EGAC: A Genetic Algorithm to Compare Chemical Reaction Networks

Stefano Tognazzi
IMT School for Advanced Studies Lucca, Italy
stefano.tognazzi@imtlucca.it

Mirco Tribastone
IMT School for Advanced Studies Lucca, Italy
mirco.tribastone@imtlucca.it

Max Tschaikowski
IMT School for Advanced Studies Lucca, Italy
max.tschaikowski@imtlucca.it

Andrea Vandin
IMT School for Advanced Studies Lucca, Italy
andrea.vandin@imtlucca.it

ABSTRACT

Discovering relations between chemical reaction networks (CRNs) is a relevant problem in computational systems biology for model reduction, to explain if a given system can be seen as an abstraction of another one; and for model comparison, useful to establish an evolutionary path from simpler networks to more complex ones. This is also related to foundational issues in computer science regarding program equivalence, in light of the established interpretation of a CRN as a kernel programming language for concurrency. Criteria for deciding if two CRNs can be formally related have been recently developed, but these require that a candidate mapping be provided. Automatically finding candidate mappings is very hard in general since the search space essentially consists of all possible partitions of a set. In this paper we tackle this problem by developing a genetic algorithm for a class of CRNs called *influence networks*, which can be used to model a variety of biological systems including cell-cycle switches and gene networks. An extensive numerical evaluation shows that our approach can successfully establish relations between influence networks from the literature which cannot be found by exact algorithms due to their large computational requirements.

KEYWORDS

Chemical Reaction Networks; Ordinary Differential Equations; Model Comparison

ACM Reference format:
Stefano Tognazzi, Mirco Tribastone, Max Tschaikowski, and Andrea Vandin. 2017. EGAC: A Genetic Algorithm to Compare Chemical Reaction Networks. In *Proceedings of GECCO '17, Berlin, Germany, July 15-19, 2017,* 8 pages.
DOI: http://dx.doi.org/10.1145/3071178.3071265

1 INTRODUCTION

A popular model in many natural sciences, chemical reaction networks (CRNs) have become popular in computer science in light of the convergence between computational processes and molecular systems [23]. In this paper we study the problem of comparing CRNs: that is, to decide whether a given *source* CRN can be related to a *target* one in some appropriate sense. This is mainly motivated by applications in computational biochemistry. For example, one may want to explain an evolutionary pathway from a simple system to a more complex one which still exhibits some of the original behavior [4–6]. In DNA computing, one would like to compare a *specification* CRN, representing the observable dynamics of interest, against an *implementation* CRN which takes into account constraints imposed by the materials and the technology used [24]. Seeing CRNs as a basic programming language for concurrency [3], CRN comparison resembles the problem of program comparison and minimization, a fundamental issue in computer science.

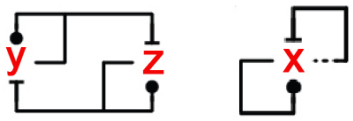
Notions of CRN comparison that do not consider quantitative dynamics ([13, 18, 19]) do not allow answering questions regarding important dynamical properties: for instance, whether two distinct CRNs can achieve the same switch-like behavior under appropriate conditions [6]. A kinetics-aware CRN comparison is presented in [24], but this is specialized for DNA implementation.

A more general result based on the notion of *emulation* has been recently proposed by Cardelli [4]. It is based on the well-known interpretation of a CRN as a system of ordinary differential equations (ODEs) whereby each species of the CRN is associated with an ODE that governs the net change of its concentration as a function of time (e.g. [26]). Roughly speaking, emulation describes that the ODE solutions of a source CRN exactly overlap those of a target CRN at all time points, for a given mapping of species of one CRN onto the other. However, an algorithm to identify an emulation (i.e., to synthesize a mapping of species) is not given in [4]. Recently, emulation has been shown to be a stricter variant of *backward differential equivalence* (BDE) [11], an equivalence relation over the ODE variables. A partition-refinement algorithm computes the coarsest BDE that refines a given initial partition of species, running in polynomial time and space [8, 11]. On its own, however, it does not directly give emulations because a coarsest BDE refinement is not necessarily an emulation; however, an emulation must be a refinement of the largest BDE.

Instead, finding all possible emulations between two CRNs is possible through an algorithm, *CAGE*, that computes all BDEs (which thus will contain all emulations) [7]. However often emulations may be a small fraction thereof. This may waste most of the computation required by the algorithm. In practice, this means that networks no larger than 30 species can currently be analyzed, beyond which the large space complexity of *CAGE* starts to have an effect.

GECCO '17, Berlin, Germany

DOI: http://dx.doi.org/10.1145/3071178.3071265



(a) MI (b) AM

Figure 1: Examples of influence networks.

In this paper we present an approach to discovering relations between CRNs through emulations using a genetic algorithm, which we name *EGAC*: *Emulations through Genetic Algorithm Computations*. We focus on a class of CRNs called *influence networks* [4], which is relevant for the formal description of biomolecular interactions such as inhibitions and activations; as such, influence networks have been used to describe cell cycles and gene networks, for instance [6]. The main features of our approach are: (i) a compact encoding of an individual, which exploits information about the structure of an influence network and about necessary conditions for the existence of an emulation; and (ii) the design of a fitness function that uses the BDE partition-refinement algorithm to measure the distance between the individual and a possible emulation.

We apply our algorithm to the influence networks first introduced in [4] and then algorithmically analyzed with *CAGE* [7]. With a prototype implementation, we find that *EGAC* recovers previously found emulations. More important, it can successfully discover relations in instances where *CAGE* fails, thus promoting *EGAC* as an alternative for the analysis of larger scale networks.

Paper outline. Section 2 briefly reviews background material. Section 3 discusses the design of *EGAC*. Section 4 presents its implementation and the numerical results. Section 5 discusses related work while Section 6 concludes.

2 BACKGROUND

To make the paper self-contained we review of CRNs, we discuss the notions of emulation to compare CRNs, and BDE, the induced equivalence relation over species.

