Efficient Computation of All Distinct Realization Structures of Kinetic Systems

Zoltan A. Tuza [*] Bernadett Ács [**] Gábor Szederkényi [**]
Frank Allgöwer [*]

[*] *Institute for Systems Theory and Automatic Control, University of Stuttgart, Pfaffenwaldring 9, 70550 Stuttgart, Germany, Email: {zoltan.tuza, frank.allgower}@ist.uni-stuttgart.de*
[**] *Pázmány Péter Catholic University Faculty of Information Technology and Bionics, Práter u. 50/a, 1083 Budapest, Hungary, Email: {acs.bernadett, szederkenyi}@itk.ppke.hu*

Abstract: Structural non-uniqueness of (bio)chemical reaction networks realizing a given kinetic dynamics has been known for a long time, but it is often overlooked in practice. However, without appropriate prior information, this phenomenon seriously hinders the successful identification of biochemical models. Recently an algorithm with guaranteed polynomial time complexity between iterations has been developed to compute all distinct reaction graph structures corresponding to a given dynamics. This paper presents an improved version of this algorithm that is suitable to take the advantage of a multiprocessor environment. The computed structures are collected in a task queue, and two server processes coordinate the operation of the set of workers. The implementation is briefly described and the performance of the approach is illustrated on computational examples taken from the literature.

Keywords: Chemical reaction networks, Optimization, Parallel computation, Polynomial models

1. INTRODUCTION

Kinetic systems and their realizing reaction networks are popular tools for modeling (bio)chemical reactions in chemical and process engineering as well as in systems biology. This is mainly due to the useful relations between the structure of reaction networks and the properties of the differential equations describing the dynamics of the chemical reaction network. It is known, however, that the correspondence between the network structure and the dynamics is generally not one-to-one. This means that there might exist many different network structures (called realizations) translating to exactly the same set of differential equations, even if the set of complexes is fixed; see in Horn and Jackson (1972). It is also well-known that the inference of biochemical reaction networks is a challenging task due to the frequent lack of structural or practical identifiability, see more in Chis et al. (2011). Naturally, the possible structural non-uniqueness has a substantial impact on the identifiability of kinetic systems (see e.g. Craciun and Pantea (2008); Szederkényi et al. (2011)). Clearly, the different realizations share the same dynamical properties: e.g. a stability condition obtained using a certain reaction graph proves this property for the corresponding dynamical system itself. To exploit such properties several optimization based algorithms have been developed to determine realizations with preferred properties (cf. Szederkényi (2010b,a)).

In this paper we use two main notions to relate the kinetic dynamical system and the corresponding graph structures. The reaction networks yielding exactly the same kinetic differential equations are called *dynamically equivalent*. Johnston and Siegel (2011) introduced a positive linear diagonal transformation between the solutions (trajectories) of kinetic systems, in this case kinetic systems are called *linearly conjugate*. It is easy to see that dynamical equivalence is a special case of linear conjugacy (cf. Craciun and Pantea (2008), Johnston et al. (2012b)).

The computation of all structurally different graph structures is generally a combinatorial problem, but there are special properties we can exploit to significantly reduce the search space. First, there exist an upper and a lower bound for the possible number of reactions in a reaction network realizing a kinetic system. Szederkényi (2010a) reported a way to compute these bounds which are the so-called dense and sparse realizations, respectively. Second, it was proved that the dense realization is a unique super-structure for a given dynamics and contains all other possible realizations as proper sub-graphs (see e.g. Johnston et al. (2012a)).

Beyond the possibility of an exhaustive search, knowing all different realizations of a kinetic system enables us to study such properties of the whole solution set that we are not yet able to compute directly. An example for that is the enumeration of all structures with a given deficiency. Deficiency is a realization property but it may have immediate consequences on the stability and uniqueness of equilibria for kinetic systems as it was introduced in Feinberg (1987) and Feinberg (1988).

Tuza et al. (2013) reported the first solution to enumerate all sparse realizations of a kinetic system. Recently, Ács et al. (2016) developed the first provably correct algorithm with advantageous time complexity properties for computing all distinct reaction graph structures. Basically, this algorithm builds a hierarchical tree structure which contains all realizations as vertices of a tree. In the root we have the dense realization (the upper bound) and below that in each horizontal level we can find all the realizations with a smaller number of reactions, than one level above. Finally, at the bottom level we can find all realizations with the minimum number of reactions (i.e. the sparse realizations).

Based on the above, the aim of this paper is to present and evaluate an improved, computationally more efficient implementation of the method presented in Ács et al. (2016), and to illustrate the possible degree of structural non-uniqueness (and consequently, that of non-identifiability) using examples taken from the literature including models related to systems biology.

The paper is organized as follows: the next section briefly introduces the necessary mathematical background for the implementation. The third section presents the improved algorithm for computing all different graph structures and outlines the proof of correctness. Section 4 details the implementation. The computational results are presented in Section 5, while Section 6 concludes the paper.

