 3000. Together with the timings provided in Table 2 (bottom) this clearly demonstrates that using the dynamical low-rank approach proposed here enables us to easily conduct these simulations on a desktop workstation within a reasonable amount of wall clock time. Thus, the only question that needs to be answered is how accurate the low-rank approximation is for realistic applications. This will be the goal of the

problem	$r = 2$	$r = 5$	$r = 10$	$r = 20$	$r = 30$	$r = 50$	direct simulation
mTOR	34 MB	38 MB	83 MB	191 MB	–	113 MB	> 66 MB
pancreatic	160 MB	267 MB	421 MB	671 MB	–	1.4 GB	> 275 GB
apoptosis	2.5 GB	6.1 GB	12 GB	24 GB	36 GB	–	> 35 TB

problem	$r = 2$	$r = 5$	$r = 10$	$r = 20$	$r = 30$	$r = 50$
mTOR	0.6 s	1.1 s	3.4 s	4.3 s	–	10 s
pancreatic	1 m 39 s	2 m 55 s	5 m 37 s	9 m 08 s	–	28 m 47 s
apoptosis	13 m 08 s	33 m 21 s	1 h 06 m 33 s	1 h 39 m 06 s	2 h 39 m 35 s	–

Table 2: Main memory (RAM) requirements for the mTOR pathway model (22 species), pancreatic cancer (34 species), and apoptosis (41 species) for the dynamical low-rank algorithm and different values of the rank r is shown on the top. For comparison, a theoretical lower bound for the memory that a direct solver would require is also provided (note, however, that an actual implementation will most likely significantly exceed this value). In all simulations the reference partitioning (blue and setup a, respectively) is used. On the bottom, timing measurements for the dynamical low-rank algorithm and different values of the rank r are presented. The problem is integrated until a final time $T = 1$ and all simulations have been run on a Intel Core i9-7940X CPU with 14 cores and 64 GB of DDR4 memory.

remainder of the paper.

5. Pancreatic cancer model and some observations on asynchronous vs. synchronous updating

Results using the stochastic simulation algorithm (SSA) can be obtained using two distinct update rules. In synchronous updating all rules are applied at the same time and thus all species are updated simultaneously. While synchronous updating is commonly used, there is significant debate as to its biological interpretation [4, 20, 33, 45]. Asynchronous updating has been proposed as an alternative [39, 49, 50, 51, 53]. In asynchronous updating a single reaction is selected at random and then the corresponding rule is applied. As such, it has a direct interpretation in terms of chemical reactions and it has been shown that asynchronous updated SSA in the limit of a large number of samples gives the same dynamics as the chemical master equation [42, 47].

Nevertheless, it has been argued that synchronous updating is sufficient in order to obtain the relevant steady states [44, 43, 28, 7, 8]. However, recent

work, performed using direct and SSA simulation, has called this assertion into question [21, 1]. Our dynamical low-rank approach gives us a way to easily investigate the full dynamics of large systems in this context. As it is based on the chemical master equation, the dynamical low-rank approach encodes the biologically relevant asynchronous updating, while allowing us to obtain statistics over a range of trajectories and initial values with a single run of the simulation.

Gong et al. [24] use mathematical techniques in order to prove that certain properties of the Boolean network are compatible with the experimental observations. For example, it is shown that overexpression of *HMGB1* will necessarily result in steady states with *Proliferate* switched on and *Apoptosis* switched off. However, these results only apply to synchronous updating. The dynamical low-rank approach, on the other hand, allows us to investigate this for the chemical master equation (i.e. with asynchronous updating). The time evolution of the probability that the cell is in the *Proliferate* state or that *Apoptosis* was triggered is shown in Figure 6. Note that we conclude from these results that in the asynchronous case the dynamics are significantly more complicated. It is possible, that apoptosis is triggered if *HMGB1* is turned on (the steady-state probability is approximately 16%). Moreover, there is no difference in the ultimate cell fate (i.e. with respect to proliferation or apoptosis) whether *HMGB1* is switched on or off. Thus, there is a significant difference between the steady states in the asynchronous case compared to the synchronous case that is discussed in Gong et al. [24]. This casts significant doubt that the model (at least in the asynchronous case) can describe the experimentally observed effects accurately. This also highlights that models derived in the synchronous case can not (at least not without alteration) be considered a viable description of the relevant biology.

Our results show that even if one is only interested in the steady states, the result for synchronous and asynchronous updating may exhibit significant differences. The dynamical low-rank approach is a tool particularly well suited for obtaining steady states that are consistent with the biologically more relevant asynchronous updating. The most likely steady states are shown in Figure 6. We once again highlight the fact that even a small rank (e.g. $r = 5$) is sufficient to capture the steady states as well as their probability very well.