2.1 Chemical Reaction Networks

Formally, a CRN is a pair $(\mathcal{S}, \mathcal{R}_\mathcal{S})$ of a finite set of *species* $\mathcal{S}$ and a finite set of *reactions* $\mathcal{R}_\mathcal{S}$. A reaction is a triple written in the form $\rho \xrightarrow{k} \pi$, where ρ and π are multisets of species, called *reactants* and *products*, respectively, and $k > 0$ is the *reaction rate*. The operator $+$ denotes multiset union, e.g., $X + Y + Y$ is the multiset $\{\!\{X, Y, Y\}\!\}$. For example, the reaction

$$A + B \xrightarrow{k} C \tag{1}$$

indicates that species A and B are transformed into species C with rate k. Throughout this paper we shall consider the well-known interpretation of a CRN as a system of ODEs with mass-action kinetics [26]. This associates each species X with a variable V_X. Its ODE describes the net change of the concentration as a function of time. More specifically, a CRN $(\mathcal{S}, \mathcal{R}_\mathcal{S})$ is associated with the ODE system $\dot{V} = f(V)$, with $f : \mathbb{R}^\mathcal{S} \to \mathbb{R}^\mathcal{S}$, where each component f_X, with $X \in \mathcal{S}$ is defined as:

$$f_X(V) := \sum_{\rho \xrightarrow{\alpha} \pi \in \mathcal{S}} (\pi(X) - \rho(X)) \cdot \alpha \cdot \prod_{Y \in \mathcal{S}} V_Y^{\rho(Y)} .$$

Here $\rho(X)$ and $\pi(X)$ denote the multiplicity of species X in the multisets ρ and π, respectively. It satisfies a unique solution $V(t) = (V_X(t))_{X \in S}$ for any given *initial condition* $V(0)$. For example, in the reaction (1), the ODEs are given by

$$\dot{V}_A = -kV_AV_B \qquad \dot{V}_B = -kV_AV_B \qquad \dot{V}_C = kV_AV_B$$

which can be solved using standard numerical techniques.

2.2 Influence Networks

Influence networks are a special class of CRNs that can be used to describe a variety of biological processes [4]. An influence network can be represented as a graph of (stateful) *influence nodes* connected via *influence edges* that express activation or inhibition. Figure 1 shows two networks, MI and AM, which will be used throughout the remainder of this paper. The AM network models a cell cycle switch that is needed to avoid genetic instability during replication, while MI is a mutual inhibition system [4].

Each influence node (e.g., X and Y) is translated into a pattern of three species (e.g., X_0, X_1, X_2, and Y_0, Y_1, Y_2) and four reactions. The reactions realize the influence edges. Each node can have a connection at each terminal: *high output* (solid line), representing the species with subscript 0, *low output* (dashed line), representing the species with subscript 2, *activation input* (circle) and *inhibition input* (bar). Species with index 1 introduce nonlinearity in transitions and are never otherwise connected to the network [4]. If i and a are the inhibitor and activation input species for the influence node X, respectively, then X is associated with the following four reactions:

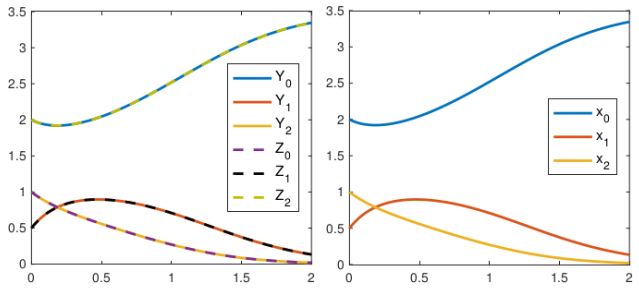
$$X_0 + i \xrightarrow{\alpha_{01}} i + X_1, \qquad X_1 + i \xrightarrow{\alpha_{12}} i + X_2,$$
$$X_2 + a \xrightarrow{\alpha_{21}} a + X_1, \qquad X_1 + a \xrightarrow{\alpha_{10}} a + X_0,$$

where $\alpha_{01}, \alpha_{12}, \alpha_{21}, \alpha_{10}$ are given rate coefficients.

Example 2.1. For a choice of rates useful in the forthcoming examples, MI and AM have the following CRNs (left and right, respectively):

$$Y_0 + Z_0 \xrightarrow{k_1} Z_0 + Y_1$$
$$Y_1 + Z_0 \xrightarrow{k_2} Z_0 + Y_2$$
$$Y_2 + Y_0 \xrightarrow{k_3} Y_0 + Y_1$$
$$Y_1 + Y_0 \xrightarrow{k_4} Y_0 + Y_0$$
$$Z_2 + Z_0 \xrightarrow{k_1} Z_0 + Z_1$$
$$Z_1 + Z_0 \xrightarrow{k_2} Z_0 + Z_0$$
$$Z_0 + Y_0 \xrightarrow{k_3} Y_0 + Z_1$$
$$Z_1 + Y_0 \xrightarrow{k_4} Y_0 + Z_2$$

$$x_0 + x_2 \xrightarrow{k_1} x_2 + x_1$$
$$x_1 + x_2 \xrightarrow{k_2} x_2 + x_2$$
$$x_2 + x_0 \xrightarrow{k_3} x_0 + x_1$$
$$x_1 + x_0 \xrightarrow{k_4} x_0 + x_0$$



(a) MI (b) AM

Figure 2: ODE solutions of (a) MI and (b) AM. Trajectories of MI overlap in pairs (i.e., Y_0/Z_2, Y_1/Z_1, Y_2/Z_0) thus they are not visually distinguishable in the plot.

Hereafter, with the aim of distinguishing between source and target CRN we will use upper-case and lower-case symbols, respectively, for both influence nodes and their respective species names.

