Nonlinear Dyn (2021) 104:3991–4005
https://doi.org/10.1007/s11071-021-06482-4

ORIGINAL PAPER

Parameter identification of complex network dynamics

Arian Bakhtiarnia · Ali Fahim · Ehsan Maani Miandoab

Received: 12 August 2020 / Accepted: 17 April 2021 / Published online: 29 April 2021

Abstract Here, we present a novel approach called the "parameter identification of complex network dynamics" algorithm which combines elements of the sparse identification of nonlinear dynamics algorithm with a genetic algorithm to automatically and efficiently discover the underlying dynamics of complex networks from data with minimal domain-specific knowledge requirements. Testing the proposed algorithm on empirical complex network data verifies the accuracy and efficiency of this method compared to a purely evolutionary approach.

Keywords Data-driven methods · Complex networks · Dynamical systems · Machine learning · Evolutionary algorithms

1 Introduction

In many cases, real-world networks have similar structures [1,2] and display similar characteristics [3,4] even though they come from completely different domains of science such as biology or sociology [5–11]. However, despite these similarities, they may exhibit radically different dynamics [12–14]. Barzel et al. [15] have developed a formal universal mathematical framework to describe the dynamics of complex networks that covers a wide range of dynamical processes.

In this framework, complex dynamical networks, or simply complex networks, are a type of complex systems—that is, systems composed of many simple interconnected components that collectively display a complex and nonlinear behavior through their interactions—that consist of three main components: nodes connected to each other by edges; numerical amounts assigned to each node of the graph—called activities—that may vary over time; and a differential equation governing the variations of these activities, which depends both on the activity of the node itself and the activities of its neighbors.

Other variations of this framework have been developed and researched [16]; however, the framework by Barzel et al. [15] is the one used in this publication since we found that there are more publications based on this model than other variations. For instance, in recent years, general equations have been discovered for some useful properties of complex networks based on this model, namely their resilience [17], information flow [18] and signal propagation time [19].

Many real-world networks perfectly fit the description of complex networks. Take social networks as an example: individuals in a social network can be modeled by nodes, their affiliations can be modeled with edges, and the exchange of messages between individ-

A. Bakhtiarnia · A. Fahim (✉)· E. M. Miandoab
College of Engineering, University of Tehran, Tehran, Iran
e-mail: a.fahim@ut.ac.ir
A. Bakhtiarnia
e-mail: arian.bakhtiarnia@ut.ac.ir
E. M. Miandoab
e-mail: e.maani@ut.ac.ir

uals can be modeled by activities. Our goal is to develop a method that given data about the activities of individuals, can discover the differential equation that governs a social network, which can aid researchers in obtaining concise equations for the dynamics of social networks as well as discovering some of their properties.

This is basically a symbolic regression problem, in which our aim is to search the space of candidate mathematical expressions to find the expression that best fits our data. Various methods have been proposed for symbolic regression in the literature, many of which have been developed in recent years. Barzel et al. [20] present a symbolic regression method for complex networks by observing the response of the network to external perturbations. Unfortunately, in many networks such as social networks, it is extremely difficult to introduce isolated and adequately large perturbations. To overcome this impediment, Barzel et al. resort to use some domain-specific knowledge in their equations; however, this approach makes it impossible to automate the process and requires the manual intervention of an expert.

Sparse identification of nonlinear dynamics (SINDy) is another method of symbolic regression for dynamical systems introduced by Brunton et al. [21], which utilizes techniques from machine learning and sparse regression. This method performs an independent regression for each variable of the system—in our case, the activities of specific nodes—and discovers a separate differential equation for each variable. When the activities of different nodes are governed by the same dynamics, performing independent regressions is much more susceptible to noise and overfit that performing a single regression for the entire system, simply because in the latter case, regression would be performed on a much larger dataset. Therefore, when different nodes have the same dynamics, it is desirable for a symbolic regression algorithm to perform a single regression on the entire system to avoid discovering different governing equations for the activities of these nodes. Furthermore, SINDy relies on a library of candidate nonlinear functions, which should be given as an input to the algorithm; therefore, SINDy requires some domain-specific knowledge. Moreover, SINDy does not perform well if the nonlinear functions in the library are similar, for instance, if functions $x^{1.49}, x^{1.50}, x^{1.51}, \ldots$ are simultaneously present in the library. To solve this problem, the library of nonlinear functions in SINDy usually contains only functions with integer exponents such as $x^1, x^2, x^3, \ldots$ which results in the discovery of the Taylor series of the target function that is less concise than the function itself.