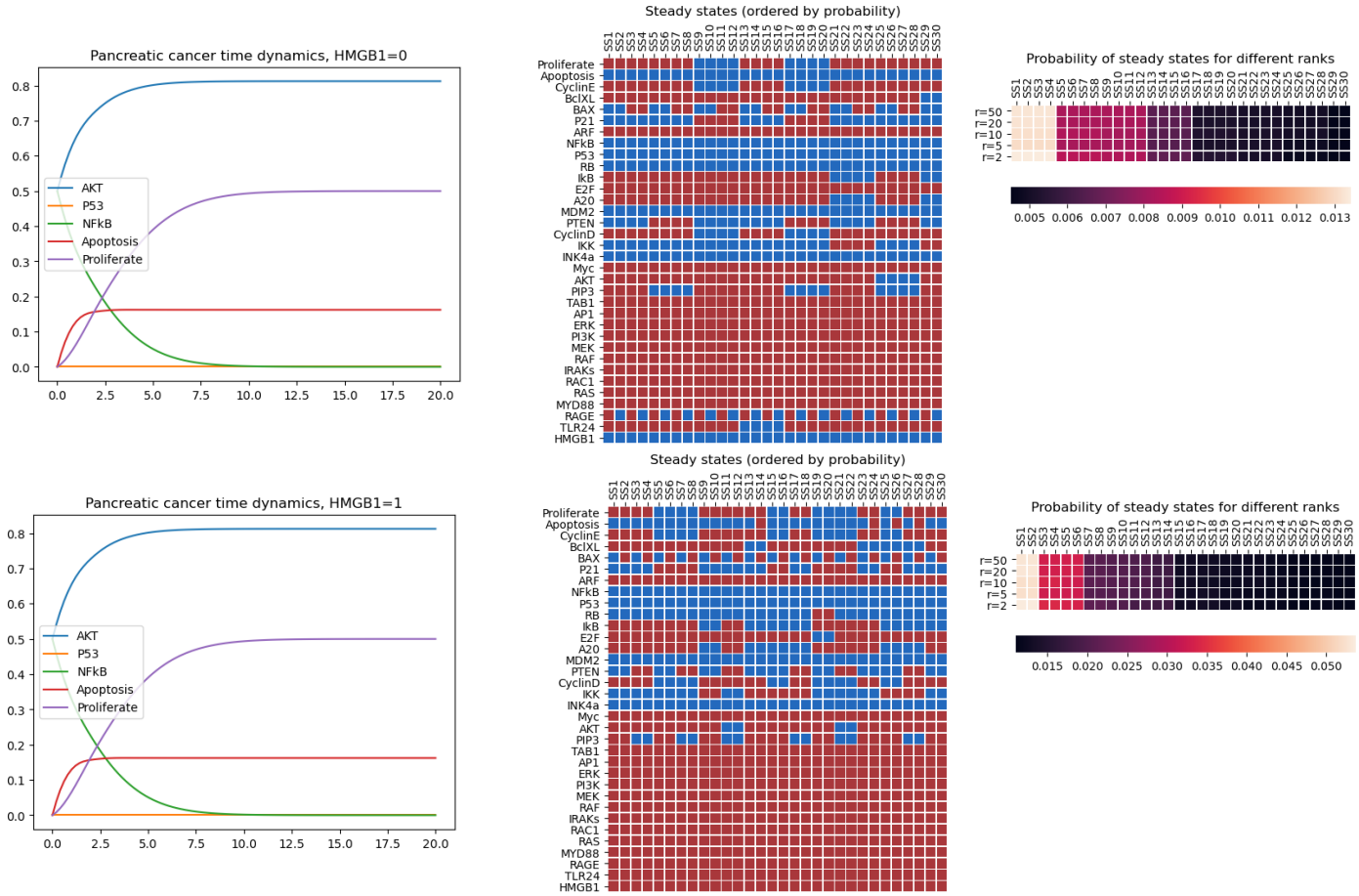

Figure 6: The dynamics of selected species for the pancreatic cancer model (left) and the steady sates (right) are shown for *HMGB1* switched off (top) and *HMGB1* switched on (bottom). Steady states are sorted according to their probability. Red represents that the species is on, blue represents that the species in this steady state is off. Note, that the dynamics for *Akt*, *p53*, *NFkB*, *Apoptosis*, and *Proliferate* are exactly the same. The number of steady states, however, are much reduced for *HMGB=1* (the first couple of significant steady states have a much higher probability than for *HMGB=0*).

6. A model for cell death regulation

To demonstrate the use of a low-rank approximation in another real-world problem, we now consider a Boolean network describing programmed cell death regulation taken from Mai et al.[32]. In this model, competing signals for death and growth and their interactions are modeled to explore the mechanisms that ultimately determine the fate of the cell. The model inputs are either a signal representing Tumor Necrosis Factor (TNF) or a growth factor (GF) meant to capture stimulus by a ligand such as Epidermal Growth Factor. The main observation in the original paper, is that setting *GF* to "on" is ineffective in reducing apoptosis in isolation, but can reduce apoptosis execution when combined with a *TNF* signal. This behavior can also be observed in the simulation results presented in Figure 7.

As shown in Table 2, direct evaluation of the chemical master equation with traditional methods would require 35 Terabytes of RAM and immense amounts of computer time. However, dynamic low-rank approximation of the problem with various ranks converts the problem from intractable to easily accessible with modest computation resources on a modern desktop or laptop computer.

Figure 7 shows the dynamics of *DNAdam* (which in this model is a proxy for programmed cell death) alongside five of the important proteins/genes. With *TNF* switched off the probability of *Apoptosis* is approximately 60%. This is independent whether *GF* is on or off. Switching *TNF* on increases the likelihood of apoptosis dramatically. In this setting switching on *GF* significantly suppresses apoptosis. This is consistent with previous results from the literature [32]. The mechanism of the network can be understood from Figure 8 as follows: *TNF* directly affects a cascade that includes various caspases that via *Cas3* as the last link in the chain lead to *Apoptosis*. If this cascade is active *AKT* (which is downstream of *GF*) can act as an inhibitor to *Cas9* breaking the direct link between *TNF* and *Cas3*. Thus, if the caspase cascade is active *GF* is able to suppress it. It has, however, no significant effect on the baseline *Apoptosis* rate (which is mediated via *p53*) and does not rely on the caspase cascade.

7. Conclusion

We have shown that our dynamical low-rank approach based on partitioning the network into biological relevant parts (as opposed to use single

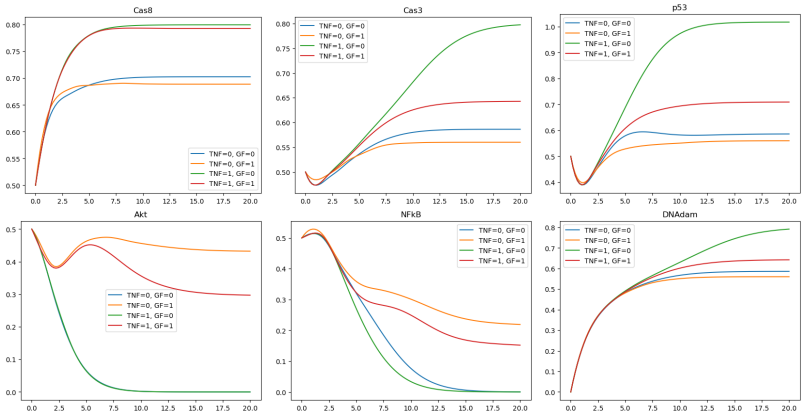

Figure 7: The dynamics of selected species for the apoptosis model is shown for the four different configurations of *TNF* and *GF* (i.e. *TNF*, *GF* set to 00, 01, 10, and 11).

species basis function) can achieve accurate results for the CME both in terms of dynamics and steady state results. Since a small rank (usually between $r = 5$ and $r = 15$) is sufficient, this approach results in a drastic reduction in memory and consequently allows us to solve large problems (such as the apoptosis model with 41 species for which a direct simulation would clearly be prohibitive) on a typical desktop workstation. We have also used our method to gain insight into the nature of synchronous vs. asynchronous updating, which is still heavily debated in the biochemistry community. Our work shows that for the studied pancreatic cancer model neither the steady states nor the dynamics under synchronous updating reflects well the observations made by solving the chemical master equation (i.e. asynchronous updating).