2.3 Emulation

The notion of emulation has been introduced in [4] to compare CRNs. Roughly speaking, a source network is said to emulate a smaller target network if it is possible to find appropriate initial conditions such that the ODE trajectories of the two networks coincide. For instance, let us consider the ODE trajectories of MI and AM shown in Figure 2. For that choice of initial conditions the trajectories of MI overlap in pairs, Y_0/Z_2, Y_1/Z_1, Y_2/Z_0, and each pair does match one of AM, i.e., x_0, x_1, and x_2, respectively. The intuitive interpretation of emulation is that, while a more complex network exhibits richer behavior than a simpler one, under appropriate conditions it can reproduce the simple dynamics. This underlies an evolutionary argument that the complex network may descend from the simple one conservatively.

Formally, an emulation can be defined as a mapping μ from species of the source network to species of the target network that satisfies certain criteria. For instance, the overlappings in Figure 2 correspond to the emulation defined by the mapping:

$$\mu(Z_0) = x_0 \qquad \mu(Z_1) = x_1 \qquad \mu(Z_2) = x_2 \tag{2}$$

$$\mu(Y_0) = x_2 \qquad \mu(Y_1) = x_1 \qquad \mu(Y_2) = x_0 \tag{3}$$

Since nodes of influence networks correspond to triplet species, one is interested in *biologically meaningful* emulations by means of a *node mapping* that relates nodes with nodes, with the further constraint that 1-indexed species are mapped to 1-indexed species since those represent internal behavior. For instance, (2) and (3) are such a node mapping. Given target and source nodes x and Y, respectively, we denote by $Y \to x$ a mapping that does not swap the indices 0 and 2. Else, the mapping is denoted by $Y \to \sim x$. Therefore, (2) and (3) can be equivalently represented as $Z \to x$ and $Y \to \sim x$. A node mapping which is an emulation is called *node emulation*.

We do not provide the original definition of emulation. Rather, we state an alternative characterization that is based on BDE, which relates ODE variables that have the same solution at all time points if initialized with the same initial conditions. This will be instrumental to the development of *EGAC*.

An emulation μ from a source CRN $(\mathcal{S},\mathcal{R}_\mathcal{S})$ to a target CRN $(\mathcal{T},\mathcal{R}_\mathcal{T})$ is characterized by the following two properties:

i) $\left\{\mu^{-1}(X_i) : X_i \in \mathcal{S}\right\}$ is a BDE of $(\mathcal{S},\mathcal{R}_\mathcal{S})$.

ii) If $\mathcal{S} \cap \mathcal{T} = \emptyset$, then $\left\{\mu^{-1}(x_i) \cup \{x_i\} : x_i \in \mathcal{T}\right\}$ is a BDE of the union CRN $(\mathcal{S} \cup \mathcal{T},\mathcal{R}_\mathcal{S} \cup \mathcal{R}_\mathcal{T})$.

In words, all source species mapped to the same target species are BDE equivalent in the source CRN, as well as in the union CRN. For instance, in our example $\mathcal{H}_{MI} = \{\{Y_1,Z_1\},\{Y_0,Z_2\},\{Y_2,Z_0\}\}$ is a BDE of MI, while $\mathcal{H}_{AMI} = \{\{Y_1,Z_1,x_1\},\{Y_0,Z_2,x_2\},\{Y_2,Z_0,x_0\}\}$ is a BDE of the CRN obtained by the union of MI and AM. The requirement $\mathcal{S} \cap \mathcal{T} = \emptyset$ is without loss of generality, as it is always possible to rename the species in a CRN. In particular, this is the case with the capitalization convention adopted in this paper.

Conditions i) and ii) are important because [4] describes criteria to check if a given candidate mapping is an emulation but does not provide an algorithm to search for emulations. The relationship with BDE goes toward an algorithm treatment. However the possibility of computing the largest BDE alone does not suffice because, in general, the largest BDE does not satisfy the emulation criteria. Also, there might be more than one emulation among two CRNs. This is overcome in [7] where the authors provide an algorithm, *CAGE*, that uses the largest-BDE computation as an inner step and returns all possible emulations among two CRNs. This is done by first computing all possible BDEs of the union CRN, filtered through the above two characterizing properties.

CAGE is general: in fact, it can be used to compute all BDEs of a large class of nonlinear ODE systems (including, for instance, mass-action CRNs that are not induced by influence networks). The algorithm is driven by purely geometric properties of the ODEs. As a consequence, domain-specific information cannot be encoded straightforwardly within *CAGE*, such as:

D1) only search biologically meaningful emulations between influence networks (i.e., node mappings);

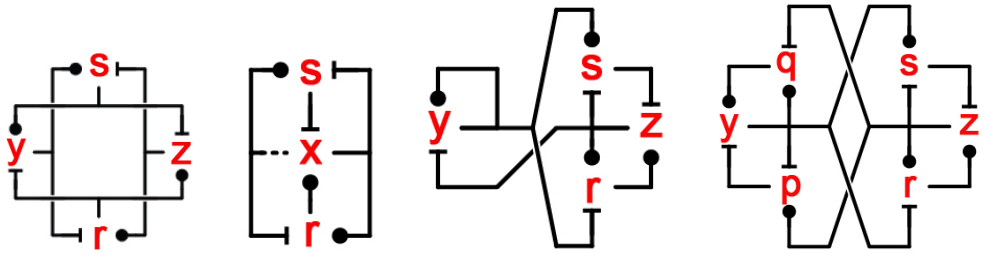
D2) avoid searching for BDEs that are not emulations.

Because of this, in practice *CAGE* may not scale well with increasing network sizes. In addition to the exact algorithm, a bounded version (hereafter called *bCAGE*) is discussed in [7] which provides an under-approximation of the set of all BDEs with the aim of reducing memory consumption. With our domain-specific *EGAC* we encode domain-specific knowledge directly in the algorithm, showing its effectiveness with respect to both *CAGE* and *bCAGE* when applied to discovering emulations for influence networks.

3 EGAC

In *EGAC*, we express the domain-specific knowledge D1 and D2 with the encoding of the individual and a specific design of the fitness function, respectively.