2. REPRESENTATIONS OF KINETIC SYSTEMS

We consider reaction networks as a general system class representing a wide class of nonlinear dynamical systems with non-negative states (see e.g. in Chellaboina et al. (2009)). Throughout the paper we follow the notations of Ács et al. (2016).

2.1 Algebraic characterization

A reaction network can be characterized by three sets:

(1) a set of **species**: $\mathcal{S} = \{X_i \mid i \in \{1, \dots, n\}\}$
(2) a set of **complexes**: $\mathcal{C} = \{C_j \mid j \in \{1, \dots, m\}\}$, where
$$C_j = \sum_{i=1}^{n} \alpha_{ji} X_i \qquad j \in \{1, \dots, m\}$$
$$\alpha_{ji} \in \mathbb{N}_0 \qquad j \in \{1, \dots, m\},\ i \in \{1, \dots, n\},$$
(3) and a set of **reactions**: $\mathcal{R} \subseteq \{(C_i, C_j) \mid C_i, C_j \in \mathcal{C}\}$ Each ordered pair (C_i, C_j) has a reaction rate coefficient $k_{ij} \in \mathbb{R}_+^n$ so that the corresponding reaction $C_i \to C_j$ takes place if and only if $k_{ij} > 0$.

The structure of the reaction network can be characterized by special matrices: the complex composition matrix $Y \in \mathbb{N}_0^{n \times m}$ describes the complexes as follows
$$[Y]_{ij} = \alpha_{ji} \qquad i \in \{1, \dots, n\},\ j \in \{1, \dots, m\},$$
and the set of reactions is encoded by the Kirchhoff matrix $A_k \in \mathbb{R}^{m \times m}$ as
$$[A_k]_{ij} = \begin{cases} k_{ji} & \text{if } i \neq j \\ -\sum\limits_{l=1, l \neq i}^{m} k_{il} & \text{if } i = j. \end{cases}$$

2.2 Dynamical description

If mass action kinetics is assumed and the concentrations of the species depending on time are represented by the function $x : \mathbb{R} \to \mathbb{R}_+^n$, the time evolution of the model can be characterized by a polynomial dynamical system:
$$\dot{x} = Y \cdot A_k \cdot \psi(x) \tag{1}$$
where $\psi : \mathbb{R}_+^n \to \mathbb{R}_+^m$ is a monomial-type vector-mapping defined by
$$\psi_j(x) = \prod_{i=1}^{n} x_i^{Y_{ij}}, \quad j = 1, \dots, m. \tag{2}$$

It is visible from (1) and (2) that the ODEs of a reaction network can be characterized by the matrix pair Y and A_k.

Obviously, (1) can be written as a polynomial system of the form
$$\dot{x} = M \cdot \varphi(x) \tag{3}$$
where $x : \mathbb{R} \to \mathbb{R}_+^n$ is a function, $M \in \mathbb{R}^{n \times p}$ a coefficient matrix and $\varphi : \mathbb{R}_+^n \to \mathbb{R}_+^p$ a monomial function. The polynomial system (3) is called a **kinetic system** if there exists a reaction network governed by the same dynamics, i.e. the following equation is fulfilled:
$$M \cdot \varphi(x) = Y \cdot A_k \cdot \psi(x), \ \ \forall x. \tag{4}$$
In this case the reaction network represented by the matrices Y and A_k is called a **dynamically equivalent realization** of the kinetic system (3). We remark that the monomial functions φ and ψ are generally not identical, since the set of complexes determining the monomials is not fixed. However, the set of complexes can be suitably complemented; we can assume without the loss of generality that $\varphi = \psi$ holds (see e.g. Ács et al. (2016) for details).

The notion of dynamical equivalence can be generalized to the case when the state vector is subject to a positive linear diagonal state transformation, performed by a positive definite diagonal matrix $T \in \mathbb{R}^{n \times n}$ as follows: $\bar{x} = T^{-1} \cdot x$. According to the definition in Johnston et al. (2012a): a reaction network defined by matrices Y and A_k' is called a **linearly conjugate** realization of the kinetic system (3) if the following equations hold:
$$Y \cdot A_k = T^{-1} \cdot M \tag{5}$$
$$A_k' = A_k \cdot \Phi_T, \tag{6}$$
where $\Phi_T \in \mathbb{R}^{m \times m}$ is a positive definite diagonal matrix so that $[\Phi_T]_{ii} = \psi_i(T \cdot \mathbf{1})$ for $i \in \{1, \dots, n\}$, and $\mathbf{1} \in \mathbb{R}^n$ is a vector with all coordinates equal to 1.