In order to enable the efficient simulation of even larger models with hundreds of species our goal is to use hierarchical tensor based dynamical low-rank approximations (such as the ones considered, e.g., in [30, 18, 31, 11]) in future work. Also in this case our viewpoint is that, using domain specific expertise, an appropriate hierarchy of subnetworks has to be identified in order to obtain accurate approximation at low rank (and thus at low computational cost).

Code availability & validation

The code used to run the simulation and generate the plots is available at `https://bitbucket.org/mprugger/low_rank_cme`.

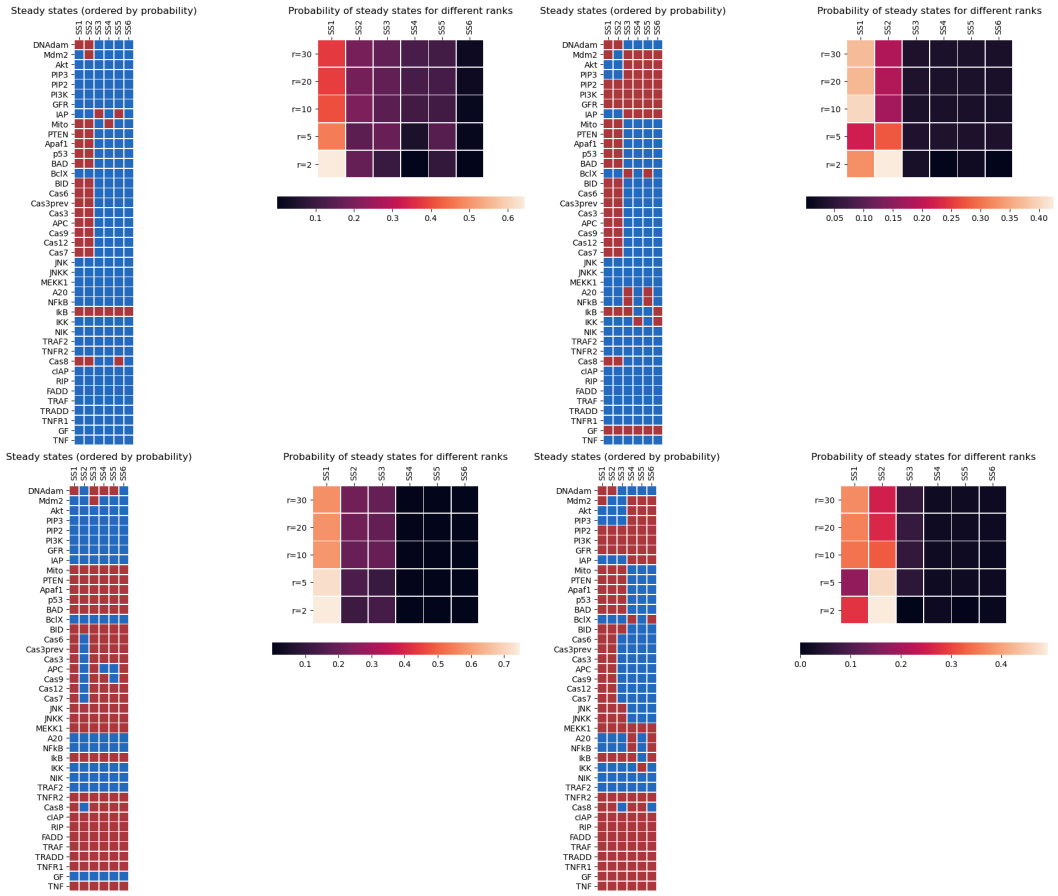

Figure 8: The steady states are depicted corresponding to the four different configurations of *TNF* and *GF* (i.e. *TNF*, *GF* set to 00, 01, 10, and 11; from top-left to bottom-right). The steady states are sorted according to their probability. Red represents that the species is on, blue represents that the species in this steady state is off.

We have compared the results obtained with the dynamical low-rank implementation with a direct simulation method for a range of smaller networks (including the mTOR pathway model considered in this paper). In addition, as part of the code, a number of automated unit tests are available.

Acknowledgments

This work was supported by National Science Foundation (NSF) CAREER award MCB 1942255 to CFL, National Institutes of Health (NIH) award 1U01CA215845 to CFL (M-PI).

References

[1] R. Albert and R. Robeva. Signaling networks: Asynchronous boolean models. In *Algebraic and discrete mathematical methods for modern biology*, pages 65–91. Elsevier, 2015.

[2] A. C. Babtie, T. E. Chan, and M. PH. Stumpf. Learning regulatory models for cell development from single cell transcriptomic data. *Current Opinion in Systems Biology*, 5:72–81, 2017.

[3] A.-L. Barabási. Scale-Free Networks: A Decade and Beyond. *Science*, 325(5939):412–413, 2009.

[4] S. Barman and Y.-K. Kwon. A boolean network inference from time-series gene expression data using a genetic algorithm. *Bioinformatics*, 34(17):i927–i933, 2018.

[5] J. Barwise, J. Etchemendy, G. Allwein, D. Barker-Plummer, and A. Liu. *Language, proof and logic*. CSLI publications, 2000.

[6] A. Benso, S. Di Carlo, G. Politano, A. Savino, and A. Vasciaveo. An extended gene protein/products boolean network model including post-transcriptional regulation. *Theoretical Biology and Medical Modelling*, 11(1):1–17, 2014.