Individual. To encode D1, we only consider partitions of $\mathcal{S} \cup \mathcal{T}$ that represent a node mapping from $\mathcal{S}$ to $\mathcal{T}$. For this, we denote by $\mathcal{N}(\mathcal{S})$ (resp., $\mathcal{N}(\mathcal{T})$) the set of influence nodes of the source (resp., target) influence network, and equip $\mathcal{N}(\mathcal{S})$ with an order, denoted by $\sqsubseteq$. Then, the individual consists of an array where each element contains the image of the source node at that position, according



(a) QI (b) CCR (c) GW (d) NCC

Figure 3: Further considered influence networks.

to the given ordering. For instance, the node emulation given by $Y \to \sim x$ and $Z \to x$ in our running example is represented with the individual $[\sim x, x]$ according to the lexicographical ordering.

More precisely, we have the following:

(1) The individual is a sequence of symbols $[s_1, \ldots, s_n]$ with $n = |\mathcal{N}(\mathcal{S})|$ and $s_i \in \{x \mid x \in \mathcal{N}(\mathcal{T})\} \dot\cup \{\sim x \mid x \in \mathcal{N}(\mathcal{T})\}$.
(2) An x, with $x \in \mathcal{N}(\mathcal{T})$, at position $1 \leq i \leq n$ in the individual induces the mapping $X \to x$, where X is the i-th symbol in the ordered sequence induced by $\sqsubseteq$.
(3) A $\sim x$, with $x \in \mathcal{N}(\mathcal{T})$, at position $1 \leq i \leq n$ in the individual induces the mapping $X \to \sim x$, where X is the i-th symbol in the ordered sequence induced by $\sqsubseteq$.

The partition $\mathcal{H}$ of $\mathcal{S} \cup \mathcal{T}$ corresponding to an individual *ind* is constructed in Algorithm 1. First, it generates the set of singletons $\{\{x_i\} \mid x_i \in \mathcal{T}\}$. Then it populates each singleton block with the source species from $\mathcal{S}$ mapped to the corresponding target species.

Selection, Mutation and Fitness. Selection is done with a standard tournament scheme. At every generation a series of tournaments determines which individuals are the most fit. Then it proceeds with the two-point cross-over and mutation operations on the surviving individuals. Mutation was implemented using Algorithm 2, performing a swap between two elements of an individual. In experiments not reported here, using standard mutation functions such as bit-flip led to less successful results.

The routine implementing the fitness function relies on the auxiliary function *isNodeMapping* (Algorithm 3), which verifies whether a given individual does encode a node mapping. Such a check is necessary because, in contrast to mutation, cross-over does not transform node mappings to node mappings in general.

Algorithm 4 shows the overall computation of the fitness function. In case an individual does not encode a node mapping, it is associated with a constant penalty P. Otherwise, the auxiliary routine *individualToPartition* computes $\mathcal{H}$, the partition of $\mathcal{S} \cup \mathcal{T}$ representing the individual *ind*. Then $\mathcal{H}'$ is obtained as the coarsest BDE partition of $\mathcal{S} \cup \mathcal{T}$ which refines $\mathcal{H}$. This is done using the partition-refinement algorithm from [8] whose time complexity is polynomial in the number of species and reactions of the input CRN. If $\mathcal{H}' = \mathcal{H}$, then the input partition $\mathcal{H}$ is a BDE, hence an emulation. Instead, if $\mathcal{H}' \neq \mathcal{H}$, then partition $\mathcal{H}'$ has more blocks

Algorithm 1 Routine computing the partition $\mathcal{H}$ underlying an individual *ind*; i_X denotes the position in *ind* associated to $X \in \mathcal{N}(\mathcal{S})$ while $ind[i_X]$ gives the element of the individual at i_X.

```
function INDIVIDUALTOPARTITION(ind)
    for x in N(T) do
        H_{x,0} ← {x_0}
        H_{x,1} ← {x_1}
        H_{x,2} ← {x_2}
    end for
    for X in N(S) do
        if ind[i_X] ∈ N(T) then
            let x ∈ N(T) be such that x = ind[i_X]
            H_{x,0} ← H_{x,0} ∪ {X_0}
            H_{x,1} ← H_{x,1} ∪ {X_1}
            H_{x,2} ← H_{x,2} ∪ {X_2}
        else
            let x ∈ N(T) be such that ~x = ind[i_X]
            H_{x,0} ← H_{x,0} ∪ {X_2}
            H_{x,1} ← H_{x,1} ∪ {X_1}
            H_{x,2} ← H_{x,2} ∪ {X_0}
        end if
    end for
    return {H_{x,j} | x ∈ N(T), 0 ≤ j ≤ 2}
end function
```

Algorithm 2 The mutation function.

```
function MUTATE(ind)
    pick randomly i,j ∈ {1,...,|N(S)|}
    swap the nodes ind[i] and ind[j]
    return ind
end function
```

than $\mathcal{H}$. Since $|\mathcal{N}(\mathcal{T})| \leq |\mathcal{H}'|$ because $\mathcal{H}$ encodes a node mapping, $\mathcal{H}'$ encodes an emulation if and only if $|\mathcal{N}(\mathcal{T})| = |\mathcal{H}'|$. This motivates to set the fitness of a node-mapping partition to $|\mathcal{H}'|$.

In other words, with the objective of minimizing the fitness function, *EGAC* favors individuals such that the number of equivalence classes of their largest BDE refinement is closer to that of an emulation, $|\mathcal{N}(\mathcal{T})|$. In the worst case, *refineBDE* returns a partition with singleton blocks, of size $|\mathcal{S}| + |\mathcal{T}|$. Thus, it suffices to set

Algorithm 3 Auxiliary routine verifying whether a given individual encodes a node mapping.

```
function isNodeMapping(ind)
    H ← ∅
    for X in N(S) do
        if ind[i_X] ∈ N(T) then
            let x ∈ N(T) be such that x = ind[i_X]
        else
            let x ∈ N(T) be such that ~x = ind[i_X]
        end if
        H ← H ∪ {x}
    end for
    if H = N(T) then
        return true
    else
        return false
    end if
end function
```

Algorithm 4 The fitness function.

```
function fitnessFunction(ind)
    if not isNodeMapping(ind) then
        return P
    end if
    H ← individualToPartition(ind)
    H' ← refineBDE(H)
    return |H'|
end function
```

Table 1: Influence networks used for the experiments.