2.3 Graph representation

The above notions give reaction networks a natural representation as weighted directed graphs $G(V, E)$ called **Feinberg-Horn-Jackson graphs**, or simply **reaction graphs** as follows:

- vertices: $V(G) = \mathcal{C}$
- edges: $E(G) = \mathcal{R}$
- edge weights: $w(C_i, C_j) = k_{ij}$

It can be seen that the reaction graph is encoded by the matrix A_k, but since we want to determine only the structures of the realizations from now on we consider reaction graphs as unweighted directed graphs.

2.4 Distinguished reaction graph structures

The **dense realization** of a kinetic system—where the maximum number of reactions take place—has a special property that guarantees the applicability of the algorithm presented in this paper. Ács et al. (2015) proves that among linearly conjugate realizations fulfilling a finite set of linear constraints on the parameters there is a realization determining a super-structure. This super-structure is the unweighted reaction graph of the dense realization which contains the reaction graph structures of all possible linearly conjugate realizations of the model with linear constraints as subgraphs.

The **sparse realizations** of a kinetic system are such realizations where the minimum number of reactions take place. In general there may exist several sparse realizations of a kinetic system.

2.5 Computing graph structures based on optimization

Johnston et al. (2012a) introduced a method to compute linearly conjugate realizations by linear programming. For this method we assume that matrices Y and M are given and the matrices A_k and T^{-1} are the decision variables. The linear constraints characterizing feasible linearly conjugate realizations are the following:

$$
\begin{aligned}
&T^{-1} \cdot M = Y \cdot A_k \\
&[A_k]_{ij} \geq 0 \quad i,j \in \{1,\dots,m\},\ i \neq j \qquad (7)\\
&[A_k]_{ii} = -\sum_{\substack{j=1\\ j\neq i}}^{m} [A_k]_{ji} \quad i \in \{1,\dots,m\} \\
&[T^{-1}]_{ii} > 0 \quad i \in \{1,\dots,n\}.
\end{aligned}
$$

A usual restriction in the computation steps is that a set $\mathcal{H} \subset \mathcal{R}$ of reactions has to be excluded from the network, which can also be written as a linear constraint:

$$[A_k]_{ji} = 0 \qquad (C_i, C_j) \in \mathcal{H}. \qquad (8)$$

To execute the algorithm presented in the next section it is necessary to compute constrained dense linearly conjugate realizations. These can be determined efficiently in polynomial time using linear programming as it was presented in Ács et al. (2015). For the computations described in this paper we transformed the constraints (7)-(8) into a constant size linear program in pure inequality form. This is the most suitable form for the applied solver, and with the repeatedly modified edge-exclusions in (8) the optimization problem can be solved in minimal time.

2.6 Example

In this section we introduce a reaction network originally presented in Császár et al. (1981), and throughout the paper this example will be referred to as $A1$. The main purpose of using this example is to compare the operation of the improved method with that of Ács et al. (2016). The reaction network is characterized by the following sets:

$$
\begin{aligned}
\mathcal{S} &= \{X_1, X_2\} \\
\mathcal{C} &= \{C_1 = 0, C_2 = X_1, C_3 = X_2, C_4 = 2X_1 \\
&\quad C_5 = 2X_1 + X_2, C_6 = 3X_1\} \\
\mathcal{R} &= \{(C_1,C_3),(C_3,C_2),(C_2,C_1),(C_4,C_3),(C_5,C_6)\}
\end{aligned}
$$

reaction rate coefficients: $k_{13} = k_1, k_{32} = k_3, k_{21} = k_2, k_{43} = k_4, k_{56} = k_5$.

The following matrices encode the network structure:

$$Y = \begin{bmatrix} 0 & 1 & 0 & 2 & 2 & 3 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

$$A_k = \begin{bmatrix} -k_1 & k_2 & 0 & 0 & 0 & 0 \\ 0 & -k_2 & k_3 & 0 & 0 & 0 \\ k_1 & 0 & -k_3 & k_4 & 0 & 0 \\ 0 & 0 & 0 & -k_4 & 0 & 0 \\ 0 & 0 & 0 & 0 & -k_5 & 0 \\ 0 & 0 & 0 & 0 & k_5 & 0 \end{bmatrix}.$$

Assuming mass action kinetics the dynamical equations governing the model are the following:

$$
\begin{aligned}
\dot{x}_1 &= -k_2x_1 + k_3x_2 2k_4x_1^2 + k_5x_1^2x_2 \\
\dot{x}_2 &= k_1 - k_3x_2 + k_4x_1^2 - k_5x_1^2x_2. \qquad (9)
\end{aligned}
$$

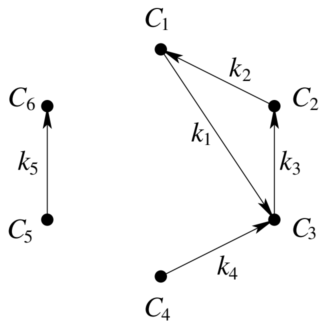


Figure 1. The reaction graph representing the reaction network $A1$ with 6 vertices and and 5 edges.