[7] S. Bornholdt. Boolean network models of cellular regulation: prospects and limitations. *Journal of the Royal Society Interface*, 5(suppl_1):S85–S94, 2008.

[8] O. Brandman, J. E. Ferrell Jr, R. Li, and T. Meyer. Interlinked fast and slow positive feedback loops drive reliable cell decisions. *Science*, 310(5747):496–498, 2005.

[9] Gianluca Ceruti, Jonas Kusch, and Christian Lubich. A rank-adaptive robust integrator for dynamical low-rank approximation. *BIT Numerical Mathematics*, pages 1–26, 2022.

[10] Gianluca Ceruti and Christian Lubich. An unconventional robust integrator for dynamical low-rank approximation. *BIT Numerical Mathematics*, 62(1):23–44, 2022.

[11] Gianluca Ceruti, Christian Lubich, and Hanna Walach. Time integration of tree tensor networks. *SIAM Journal on Numerical Analysis*, 59(1):289–313, 2021.

[12] W. W. Chen, M. Niepel, and P. K. Sorger. Classic and contemporary approaches to modeling biochemical reactions. *Genes & development*, 24(17):1861–1875, 2010.

[13] M. A. Clarke and J. Fisher. Executable cancer models: successes and challenges. *Nature Reviews Cancer*, 20(6):343–354, 2020.

[14] Z. Ding, L. Einkemmer, and Q. Li. Dynamical low-rank integrator for the linear Boltzmann equation: error analysis in the diffusion limit. *SIAM Journal on Numerical Analysis*, 59(4):2254–2285, 2021.

[15] L. Einkemmer, J. Hu, and Y. Wang. An asymptotic-preserving dynamical low-rank method for the multi-scale multi-dimensional linear transport equation. *Journal of Computational Physics*, 439:110353, 2021.

[16] L. Einkemmer, J. Hu, and L. Ying. An Efficient Dynamical Low-Rank Algorithm for the Boltzmann–BGK Equation Close to the Compressible Viscous Flow Regime. *SIAM Journal on Scientific Computing*, 43(5), 2021.

[17] L. Einkemmer and I. Joseph. A mass, momentum, and energy conservative dynamical low-rank scheme for the Vlasov equation. *Journal of Computational Physics*, page 110495, 2021.

[18] L. Einkemmer and C. Lubich. A low-rank projector-splitting integrator for the Vlasov–Poisson equation. *SIAM Journal on Scientific Computing*, 40(5):B1330–B1360, 2018.

[19] L. Einkemmer, A. Ostermann, and C. Piazzola. A low-rank projector-splitting integrator for the Vlasov–Maxwell equations with divergence correction. *Journal of Computational Physics*, 403:109063, 2020.

[20] S. Gao, C. Xiang, C. Sun, K. Qin, and T. H. Lee. Efficient boolean modeling of gene regulatory networks via random forest based feature selection and best-fit extension. In *2018 IEEE 14th International Conference on Control and Automation (ICCA)*, pages 1076–1081. IEEE, 2018.

[21] A. Garg, A. Di Cara, I. Xenarios, L. Mendoza, and G. De Micheli. Synchronous versus asynchronous modeling of gene regulatory networks. *Bioinformatics*, 24(17):1917–1925, 2008.

[22] D. T. Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of computational physics*, 22(4):403–434, 1976.

[23] D. T. Gillespie, A. Hellander, and L. R. Petzold. Perspective: Stochastic algorithms for chemical kinetics. *The Journal of chemical physics*, 138(17):05B201_1, 2013.

[24] H. Gong, Q. Wang, P. Zuliani, J. R. Faeder, M. Lotze, and E. Clarke. Symbolic Model Checking of Signaling Pathways in Pancreatic Cancer. In *BICoB*, page 245, 2011.

[25] S. Grima and S. Schnell. Modelling reaction kinetics inside cells. *Essays in Biochemistry*, 45:41–56, 2008.

[26] L. A. Harris and P. Clancy. A "partitioned leaping" approach for multiscale modeling of chemical reaction dynamics. *The Journal of chemical physics*, 125(14):144107, 2006.

[27] T. Jahnke and W. Huisinga. A dynamical low-rank approach to the chemical master equation. *Bulletin of mathematical biology*, 70(8):2283, 2008.

[28] K. Klemm and S. Bornholdt. Stable and unstable attractors in boolean networks. *Physical Review E*, 72(5):055101, 2005.

[29] C. Lubich and I.V. Oseledets. A projector-splitting integrator for dynamical low-rank approximation. *BIT Numer. Math.*, 54(1):171–188, 2014.

[30] C. Lubich, T. Rohwedder, R. Schneider, and B. Vandereycken. Dynamical approximation by hierarchical Tucker and tensor-train tensors. *SIAM Journal on Matrix Analysis and Applications*, 34(2):470–494, 2013.

[31] Christian Lubich, Bart Vandereycken, and Hanna Walach. Time integration of rank-constrained tucker tensors. *SIAM Journal on Numerical Analysis*, 56(3):1273–1290, 2018.

[32] Z. Mai and H. Liu. Boolean network-based analysis of the apoptosis network: irreversible apoptosis and stable surviving. *Journal of theoretical biology*, 259(4):760–769, 2009.

[33] S. Martin, Z. Zhang, A. Martino, and J.-L. Faulon. Boolean dynamics of genetic regulatory networks inferred from microarray time series data. *Bioinformatics*, 23(7):866–874, 2007.

[34] H.-D. Meyer, U. Manthe, and L. S. Cederbaum. The multi-configurational time-dependent Hartree approach. *Chemical Physics Letters*, 165(1):73–78, 1990.

[35] B. Munsky and M. Khammash. The finite state projection algorithm for the solution of the chemical master equation. *The Journal of chemical physics*, 124(4):044104, 2006.

[36] M. Niepel, S. L. Spencer, and P. K. Sorger. Non-genetic cell-to-cell variability and the consequences for pharmacology. *Current opinion in chemical biology*, 13(5-6):556–561, 2009.