Network	AM	MI	CCR	GW	QI	NCC
Size (number of species)	3	6	9	12	12	18

Table 2: EGAC settings

Initial Population	Randomly generated
Selection	Tournament size 3
Cross-over probability	0.50
Cross-over function	Two-point
Mutation Probability	0.25
Mutation Function	Custom Mutation (Alg. 2)
Fitness Function	Custom Fitness Function (Alg. 4)
Type of optimization	Minimization
Stopping criterion	Max number of generations

$P \geq |\mathcal{S}| + |\mathcal{T}|$ in order to further penalize individuals which do not represent a node mapping.

Table 4: Sensitivity results: soundness

Source	*Target*	*Cross-over and Mutation*		
		0.30, 0.15	0.50, 0.25	0.70, 0.35
CCR	AM	2	1	1
QI	AM	2	1	2
GW	AM	2	1	2
NCC	AM	1	1	1
QI	MI	1	1	2
GW	MI	2	1	2
QI	CCR	4	1	4
NCC	CCR	1	1	1
NCC	QI	0	1	1

Table 5: Sensitivity results: scalability.

Source	*Target*	*Cross-over and Mutation*		
		0.30, 0.15	0.50, 0.25	0.70, 0.35
NCC2	AM	4	4	4
NCC2	AM2	8	7	7
NCC2	CCR	4	3	4
NCC6	AM	63	58	51
NCC3	CCR	2	6	9
NCC3	QI	7	4	5
NCC4	CCR	9	18	11
NCC4	AM4	0	1	0

4 EXPERIMENTAL RESULTS

Set-up. In this section we evaluate *EGAC* on influence networks from the literature, as collectively studied in [4]. These are depicted in Figure 1 and Figure 3, and summarized in Table 1. We performed three kinds of experiments. First, we studied the *soundness* of *EGAC* by comparing it against the exact *CAGE* algorithm of [7]. Then, we performed *scalability* experiments by considering networks of larger size which could not be handled by *CAGE*. Finally, we studied the *sensitivity* of the performance of *EGAC* on its parameters (i.e., cross-over and mutation probabilities, population and maximum number of generations).

For the scalability analysis, we compared *EGAC* against *bCAGE*, the bounded version of *CAGE* discussed in Section 2 that under-approximates the set of emulations. The set-up of *bCAGE* was such that it ran with the best possible settings to increase the likelihood of finding emulations: For every model that could not be covered by *CAGE*, the memory bound of *bCAGE* was set to 75% of the memory requirement that made *CAGE* issue an out of memory exception. This was possible because *CAGE* reports on its memory consumption during execution.

We chose representative instances of source/target CRNs so as to explore increasingly larger search spaces. The cardinality of the search space can be computed using the following.

Proposition 4.1. *For a given source network* $(\mathcal{S}, \mathcal{R}_\mathcal{S})$ *and target network* $(\mathcal{T}, \mathcal{R}_\mathcal{T})$*, the search space size is* $(2 \cdot |\mathcal{N}(\mathcal{T})|)^{|\mathcal{N}(\mathcal{S})|}$.

Table 3: Numerical results

Soundness Experiments										
Setting			*EGAC*					*CAGE*		*RS*
Source	*Target*	*Search space*	*Pop.*	*Gen.*	*Avg*	*Best*	*Runtime*	*Found*	*Runtime*	*Found*
CCR	AM	8	100	10	1.0	1	6 s	2	1 s	2
QI	AM	16	100	10	1.0	1	8 s	2	14 s	2
GW	AM	16	100	5	1.7	2	4 s	2	1 s	2
NCC	AM	64	100	20	0.7	1	17 s	2	1 523 s	2
QI	MI	256	100	5	0.7	2	5 s	2	15 s	2
GW	MI	256	100	5	0.2	2	5 s	2	1 s	2
QI	CCR	1 296	500	5	2.0	3	27 s	4	15 s	4
NCC	CCR	46 656	1 000	5	0.2	1	70 s	4	1 564 s	0
NCC	QI	262 144	2 000	30	0.2	1	885 s	4	1 556 s	1
Scalability Experiments										
Setting			*EGAC*					*bCAGE*		*RS*
Source	*Target*	*Search space*	*Pop.*	*Gen.*	*Avg*	*Best*	*Runtime*	*Found*	*Prop. 4.2*	*Runtime*	*Found*
NCC2	AM	4.0E+03	200	10	1.8	2	35 s	0	4	4 024 s	3
NCC2	AM2	1.6E+07	500	20	1.1	2	174 s	0	≥ 8	4 117 s	0
NCC2	CCR	2.0E+09	1 000	20	7.1	12	367 s	0	16	4 037 s	0
NCC6	AM	6.8E+10	1 000	20	5.9	9	918 s	0	64	2 576 s	0
NCC3	CCR	1.0E+14	1 000	20	3.9	12	512 s	0	64	17 611 s	0
NCC3	QI	1.8E+16	10 000	20	0.1	1	5 249 s	0	64	19 973 s	0
NCC4	CCR	4.0E+18	10 000	20	1.2	8	6 588 s	0	256	15 835 s	0
NCC4	AM4	4.7E+21	10 000	30	1.1	3	9 610 s	0	≥ 384	15 598 s	0

The networks for assessing scalability were synthesized by creating multiple independent copies (via appropriate renaming) of networks from Table 1. Thus, for example, NCC2 is formed by two copies of the original network NCC. This choice of set-up has a twofold motivation. First, it affords a biological interpretation in terms of an evolutionary pathway from smaller to larger networks, driven for instance by gene duplication [6]. Second, although for all of the as-constructed instances *CAGE* ran out of memory, we can provide an estimation for the number of node emulations there exist based on the knowledge of the number of node emulations between the same networks with a single replica.