With parameters from Császár et al. (1981): $k_1 = 1$, $k_2 = 1$, $k_3 = 0.05$, $k_4 = 0.1$, $k_5 = 0.1$, the system shows oscillatory behavior. The structure of the dense realization is portrayed in Fig. 2 and it can be seen that the reaction graph of the original kinetic system in Fig. 1 is a subgraph of it.

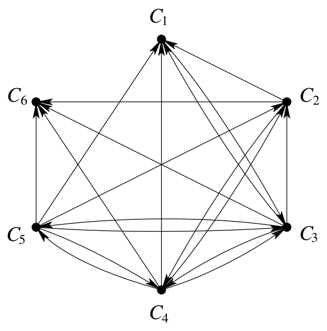


Figure 2. The reaction graph of the dense linearly conjugate realization of $A1$ with 6 vertices and 19 edges.

3. SUMMARY OF THE ALGORITHM

In order to find all graph structures we need to assign a unique identifier—a binary sequence—to each graph. Due to the superstructure property only those reactions need to be considered that take place in the dense realization. Thus, the number of digits is equal to the number of reactions (N) taking place in the dense realization, represented by an all-1 sequence.

The progress of Algorithm 1 is recorded in two different data structures: a set called *FoundSet* contains the graph identifiers that the algorithm has found so far; a first-in first-out queue called *Queue* contains all the graph identifiers that the algorithm still needs to process.

To start the algorithm the dense realization of the kinetic system—characterized by the matrices M and Y—is computed and the identifier of the dense realization is stored in the *FoundSet*. Then this identifier is pushed into the queue for further processing.

Within the algorithm the following procedure is applied repeatedly:

FindLinConjExcludeEdge(M, Y, R, i) computes a constrained dense linearly conjugate realization of the kinetic system excluding an edge (corresponding to index i) from the realization R according to constraints (7)-(8). If such a realization (denoted by the sequence U) exists, then reaction graph G_U is a subgraph of G_R and $U[i]$ is zero. If the problem is infeasible, then -1 is returned.

Algorithm 1

```
 1: procedure LIN. CONJ. GRAPH STRUCTURES($M, Y$)
 2:     while Size(Queue)$> 0$ do
 3:         $R$:= pop Queue
 4:         for $i = 1$ to $N$ do
 5:             if $R[i] = 1$ then
 6:                 $U$ :=FindLinConjExcludeEdge($M, Y, R, i$)
 7:                 if $U \geq 0$ and $U \notin FoundSet$ then
 8:                     $FoundSet$:= $FoundSet \cup \{U\}$
 9:                     push $U$ into $Queue$
10:                 end if
11:             end if
12:             print $R$
13:         end for
14:     end while
15: end procedure
```

The correctness of this algorithm can be proved similarly to the proof given in Ács et al. (2016). The basic idea is that we assume by contradiction that there is a sequence W which is not returned by the algorithm and examine the computation done on a sequence R with the following properties: R characterizes a linearly conjugate realization of the kinetic system, it is returned by the algorithm, G_W is a subgraph of G_R and considering these properties the number of edges in G_R is minimal. Among the children of R we can get either a sequence equal to W or an other sequence U with the same prescribed properties as R but with less edges. Both cases result a contradiction.

4. IMPLEMENTATION

To efficiently compute all different realizations, the software implementation has three independent components:

(1) The *Queue Server* process handles the FIFO *Queue* and it has $O(1)$ time complexity for pushing in or popping out items.
(2) The *Found Server* process stores all the structure IDs we have found so far—in the set called *FoundSet* in the pseudocode. The internal data structure requires $O(n)$ time for insertion, where n is the number of elements in the data structure, and $O(1)$ time for searching elements. Since the algorithm may find the same graph structure several times, we need such a data structure inside the *Found Server*. The time complexity for insertion can be further amortized, because during insertion between 1 and N number of graph identifiers are sent to the *Found Server*.
(3) The *Worker* applies the procedure FindLinConjExcludeEdge repeatedly to the given realization. After that it sends the IDs of the sub-graphs to the *Found Server* and sends the graph identifiers of the new ones to the *Queue Server* for further processing. After all sub-graphs of the current realization have been determined, they are written to the hard disk.

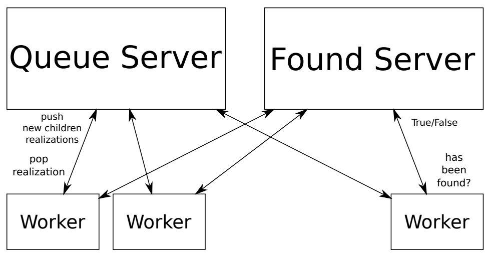


Figure 3. The main components of the implementation, a *Worker* requests a graph identifier of a realizations from the *Queue Server*, computes all of its children and checks novelty of children's graph identifiers with the *Found server*, then sends the new one to the *Queue Server*.