[37] L. Null and J. Lobur. *Essentials of Computer Organization and Architecture*. Jones & Bartlett Learning, 2018.

[38] P. Paszek, S. Ryan, L. Ashall, K. Sillitoe, C. V. Harper, D. G. Spiller, D. A. Rand, and M. RH. White. Population robustness arising from cellular heterogeneity. *Proceedings of the National Academy of Sciences*, 107(25):11644–11649, 2010.

[39] L. Paulevé, J. Kolčák, T. Chatain, and S. Haar. Reconciling qualitative, abstract, and scalable modeling of biological networks. *Nature communications*, 11(1):1–7, 2020.

[40] Z. Peng, R. G. McClarren, and M. Frank. A low-rank method for two-dimensional time-dependent radiation transport calculations. *Journal of Computational Physics*, 421:109735, 2020.

[41] M. Prugger, L. Einkemmer, S. P. Beik, P. T. Wasdin, L. A. Harris, and C. F. Lopez. Unsupervised logic-based mechanism inference for network-driven biological processes. *PLOS Computational Biology*, 17(6):e1009035, 2021.

[42] A. Ribeiro, R. Zhu, and S. A. Kauffman. A general modeling strategy for gene regulatory networks with stochastic dynamics. *Journal of computational Biology*, 13(9):1630–1639, 2006.

[43] A. Saadatpour, I. Albert, and R. Albert. Attractor analysis of asynchronous boolean models of signal transduction networks. *Journal of theoretical biology*, 266(4):641–656, 2010.

[44] J. D. Schwab, S. D. Kühlwein, N. Ikonomi, M. Kühl, and H. A. Kestler. Concepts in boolean network modeling: What do they all mean? *Computational and structural biotechnology journal*, 18:571–582, 2020.

[45] N. Shi, Z. Zhu, K. Tang, D. Parker, and S. He. Aten: And/or tree ensemble for inferring accurate boolean network topology and dynamics. *Bioinformatics*, 36(2):578–585, 2020.

[46] E. M. Shockley, J. A. Vrugt, and C. F. Lopez. PyDREAM: high-dimensional parameter inference for biological models in python. *Bioinformatics*, 34(4):695–697, 2018.

[47] G. Stoll, E. Viara, E. Barillot, and L. Calzone. Continuous time boolean modeling for biological signaling: application of gillespie algorithm. *BMC systems biology*, 6(1):1–18, 2012.

[48] M. K. Tonn, P. Thomas, M. Barahona, and D. A. Oyarzún. Stochastic modelling reveals mechanisms of metabolic heterogeneity. *Communications biology*, 2(1):1–9, 2019.

[49] J. J. Tyson, K. C. Chen, and B. Novak. Network dynamics and cell physiology. *Nature reviews Molecular cell biology*, 2(12):908–916, 2001.

[50] J. J. Tyson, K. C. Chen, and B. Novak. Sniffers, buzzers, toggles and blinkers: dynamics of regulatory and signaling pathways in the cell. *Current opinion in cell biology*, 15(2):221–231, 2003.

[51] M. L. Wynn, N. Consul, S. D. Merajver, and S. Schnell. Logic-based models in systems biology: a predictive and parameter-free network analysis method. *Integrative biology*, 4(11):1323–1337, 2012.

[52] A. Yachie-Kinoshita, K. Onishi, J. Ostblom, M. A. Langley, E. Posfai, J. Rossant, and P. W. Zandstra. Modeling signaling-dependent pluripotency with Boolean logic to predict cell fate transitions. *Molecular systems biology*, 14(1):e7952, 2018.

[53] J. GT. Zanudo and R. Albert. Cell fate reprogramming by control of intracellular network dynamics. *PLoS computational biology*, 11(4):e1004193, 2015.

[54] J. GT. Zañudo, S. N. Steinway, and R. Albert. Discrete dynamic network modeling of oncogenic signaling: Mechanistic insights for personalized treatment of cancer. *Current Opinion in Systems Biology*, 9:1–10, 2018.

Appendix A. Translate a Boolean rule into a System of ODEs

Let us first take a look into how a system of Boolean rules can be translated into a matrix-vector multiplication. For this demonstrative purpose, we consider the small system of three species A, B, and C that are updated according to the following ruleset

$$\begin{aligned} (\text{rule } 1) : A &\leftarrow B \vee C \\ (\text{rule } 2) : B &\leftarrow \overline{A} \wedge B \\ (\text{rule } 3) : C &\leftarrow A \wedge \overline{C}. \end{aligned}$$

Figures A.9, A.10, A.11 demonstrate the corresponding truth tables, as well as their translation into a matrix that multiplied by the vector $P = (000, 001, 010, 011, 100, 101, 110, 111)^T$ recreate the rules as seen above. To update P according to all the rules, the matrices for each rule are added up and we get the following system of ODEs

$A \leftarrow B \vee C :$ $\xrightarrow{\text{Truthtable}}$

B	C	A
0	0	0
0	1	1
1	0	1
1	1	1

$$
\begin{array}{rcl}
A^n B^n C^n & \rightarrow & A^{n+1} B^{n+1} C^{n+1} \\
000 & \rightarrow & 000 \\
001 & \rightarrow & 101 \\
010 & \rightarrow & 110 \\
011 & \rightarrow & 111 \\
100 & \rightarrow & 000 \\
101 & \rightarrow & 101 \\
110 & \rightarrow & 110 \\
111 & \rightarrow & 111
\end{array}
\qquad
\begin{array}{rcl}
\partial_t P(t,000) & = & 0 + P(t,100) \\
\partial_t P(t,001) & = & -P(t,001) \\
\partial_t P(t,010) & = & -P(t,010) \\
\partial_t P(t,011) & = & -P(t,011) \\
\partial_t P(t,100) & = & -P(t,100) \\
\partial_t P(t,101) & = & 0 + P(t,001) \\
\partial_t P(t,110) & = & 0 + P(t,010) \\
\partial_t P(t,111) & = & 0 + P(t,011)
\end{array}
$$

$$
\begin{array}{rl}
 & \begin{array}{cccccccc} 000 & 001 & 010 & 011 & 100 & 101 & 110 & 111 \end{array} \\
\begin{array}{rl} \text{gets a state from 100} & 000 \\ \text{looses a state to 101} & 001 \\ \text{looses a state to 110} & 010 \\ \text{looses a state to 111} & 011 \\ \text{looses a state to 000} & 100 \\ \text{gets a state from 001} & 101 \\ \text{gets a state from 010} & 110 \\ \text{gets a state from 011} & 111 \end{array} &
\begin{bmatrix}
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0
\end{bmatrix} =: A_1
\end{array}
$$

Figure A.9: Depiction of the formulation of the rule $A \leftarrow B \vee C$ into a matrix. Note, that the negative state change from, e.g., $\partial_t P(t, 001) = -P(t, 001)$ assures that $\partial_t \left(\sum_x P(T, x) \right) = 0$ and thus the probability density function always stays normalized (conservation of mass).