Proposition 4.2. *Fix an influence source network $(\mathcal{S},\mathcal{R}_\mathcal{S})$, an influence target network $(\mathcal{T},\mathcal{R}_\mathcal{T})$ and assume that there are $m \geq 1$ node emulations from $\mathcal{S}$ to $\mathcal{T}$. Let $(\mathcal{S}^{\otimes n},\mathcal{R}_{\mathcal{S}^{\otimes n}})$ and $(\mathcal{T}^{\otimes \nu},\mathcal{R}_{\mathcal{T}^{\otimes \nu}})$ denote the corresponding CRNs with n and ν independent replicas of species and reactions, respectively. In the case $n \geq \nu$, the number of node emulations from $(\mathcal{S}^{\otimes n},\mathcal{R}_{\mathcal{S}^{\otimes n}})$ to $(\mathcal{T}^{\otimes \nu},\mathcal{R}_{\mathcal{T}^{\otimes \nu}})$ is at least*

$$m^n \cdot \sum_{\substack{k_1+\ldots+k_\nu=n,\\ k_1,\ldots,k_\nu \geq 1}} \binom{n}{k_1,\ldots,k_\nu},$$

where $\binom{n}{k_1,\ldots,k_\nu}$ denotes the multinomial coefficient. The above expression simplifies to $\nu! \cdot m^n$ in the case of $\nu \in \{1, n\}$ and provides the actual number of node emulations for $\nu = 1$.

As a consequence, this set-up allowed us to study instances which guarantee the existence of at least one emulation (since *EGAC* cannot clearly prove the absence of emulations between networks), and to have a measure of the relative frequency of occurrence of emulations with respect to the overall search space.

To study the effectiveness of the genetic algorithm, we additionally performed a comparison against a *random sampler* (*RS*) that generated a number of randomly generated individuals equal to the total number of individuals explored by *EGAC*. In particular, the stopping criterion was set as a limit on the number of generations. This allowed *EGAC* to search possibly multiple emulations rather than stopping after the minimum fitness is reached. Instead, population sizes and maximum number of iterations were varied depending on the total search space of a specific problem instance. Otherwise, the experiments were run using the settings presented in Table 2. These parameters were obtained after preliminary experimental tests with the goal of reducing runtimes.

Implementation. A prototype of *EGAC* was implemented in Python, using the DEAP package for genetic algorithms [12]. *EGAC* relies on the Eclipse-based tool ERODE [9], in order to execute the *refineBDE* function of Algorithm 4. ERODE implements a recently proposed polynomial-time algorithm to compute the coarsest BDE refinement of a partition of species for CRNs with mass action kinetics [8, 10]. *CAGE* and *bCAGE* were taken from the supplementary material accompanying [7], developed as Matlab scripts interfaced with ERODE. Runtimes were taken on a laptop machine equipped with a 2.20 GHz Intel Core i5-5200U and 8 GB RAM.

Results. The results are presented in Table 3. For each problem instance, we show the source and target CRN, the search space computed using Proposition 4.1, and the model-specific *EGAC* settings concerning the population size and maximum number of generations. For *EGAC* we report the average and the best number of distinct emulations found over 10 runs. For *CAGE*, and *bCAGE* we report the number of distinct emulations found at the end of one execution as well as its overall runtime. For *RS* we only report the number of emulations, since the runtime is comparable to *EGAC*.

For the soundness tests, the first main observation is that the encoding of the individual yielded a dramatic reduction of the search space, such that emulations can be feasibly found by enumerating all possible candidates. This shows the effectiveness of the encoding provided. Nevertheless we ran *EGAC* for completeness, always returning at least one emulation in all cases. Instead emulations were not found in the instances with the largest search spaces. We remark that with those settings the *EGAC* runtimes were considerably smaller than *CAGE*. The results of *RS* confirm that the smaller instances are not challenging due to the compact representation, but no emulations are found already between NCC and CCR.

As discussed above, *CAGE* issued out-of-memory errors in all scalability experiments. This is because its major bottleneck is in the computation of all BDE partitions of the source network, which has at least twice as many species as in the previous batch of instances. Running *bCAGE* with its best memory settings on the scalability instances did not find any emulation; for comparison, alongside we also report the estimations on the number of emulations computed using Proposition 4.2. By contrast, *EGAC* was able to find at least one emulation in all cases, running at most within 3 h; Except for the smallest instance, in all cases *RS* found no emulation. This can be justified by the fact that the fraction of emulations with respect to the total search space becomes increasingly smaller, thus defying a random sampling.

In Tables 4 and 5 we report the experiments regarding the sensibility of *EGAC* on both batches of networks. We experimented with three different configurations. The middle case is the same as in the experiments reported in Table 3. However, here we changed the initial population and the maximum number of generations. In particular, for the soundness experiments we considered a population of 1000 and a generation limit of 25. Instead, given the fact that the search space of the scalability experiments is significantly larger than the networks used in the soundness experiments, here we used a population formed by 20 000 individuals and a generation limit of 25. Throughout all the sensitivity experiments we used the same random seed in order to start with the same initial population for every experiment.

Unexpectedly, with the same crossover and mutation parameters used in Table 3 *EGAC* returns a different number of emulations because of the different population size and generation limit. In general, however, *EGAC* is robust to parametric changes in that it was able to find at least one emulation in almost all cases. It is worth noticing that with this setup *EGAC* explored up to 500 000 different individuals, which, in most of the scalability experiments, is a very small percentage of the whole search space.

Summary. In summary, we highlight the major strengths and weaknesses of our approach.