The communication between the three major components can be implemented in multiple ways, for example shared memory objects or Message Passing Interface (MPI) would be good solutions. On the other hand, as we will see in Section 5, the performance of the algorithm mainly depends on the number of initialized workers. Therefore, we chose a message queue solution that sends the necessary data over the TCP network and gives us the opportunity of distributed computing over multiple computers.

4.1 Possible levels of parallelization

In order to increase the efficiency of the implementation presented so far, we can introduce the following parallelization levels.

Independent workers As the layout of the implementation suggests in Fig. 3 we can utilize more then one worker, because they work completely independently of each other. In addition, this setup allows us to dynamically scale the number of workers during the execution, since a worker does not store any shared data between iterations. As we will see in the next section, the number of workers is basically limited by the number of available processors in the system.

On the other hand, we hit another limit when a single thread is not enough for a server process. In this case multiple server threads can be initialized to accommodate the large number of requests from the workers. Although this scenario is beyond the scope of this paper.

Parallel edge exclusion Parallel computation of the edge exclusion in Algorithm 1 is also possible—as it was shown in Ács et al. (2016)—because no information is shared during the computation of the edge exclusions. However, this approach has two potential drawbacks. First, it may yield more frequent communication between the two servers and a worker, but it can be compensated by buffering queue items in the worker. Second, there is a cost to initialize a pool of threads and it is often the case some of the threads may encounter an infeasible linear program. As a result of that the resources for these threads will not be utilized. This scenario can be avoided by performing the feasibility check before the thread initialization. Hence, we can initialize as many threads as feasible linear programs.

Parallel computation of the columns of matrix A_k This solution can be applied in the case of dynamical equivalence, because in this case the reaction rate coefficients in different columns of A_k are independent of each other.

We remark that the latter two levels of parallelization can be really advantageous in the following two cases: 1, the size of the linear program is large enough to outweigh the cost of allocating resources for parallel computation; 2, efficient vectorization of the computation of edge exclusion and/or computation of the LP tasks on Graphical Processing Units can be used (see, e.g. Spampinato and Elstery (2009)).

5. COMPUTATIONAL RESULTS

In this section we analyze the performance of our software implementation on three illustrative examples. The first one is a computational example from Section 2.6. Examples 2 and 3 are selected form the systems biology literature. The general case of linear conjugacy is investigated in the first example, in the rest only dynamical equivalence is practically relevant.

5.1 Measurement setup

All the computational experiments were carried out on a Lenovo D60 workstation with two 2.60GHz Xeon (E5-2650 v2) processors and with 32 Gb RAM (DDR3 1600 MHz, 0.6ns). The software was written in Python (ver. 2.7.6). Additionally, Python packages such as pyzmq (ver. 14.7.0), cyLP (0.7.2), Cython (ver. 0.23.4) and CBC (ver. 2.8.5) were used. The linear programs were solved with the CLP solver, which is part of CBC.

To give room to the server processes and the operating system's own maintenance we limited the number of workers to 31. Each worker was implemented as one Python process mainly to escape Global Interpreter Look (GIL) in Python. During the computation only the first level of parallelization demonstrated in Section 4.1 was implemented.

5.2 Examples

Example – $A1$ The first example is taken from Section 2.6. The dense realization is shown in Fig. 2 and it contains 19 reactions.

After computing all linearly conjugate realizations of $A1$ by applying the implementation in Section 4 we end up with 17160 structurally different graphs realizing the same dynamics (determined by (9)). It was checked and confirmed that these structures are exactly the same that were published in the electronic supplement of Ács et al. (2016).

To test the scalability of our implementation we repeated the computation of $A1$ with different number of workers, the corresponding computation time are shown in Fig. 4. In each case the total computation time was calculated by averaging the execution time of the workers. The small standard deviation in the brackets indicates that the work load is evenly distributed among the workers.

The first column of Table 1 shows the internal counters of the algorithm for this example. From this we can read that how many LPs were solved during the computation, how many of them were feasible and how many of them were accepted as new realizations. We can also conclude that in this small example (there are 6 complexes in the reaction graph) only

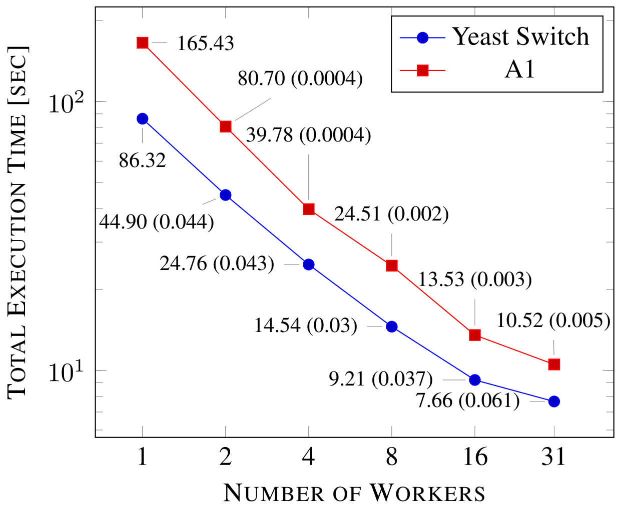


Figure 4. Average execution time of the workers on examples $A1$ and yeast switch, the axes are in log scale. The standard deviation for each case is shown in brackets.