$B \leftarrow \overline{A} \wedge B :$ $\xrightarrow{\text{Truthtable}}$

A	C	B
0	0	0
0	1	1
1	0	0
1	1	0

$$
\begin{array}{rcl}
A^n B^n C^n & \rightarrow & A^{n+1} B^{n+1} C^{n+1} \\
000 & \rightarrow & 000 \\
001 & \rightarrow & 011 \\
010 & \rightarrow & 000 \\
011 & \rightarrow & 011 \\
100 & \rightarrow & 100 \\
101 & \rightarrow & 101 \\
110 & \rightarrow & 100 \\
111 & \rightarrow & 101
\end{array}
\qquad
\begin{array}{rcl}
\partial_t P(t,000) & = & 0 + P(t,010) \\
\partial_t P(t,001) & = & -P(t,001) \\
\partial_t P(t,010) & = & -P(t,010) \\
\partial_t P(t,011) & = & 0 + P(t,001) \\
\partial_t P(t,100) & = & 0 + P(t,110) \\
\partial_t P(t,101) & = & 0 + P(t,111) \\
\partial_t P(t,110) & = & -P(t,110) \\
\partial_t P(t,111) & = & -P(t,111)
\end{array}
$$

$$
\begin{array}{rl}
 & \begin{array}{cccccccc} 000 & 001 & 010 & 011 & 100 & 101 & 110 & 111 \end{array} \\
\begin{array}{rl} \text{gets a state from 010} & 000 \\ \text{looses a state to 011} & 001 \\ \text{looses a state to 000} & 010 \\ \text{gets a state from 001} & 011 \\ \text{gets a state from 110} & 100 \\ \text{gets a state from 111} & 101 \\ \text{looses a state to 100} & 110 \\ \text{looses a state to 101} & 111 \end{array} &
\begin{bmatrix}
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1
\end{bmatrix} =: A_2
\end{array}
$$

Figure A.10: Depiction of the formulation of the rule $B \leftarrow \overline{A} \wedge B$ into a matrix.

$C \leftarrow A \wedge \overline{C}:$ $\xrightarrow{\text{Truthtable}}$

A	C	C
0	0	0
0	1	0
1	0	1
1	1	0

$$
\begin{array}{rcl}
A^n B^n C^n & \rightarrow & A^{n+1} B^{n+1} C^{n+1} \\
000 & \rightarrow & 000 \\
001 & \rightarrow & 000 \\
010 & \rightarrow & 010 \\
011 & \rightarrow & 010 \\
100 & \rightarrow & 101 \\
101 & \rightarrow & 101 \\
110 & \rightarrow & 110 \\
111 & \rightarrow & 110
\end{array}
\qquad
\begin{array}{rcl}
\partial_t P(t,000) & = & 0 + P(t,001) \\
\partial_t P(t,001) & = & -P(t,001) \\
\partial_t P(t,010) & = & 0 + P(t,011) \\
\partial_t P(t,011) & = & -P(t,011) \\
\partial_t P(t,100) & = & -P(t,100) \\
\partial_t P(t,101) & = & 0 + P(t,100) \\
\partial_t P(t,110) & = & 0 + P(t,111) \\
\partial_t P(t,111) & = & -P(t,111)
\end{array}
$$

$$
\begin{array}{rc}
 & \\
\text{gets a state from 001} & 000 \\
\text{looses a state to 000} & 001 \\
\text{gets a state from 011} & 010 \\
\text{looses a state to 010} & 011 \\
\text{looses a state to 101} & 100 \\
\text{gets a state from 100} & 101 \\
\text{gets a state from 111} & 110 \\
\text{looses a state to 110} & 111
\end{array}
\begin{array}{c}
\begin{array}{cccccccc} 000 & 001 & 010 & 011 & 100 & 101 & 110 & 111 \end{array} \\
\begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1
\end{bmatrix}
\end{array}
=: A_3
$$

Figure A.11: Depiction of the formulation of the rule $C \leftarrow A \wedge \overline{C}$ into a matrix.

$n = 22$	
`randvec = np.random.rand(2**n);` `randvec = randvec/np.sum(randvec)`	random initial setup
`randvec = np.zeros(2**n);` `randvec[randint(0,2**n-1)]=1.0`	single initial setup
`randvec = np.zeros(2**n);` `randvec[387]=1.0`	single initial setup for comparison purposes
`randvec = np.ones(2**n)/(2**n)`	uniform initial setup

Table B.3: Initial conditions used in the analysis of the $mTOR$ network.

$$\partial_t P = A \cdot P = (A_1 + A_2 + A_3)P = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & -3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -2 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -2 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -2 \end{bmatrix} \cdot \begin{bmatrix} 000 \\ 001 \\ 010 \\ 011 \\ 100 \\ 101 \\ 110 \\ 111 \end{bmatrix}.$$

Which is the formulation of the CME for this particular ruleset. Note, that each column/row of this matrix corresponds to a particular state of each species, e.g, the state $A = 0$, $B = 1$, $C = 1$ corresponds to the fourth column/row of the matrix A, (011). We therefore can see, that for a number of d species in the system, the formulation of the CME becomes a matrix of $2^d \times 2^d$ entries, and thus suffers from the curse of dimensionality.