- *EGAC* is capable of discovering emulations between influence networks that can neither be handled by exact algorithms nor be approximated by heuristic variants.
- Being based on a genetic algorithm, it can be trivially parallelized, unlike *CAGE*. By tuning its parameters larger explorations of the search space are possible, to discover further emulations than those found with our parameter set-ups.
- On the other hand, unlike *CAGE*, *EGAC* cannot prove the absence of an emulation, an equally biologically interesting question that may help rule out certain evolutionary paths between networks.
- Finally, *EGAC* is designed specifically for influence networks with mass-action kinetics. *CAGE* instead, is more general because it allows finding all BDEs of a given system of ordinary differential equations.

5 RELATED WORK

Genetic algorithms and programming [15, 22] have been used in the past to compute analytical ODE solutions [1, 2], to simplify the optimal control of continuous systems [17], and to solve partial differential equations [21]. However, to our knowledge they have not been used for comparing systems.

Formally, a CRN can be understood as a labeled directed hypergraph, hence an equivalence relation over the species such as BDE corresponds to a specific (hyper-)graph partitioning. In this respect, our present work can be related to an established line of research on graph partitioning via genetic programming (see, e.g., the review [14]). The optimization goal in the graph partitioning problem is to find a minimal-cost edge cut such that the graph obtained after the cut is splitted in k balanced blocks. The blocks are balanced if and only if the difference in size between any two blocks of the partition is at most one. On the other hand, in the emulation problem our minimization goal is different. It does not consider the cost of splitting the partition, but the focus is on the fitness of the partition based on the notion of BDE.

However, our approach shares various concerns with graph partitioning techniques. For instance, in [14] a comparison is made between encodings of individuals based on *group numbers* or *edge encoding*. In group-number encoding each node is assigned a number that identifies the block of the partition to which the node belongs. In edge encoding every chromosome of a genome is bound to an edge, and the encoding expresses how the cut will be made. Different techniques such as node-clustering and gene reordering are presented in [14] but they hold no relationship with our problem. Our approach is based on group-number encoding. However, differently from the group-number encoding technique presented in [14], our encoding does not suffer the problem of redundancy (i.e. different individuals representing the same partition).

In the graph partitioning problem, the main constraint is *balance*. In *EGAC*, individuals are constrained to represent a node mapping. Two main techniques are proposed to solve the problem of individuals violating the constraint [14]. One suggests to repair the individual immediately after crossover and mutation. The other is based on giving a penalty to violating individuals, which is the scheme also adopted in *EGAC* with the idea that individuals that

violate the constraint can become fit after a series of cross-overs or mutations. This justifies the choice of the penalty scheme rather than the immediate repair scheme.

6 CONCLUSIONS AND FUTURE WORK

In this paper we presented *EGAC*, an approach based on a genetic algorithm to formally relate influence networks. Future work will aim at generalizations to other types of chemical reaction networks. Extensions that account for different kinetic mechanisms (such as Hill's kinetics) can already be accommodated. Indeed, this essentially amounts to using a fitness-function evaluation that invokes a more general partition-refinement algorithm for ordinary differential equations with rational derivatives [11]. Another interesting direction is to explore how to embed prior knowledge or assumptions within the genetic algorithm; for instance, the reuse of computations in new comparisons where the source or the target network share some structure with previously computed ones. Since parameters are often not known precisely due to finite precision measurements, stochastic noise or lack of information, it would be also interesting to extend *CAGE* and *EGAC* to CRNs with uncertain parameters [16, 20, 25].

Finally the relative strengths and weaknesses of *EGAC* with respect to the exact algorithm of *CAGE* call for a combined approach. Here, future work will investigate the development of an exact oracle that can effectively decide the absence of emulations, leaving it to the genetic algorithm to find them if there is at least one.

REFERENCES

[1] Grigorios N. Beligiannis, Lambros Skarlas, Spiridon D. Likothanassis, and Katerina G. Perdikouri. 2005. Nonlinear model structure identification of complex biomedical data using a genetic-programming-based technique. *IEEE Trans. Instrumentation and Measurement* 54, 6 (2005), 2184–2190.

[2] Hongqing Cao, Lishan Kang, Yuping Chen, and Jingxian Yu. 2000. Evolutionary Modeling of Systems of Ordinary Differential Equations with Genetic Programming. *Genetic Programming and Evolvable Machines* 1, 4 (2000), 309–337.

[3] Luca Cardelli. 2008. On process rate semantics. *Theoretical Computer Science* 391, 3 (2008), 190–215.

[4] Luca Cardelli. 2014. Morphisms of reaction networks that couple structure to function. *BMC Systems Biology* 8, 1 (2014), 84.

[5] Luca Cardelli, Attila Csikász-Nagy, Neil Dalchau, Mirco Tribastone, and Max Tschaikowski. 2016. Noise Reduction in Complex Biological Switches. *Scientific Reports* 6 (2016), 20214.

[6] Luca Cardelli, Rosa D. Hernansaiz-Ballesteros, Neil Dalchau, and Attila Csikász-Nagy. 2017. Efficient Switches in Biology and Computer Science. *PLOS Computational Biology* 13, 1 (2017), 1–16.

[7] Luca Cardelli, Mirco Tribastone, Max Tschaikowski, and Andrea Vandin. Comparing Chemical Reaction Networks: A Categorical and Algorithmic Perspective. In *LICS'16*.

[8] Luca Cardelli, Mirco Tribastone, Max Tschaikowski, and Andrea Vandin. Efficient Syntax-driven Lumping of Differential Equations. In *TACAS'16*.

[9] Luca Cardelli, Mirco Tribastone, Max Tschaikowski, and Andrea Vandin. ERODE: A Tool for the Evaluation and Reduction of Ordinary Differential Equations. In *TACAS'17*. To appear.

[10] Luca Cardelli, Mirco Tribastone, Max Tschaikowski, and Andrea Vandin. Forward and Backward Bisimulations for Chemical Reaction Networks. In *CONCUR'15*.