8.12% of the executed linear programs lead to new realizations, the others are either infeasible LPs or the algorithm has already found those structures.

Counter	A1	Yeast Switch	5-node-repressilator
LP Solved	211,264 (8.12%)	18041 (4.00%)	4,008,995 (1.62%)
Valid Solutions	131,960 (13.00%)	4240 (17.01%)	542,919 (11.99%)
Found Again	114,801 (14.50%)	3460 (20.83%)	477,850 (13.62%)
All Realizations	17160 (100%)	721 (100%)	65071 (100%)

Table 1. These figures show the internal counters of Algorithm 1 while running different examples.

G_1/S transition in Budding Yeast This example is taken from Conradi et al. (2007), and models a switch-like behavior in yeast cell cycle regulations. The details of the model can be found in the original paper. The dynamically equivalent sparse and dense structures of this model (shown on Fig. 6) were computed in Szederkényi et al. (2011). Here we use the same state variables, namely: x_1: [Sic1], x_2: [Sic1P], x_3: [Clb], x_4: [Clb·Sic1], x_5: [Clb·Sic1P], x_6: [Cdc14], x_7: [Sic1P·Cdc14], x_8: [Clb·Sic1P·Cdc14], x_9: [Clb·Sic1·Clb]. The graph structure contains 19 complexes:

$$\begin{aligned}
&C_1 = X_2, C_2 = \emptyset, C_3 = X_1, C_4 = X_3 + X_1, C_5 = X_4,\\
&C_6 = X_3, C_7 = X_2 + X_3, C_8 = X_5, C_9 = X_3 + X_4,\\
&C_{10} = X_9, C_{11} = X_3 + X_5, C_{12} = X_2 + X_6, C_{13} = X_7,\\
&C_{14} = X_1 + X_6, C_{15} = X_5 + X_6, C_{16} = X_8,\\
&C_{17} = X_4 + X_6.
\end{aligned}$$

The parameter values applied in the computation are listed in Szederkényi et al. (2011). The dense realization contains 28 reactions, while the sparse has 18. It should be pointed out that the original realization of the model reported in Conradi et al. (2007) contains 18 reactions, i.e. this realization is sparse and it is the only sparse realization of the given dynamics.

With 31 workers the computation of this model took 7.66 seconds and we found 729 different realization structures of the kinetic system. To test the scalability of the implementation we

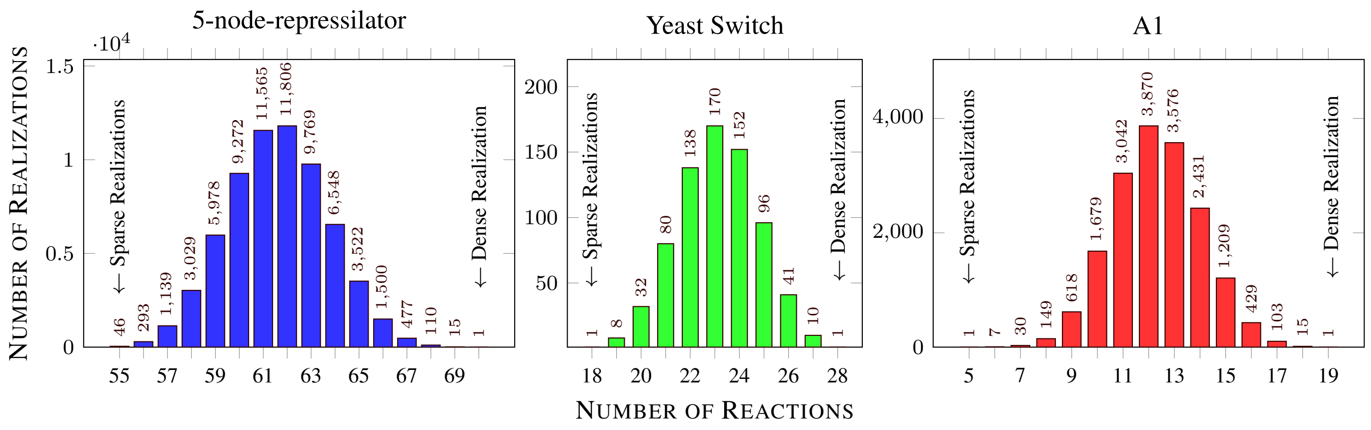


Figure 5. The number of realizations for a given reaction number is shown for each example. The 5-node-repressilator and the yeast switch example was calculated with dynamical equivalence. On the other hand $A1$ was calculated with linear conjugacy (check Section 2.5 for details).