Appendix B. Detailed error analysis

We now perform a more detailed error analysis of the dynamical low-rank approximation for different ranks and initial values. We will limit ourselves to the blue partitioning and the initial values used are summarized in Table B.3. Since the mTOR network is still small enough, we can use a reference solution obtained with a direct simulation algorithm (i.e. without performing a low-rank approximation).

The results of the comparison are illustrated in Figure B.12. For all initial values chosen the obtained results further highlight the efficiency of the low-rank approximation as in all cases a small rank is sufficient in order to obtain

	r2	r5	r10	r20	r50
blue	0.73	0.93	1.02	1.00	1.00
green	0.23	0.41	0.84	0.97	1.00
purple	0.52	0.85	0.98	1.00	1.00
red	0.70	0.96	0.99	1.00	1.00
yellow	0.10	0.23	0.27	0.68	0.89

Table B.4: Steady state mass for each simulation for the network with 22 nodes.

accurate results. The spike in the single initial condition can be attributed to the large number of intermediate states that have to be resolved at that point in time.

Figure B.13 depicts the state explosion that renders finite state projection infeasible for Boolean networks.

In Table B.4, we display the stability of the method by the mass conservation for the steady states for our five different partitions. Since the behavior of the mass is similar to the error regarding the partitioning, we consider it a good measure for the performance of partitions: Ranks 10 to 50 can not really distinguish between the blue, purple and red partition. However, while blue and red behave similar for the lower ranks, purple looses much more mass in the matter. The green partition is the least stable regarding the partitions that have not been drawn randomly, similar to Figure 2, in the main paper, where it displays the largest error for the high ranks. Unsurprisingly, the random partition performs the worst in already having a significant loss of mass for the highest rank $r = 50$ and loosing the stability with the rank completely.

Appendix C. Partitionings for the pancreatic cancer network and the apoptosis network

Appendix C.1. Network partitioning for the pancreatic cancer network

The pancreatic cancer network is taken from [24] and consists of 34 species. This network is used to investigate the effect of asynchronous updating versus synchronous updating. The distribution of the species we chose is shown in Table C.5 and Figure C.14.

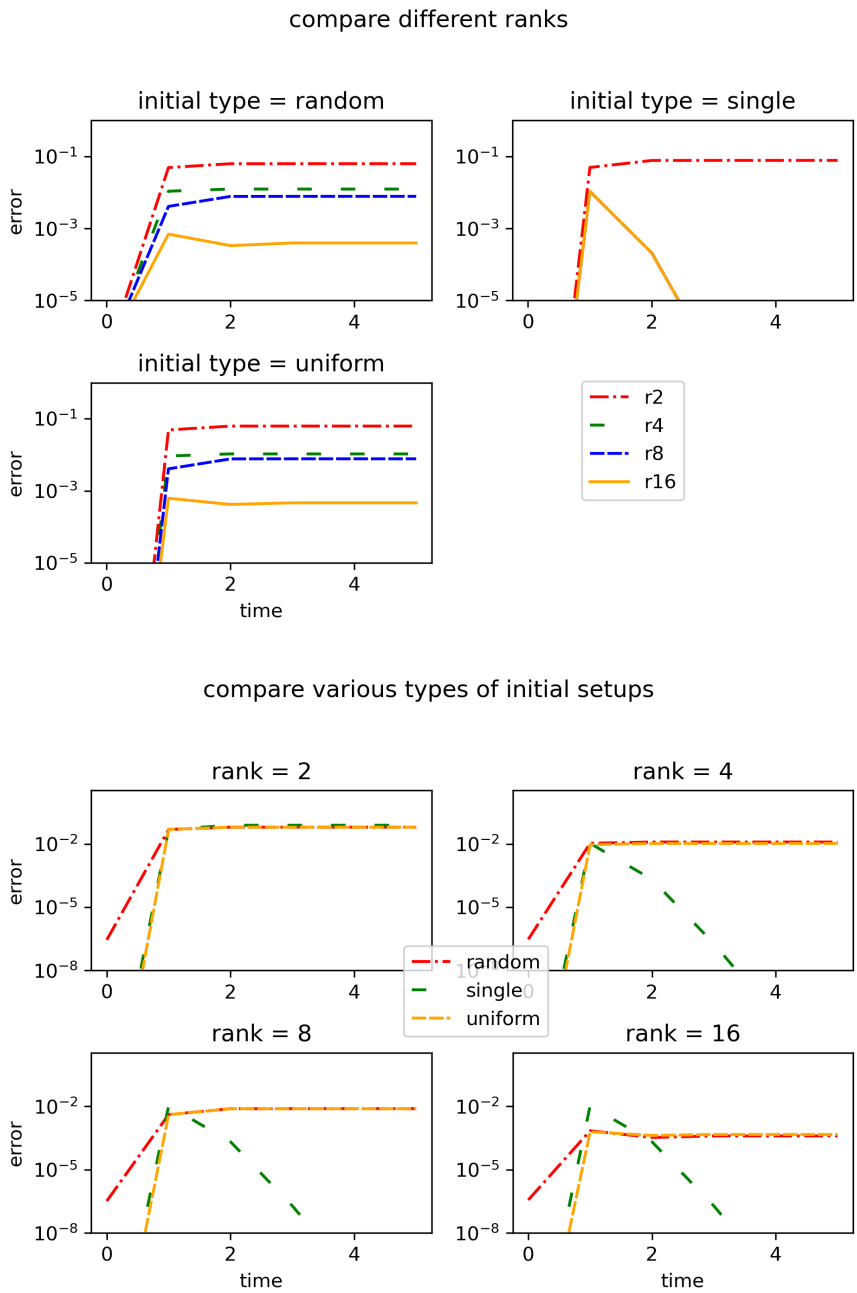

Figure B.12: Comparison of the different setups to the exact solution according to rank and initial conditions. For the single initial setup, we picked the 387th initial vector.

	partition 1	partition 2
species	MYD88	IKK
	TLR2/4	A20
	HMGB1	ARF
	RAGE	IkB
	IRAKS	MDM2
	MEK	PTEN
	RAF	E2F
	RAS	CyclinD
	RAC1	NFkB
	ERK	P53
	TAB1	P21
	AKT	RB
	PI3K	Bcl-XL
	AP1	BAX
	Myc	CyclinE
	PIP3	Apoptosis
	INK4a	Proliferate
partition interactions	10	

Table C.5: Distribution of the 34 species.

	partition 1	partition 2
species	TNF	Cas12
	GF	Cas9
	TNFR1	APC
	TRADD	Cas3
	TRAF	Cas3prev
	FADD	Cas6
	RIP	BID
	cIAP	BclX
	Cas8	BAD
	TNFR2	p53
	TRAF2	Apaf1
	NIK	PTEN
	IKK	Mito
	IkB	IAP
	NFkB	GFR
	A20	PI3K
	MEKK1	PIP2
	JNKK	PIP3
	JNK	Akt
	Cas7	Mdm2
		DNAdam
partition interactions	14	

Table C.6: Distribution of the 41 species.