[11] Luca Cardelli, Mirco Tribastone, Max Tschaikowski, and Andrea Vandin. Symbolic Computation of Differential Equivalences. In *POPL'16*.

[12] Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau, and Christian Gagné. 2012. DEAP: Evolutionary Algorithms Made Easy. *Journal of Machine Learning Research* 13 (2012), 2171–2175.

[13] Steven Gay, Sylvain Soliman, and François Fages. 2010. A graphical method for reducing and relating models in systems biology. *Bioinformatics* 26, 18 (2010), i575–i581.

[14] Jin Kim, Inwook Hwang, Yong-Hyuk Kim, and Byung-Ro Moon. Genetic Approaches for Graph Partitioning: A Survey. In *GECCO'11*.

[15] John R. Koza. 1992. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press.

[16] J. C. W. Kuo and James Wei. 1969. Lumping Analysis in Monomolecular Reaction Systems. Analysis of Approximately Lumpable System. *Industrial & Engineering Chemistry Fundamentals* 8, 1 (1969), 124–133.

[17] Ngai Ming Kwok and Sam Kwong. Control of a Flexible Manipulator Using a Sliding Mode Controller with Genetic Algorithm Tuned Manipulator Dimension. In *GECCO'03*.

[18] Matthew Lakin, Andrew Phillips, and Darko Stefanovic. 2013. Modular Verification of DNA Strand Displacement Networks via Serializability Analysis. In *DNA Computing and Molecular Programming (LNCS)*, Vol. 8141. 133–146.

[19] Matthew Lakin, Darko Stefanovic, and Andrew Phillips. 2015. Modular verification of chemical reaction network encodings via serializability analysis. *TCS* (2015). In press.

[20] Genyuan Li and Herschel Rabitz. 1990. A general analysis of approximate lumping in chemical kinetics. *Chemical Engineering Science* 45, 4 (1990), 977–1002.

[21] Paul E. MacNeil. Genetic Algorithms and Solutions of an Interesting Differential Equation. In *GECCO'08*.

[22] Melanie Mitchell. 1998. *An Introduction to Genetic Algorithms.* MIT Press.

[23] Aviv Regev and Ehud Shapiro. 2002. Cellular abstractions: Cells as computation. *Nature* 419, 6905 (9 2002), 343–343.

[24] David Soloveichik, Georg Seelig, and Erik Winfree. 2010. DNA as a universal substrate for chemical kinetics. *PNAS* 107, 12 (2010), 5393–5398.

[25] Max Tschaikowski and Mirco Tribastone. 2016. Approximate Reduction of Heterogenous Nonlinear Models With Differential Hulls. *IEEE Trans. Automat. Contr.* 61, 4 (2016), 1099–1104.

[26] Eberhard O. Voit. 2013. Biochemical Systems Theory: A Review. *ISRN Biomathematics* 2013 (2013), 53.

A PROOFS

Proof of Proposition 4.1. Note that $|\mathcal{N}(\mathcal{S})|$ is the length of individuals, while $2|\mathcal{N}(\mathcal{T})| = |\{x \mid x \in \mathcal{N}(\mathcal{T})\} \dot\cup \{\sim x \mid x \in \mathcal{N}(\mathcal{T})\}|$ accounts for the number of possible values any entry in an individual can attain. This yields the claim. □

Proof of Proposition 4.2. Define $S^i = \mathcal{S} \times \{i\}$, $T^j = \mathcal{T} \times \{j\}$, $\mathcal{S}^{\otimes n} = \bigcup_{i=1}^{n} \mathcal{S}^i$, $\mathcal{T}^{\otimes \nu} = \bigcup_{j=1}^{\nu} \mathcal{T}^j$, let $\mu_1, \ldots, \mu_m$ denote all node emulations from $\mathcal{S}$ to $\mathcal{T}$ and let $\mu_k^{i,j} : \mathcal{S}^i \to \mathcal{T}^j$ arise from μ_k by replacing the source and target species X and x with their copies $X \times \{i\}$ and $x \times \{j\}$, respectively. Noting that any $\mu_{k_1}^{1,j_1} \cup \ldots \cup \mu_{k_n}^{n,j_n}$, where $1 \le k_l \le m$ and $\{j_1, \ldots, j_n\} = \{1, \ldots, \nu\}$, is a node emulation from $\mathcal{S}^{\otimes n}$ to $\mathcal{T}^{\otimes \nu}$, we infer that there are at least

$$m^n \cdot \nu! \cdot \left| \left\{ \mathcal{H} \;\middle|\; \mathcal{H} \text{ is partition of } \{1, \ldots, n\} \text{ with } \nu \text{ blocks} \right\} \right|$$

node emulations from $\mathcal{S}^{\otimes n}$ to $\mathcal{T}^{\otimes \nu}$ because $\nu!$ corresponds to the number of ways ν blocks can be mapped to ν target networks. Using elementary combinatorics, the above formula can be rewritten to

$$m^n \cdot \sum_{\substack{k_1 + \ldots + k_\nu = n, \\ k_1, \ldots, k_\nu \ge 1}} \binom{n}{k_1, \ldots, k_\nu}$$

In the case of $\nu = 1$, the lower bound can be seen to coincide with the actual number of node emulations. To this end, let us assume that $\mu : \mathcal{S}^{\otimes n} \to \mathcal{T}^1$ is a node emulation. Since $\mathcal{S}^1, \ldots, \mathcal{S}^n$ are pairwise disjoint sets, the ODEs of $\mathcal{S}^i$ do not depend on the species in $\mathcal{S}^{i'}$ whenever $i' \neq i$. Hence, $\mu_{|\mathcal{S}^i} : \mathcal{S}^i \to \mathcal{T}^1$ must be a node emulation, where $\mu_{|\mathcal{S}^i}$ denotes the restriction of μ to the set $\mathcal{S}^i$. This, in turn, implies that $\mu_{|\mathcal{S}^i} = \mu_k^{i,1}$ for some $1 \le k \le m$, thus showing the claim. □