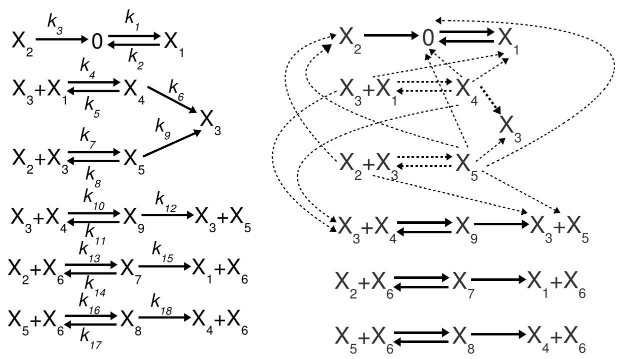

k_1 ... k_{18}
X_2, 0, X_1, X_3+X_1, X_4, X_3, X_2+X_3, X_5, X_3+X_4, X_9, X_3+X_5, X_2+X_6, X_7, X_1+X_6, X_5+X_6, X_8, X_4+X_6

Figure 6. Left: the originally published model of the G_1/S transition model from Conradi et al. (2007), which is also a sparse realization, right: the dynamical equivalent dense realization of the same model. This figure is taken from Szederkényi et al. (2011).

Example	LP variables	Avg. sol. Time	Std. of sol. Time
A1	76	0.792 msec	0.124 msec
Yeast Switch	578	7.616 msec	0.694 msec
5-node-repressilator	5202	229.4 msec	23.41 msec

Table 2. These figures show the size of the corresponding linear program as well as the average solution time for the examples.

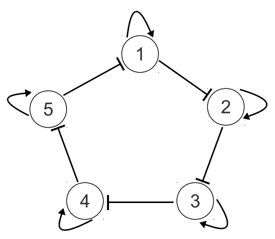
did the same computation with different number of workers as well (see Fig. 4).

Deficiency	Number of Realizations
2	81
5	19
6	629

Table 3. Table lists the deficiencies and their occurrence for the Budding Yeast example.

Given all realizations we can easily calculate the deficiency of every realization (see, Table 3). The complex C_6 becomes isolated in 81 realizations—these are the ones with deficiency 2. The role of C_6 is then taken up by combinations of reactions originating from complexes C_5 and C_8, the possible combinations are given by the dense realization (shown in the right panel of Fig. 6).

5-node-repressilator with auto activation This model was reported in Müller et al. (2006) and it was adapted and investigated in details in Szederkényi et al. (2011), where it was concluded—in the dynamically equivalent case—the dense realization has 80 reactions and a sparse one has 55 reactions. The model contains five genes with auto-activation and each gene represses another gene. The layout, connection pattern of genes and the corresponding reactions are shown in Fig. 7.



Auto-activation 1	$G_i + P_i \rightleftarrows G_i^A$
Protein production 1	$G_i^A \to G_i^A + P_i$
Repression 1	$G_i + P_j \rightleftarrows G_i^R$
Auto-activation 2	$G_i^R + P_i \rightleftarrows G_i^{AR}$
Repression 2	$G_i^A + P_j \rightleftarrows G_i^{AR}$
Protein production 2	$G_i^{AR} \to G_i^{AR} + P_i$
Protein degradation	$P_i \to \emptyset$

Figure 7. Left: Layout and connection pattern of the 5-node-repressilator. Each gene labeled with numbers 1 to 5 consists of 11 reactions responsible for gene expression, repression, protein degradation as listed on the right panel. This figure is taken from Szederkényi et al. (2011).

As in the previous example the dense-sparse gap indicates the possibility of different realizations. After computing all realizations we ended up with 65071 structurally different realizations. The computation took 554 minutes, this is due to the size of the linear program which has 5202 decision variables (shown in the third row of Table 2).

Finally, we can investigate the distribution of the number of reactions among the distinct realizations (Fig. 5).

6. CONCLUSION

We have presented an improved algorithm which calculates all possible reaction graph structures corresponding to linearly conjugate realizations of kinetic systems. Due to the combinatorial nature of the problem the efficient organization of the computations is of significant interest. Therefore, the implementation procedure has been aimed to maximize the available processing throughput by defining simple workers working on the same task queue. Based on previous results the correctness of the new approach with a task queue instead of a hierarchical stack-structure has also been outlined, ensuring that the improved implementation finds all possible graph structures. This was also computationally confirmed on a previously studied benchmark model having 17160 different graph structures. We have demonstrated the scalability of the solution on further computation examples and investigated their performances. It has been shown that the improved implementation runs significantly faster on the same hardware than the solution reported in Ács et al. (2016). Further work will be focused on implementing the briefly mentioned additional levels of parallelization. Potential application fields of the method are the efficient identifiability-related structural analysis of biochemical network models or the design of an entire set of kinetic structures based on prescribed nonlinear dynamics.