Appendix C.2. Network partitioning for the apoptosis network

We chose the Boolean network with 41 nodes describing Apoptosis taken from [32]. As pointed out, we would need at least 35 Terabyte of memory to compute the exact solution. Note that for our reference solution, we did not have enough memory on our system to simulate rank 50. We therefore settled for a reference solution of rank $r = 30$. Since we wanted to demonstrate how to deal with an odd number of species in our case, we also did not include the Apoptosis event, which is turned on, if the DNA Damage Event kept on in 20 continuous steps. We display the partitioning we used for our simulations in Table C.6.

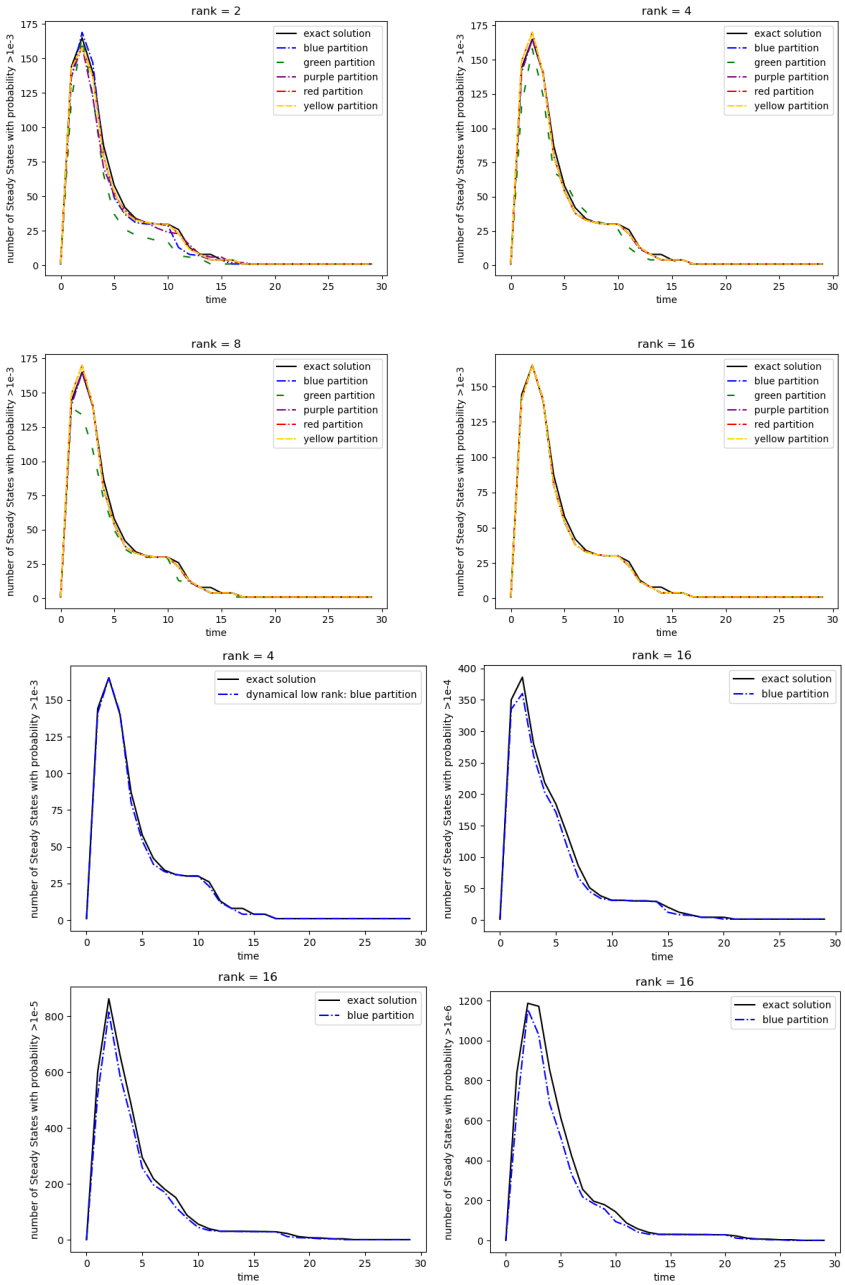

Figure B.13: State explosion at the beginning of the mTOR network simulation for the single initial condition setup. On top, the different ranks are resolved according to the different ranks. On the bottom, for the blue partition, and rank 16, the sensitivity is lowered.

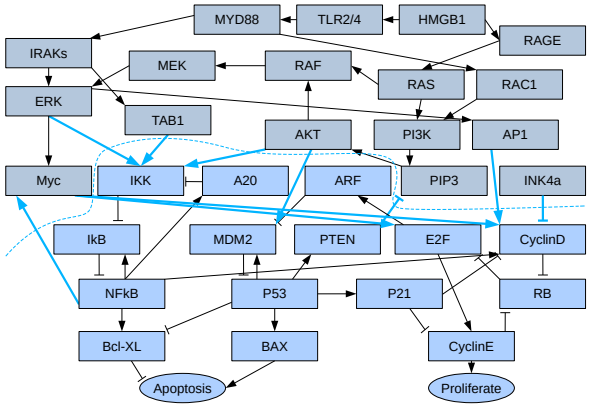

Figure C.14: Network partitionings for pancreatic cancer taken from [24].