7. ACKNOWLEDGMENT

The authors would like to thank Dr. István Reguly for his help on the manuscript and György Lipták for his help with the implementation. GSz, BÁ and ZAT gratefully acknowledge the support of the Hungarian National Research, Development and Innovation Office – NKFIH through grant OTKA NF104706. Furthermore, the partial support support of PPKE through the grant KAP 15-052-1.1 is also acknowledged.

REFERENCES

Chellaboina, V., Bhat, S.P., Haddad, W.M., and Bernstein, D.S. (2009). Modeling and analysis of mass-action kinetics – nonnegativity, realizability, reducibility, and semistability. *IEEE Control Systems Magazine*, 29, 60–78.

Chis, O., Banga, J.R., and Balsa-Canto, E. (2011). Structural identifiability of systems biology models: A critical comparison of methods. *PLoS ONE*, 6(11), e27755:1–16.

Conradi, C., Flockerzi, D., Raisch, J., and Stelling, J. (2007). Subnetwork analysis reveals dynamic features of complex (bio)chemical networks. *Proceedings of the National Academy of Sciences*, 104(49), 19175–19180. doi: 10.1073/pnas.0705731104.

Craciun, G. and Pantea, C. (2008). Identifiability of chemical reaction networks. *Journal of Mathematical Chemistry*, 44, 244–259.

Császár, A., Jicsinszky, L., and Turányi, T. (1981). Generation of model reactions leading to limit cycle behaviour. *Reaction Kinetics and Catalysis Letters*, 18, 65–71.

Feinberg, M. (1987). Chemical reaction network structure and the stability of complex isothermal reactors - I. The deficiency zero and deficiency one theorems. *Chemical Engineering Science*, 42 (10), 2229–2268.

Feinberg, M. (1988). Chemical reaction network structure and the stability of complex isothermal reactors - II. Multiple steady states for networks of deficiency one. *Chemical Engineering Science*, 43, 1–25.

Horn, F. and Jackson, R. (1972). General mass action kinetics. *Archive for Rational Mechanics and Analysis*, 47, 81–116.

Johnston, M.D. and Siegel, D. (2011). Linear conjugacy of chemical reaction networks. *Journal of Mathematical Chemistry*, 49, 1263–1282.

Johnston, M.D., Siegel, D., and Szederkényi, G. (2012a). A linear programming approach to weak reversibility and linear conjugacy of chemical reaction networks. *Journal of Mathematical Chemistry*, 50, 274–288. doi:10.1007/s10910-011-9911-7.

Johnston, M.D., Siegel, D., and Szederkényi, G. (2012b). Dynamical equivalence and linear conjugacy of chemical reaction networks: New results and methods. *MATCH Commun. Math. Comput. Chem.*, 68, 443–468.

Müller, S., Hofbauer, J., Endler, L., Flamm, C., Widder, S., and Schuster, P. (2006). A generalized model of the repressilator. *Journal of Mathematical Biology*, 53(6), 905–937. doi: 10.1007/s00285-006-0035-9.

Spampinato, D.G. and Elstery, A.C. (2009). Linear optimization on modern gpus. In *IEEE International Symposium on Parallel Distributed Processing*, 1–8. doi: 10.1109/IPDPS.2009.5161106.

Szederkényi, G. (2010a). Computing reaction kinetic realizations of positive nonlinear systems using mixed integer programming. In *8th IFAC Symposium on Nonlinear Control Systems - NOLCOS 2010, Bologna, Italy, 1-3 September*, ThP04.2.

Szederkényi, G. (2010b). Computing sparse and dense realizations of reaction kinetic systems. *Journal of Mathematical Chemistry*, 47, 551–568. doi:10.1007/s10910-009-9525-5.

Szederkényi, G., Banga, J.R., and Alonso, A.A. (2011). Inference of complex biological networks: distinguishability issues and optimization-based solutions. *BMC Systems Biology*, 5, 177. doi:10.1186/1752-0509-5-177.

Tuza, Z.A., Szederkényi, G., Hangos, K.M., and A. A. Alonso, J.R.B. (2013). Computing all sparse kinetic structures for a Lorenz system using optimization methods. *International Journal of Bifurcation and Chaos*, 23, 1350141(1–17).

Ács, B., Szederkényi, G., Tuza, Z.A., and Tuza, Z. (2015). Computing linearly conjugate weakly reversible kinetic structures using optimization and graph theory. *MATCH Commun. Math. Comput. Chem.*, 74, 481–504.

Ács, B., Szederkényi, G., Tuza, Z., and Tuza, Z.A. (2016). Computing all possible graph structures describing linearly conjugate realizations of kinetic systems. *Computer Physics Communications*, 204, 11–20. doi: doi:10.1016/j.cpc.2016.02.020.