Perhaps the most commonly used symbolic regression method is genetic programming [22] which has been used for identification of nonlinear dynamical systems [23]. Although genetic programming does not have any of the aforementioned drawbacks, it is extremely slow compared to the other approaches.

Our proposed method is the parameter identification of complex network dynamics (PICND) algorithm, which combines elements of SINDy with a genetic algorithm in order to avoid these drawbacks while staying efficient. Our code is available at https://github.com/arianbakh/PICND.

2 The mathematical model of complex networks

Formally, the activity of node i is denoted by $x_i(t)$. The general form of the differential equation that governs the variations of $x_i(t)$ is shown in Eq. (1) where F and G are arbitrary nonlinear functions dictated by the nature of the system and $j \rightarrow i$ iterates over the nodes j that have an outgoing edge to node i. Here, function F captures the self-dynamics of the network which accounts for processes that depend solely on the internals of the entity represented by the node, while function G captures the impact of neighboring nodes on the dynamics which accounts for processes that depend on the interactions of entities represented by neighboring nodes. For instance, in the case of the epidemic spread of a disease between communities, if we take x_i to be the number of active cases in the i-th community, then the self-dynamics could account for the natural decrease in the active cases within a community as a result of individuals either recovering or dying as well as the natural increase in the active cases within a community where the disease is spread from infected individuals within the community to healthy individuals in the same community, and the neighbor dynamics could account for the spread of the disease a result of interaction between communities.

$$\frac{dx_i}{dt} = F(x_i) + \sum_{j \rightarrow i} A_{ij} G(x_i, x_j) \tag{1}$$

In the complex networks model, the underlying graph can be directed, undirected, weighted or unweighted. A is the adjacency matrix, and element A_{ij} of the adjacency matrix is equal to the weight of the edge from

Table 1 Common dynamic models

Dynamic model	Equation	Example(s)
Simple regulatory dynamics	$\frac{dx_i}{dt} = -x_i + \sum_{j\to i} A_{ij} \frac{x_j}{1+x_j}$	Gene regulation
Regulatory dynamics with non-integer exponents	$\frac{dx_i}{dt} = -x_i^{0.4} + \sum_{j\to i} A_{ij} \frac{x_j^{0.2}}{1+x_j^{0.2}}$	Gene regulation
Neuronal dynamics	$\frac{dx_i}{dt} = -Bx_i + C\tanh x_i + \sum_{j\to i} A_{ij} \tanh x_j$	Activation dynamics between brain regions
Population dynamics	$\frac{dx_i}{dt} = -x_i^{0.5} + \sum_{j\to i} A_{ij} x_j^{0.2}$	Birth-death processes
Epidemic dynamics	$\frac{dx_i}{dt} = -x_i + \sum_{j\to i} A_{ij}(1-x_i)x_j$	Social networks, Epidemic spread of disease
Mutualistic dynamics	$\frac{dx_i}{dt} = x_i(1-x_i^2) + \sum_{j\to i} A_{ij} \frac{x_i x_j}{1+x_j}$	Plat-pollinator relationships in ecological networks

node j to node i. If there is no edge from node j to node i, then $A_{ij} = 0$. If the graph is undirected, we assume edges in both directions exist; in other words, the adjacency matrix is symmetric and $A_{ij} = A_{ji}$. If the graph is unweighted, then $A_{ij} = 1$ if there is an edge from node j to node i. As the self-dynamics are captured in a separate term within the differential equations, we assume there are no loops in the graph, meaning $A_{ii} = 0$.

If the impact of variables $x_i(t)$ and $x_j(t)$ is separable within the function G, the governing differential equation can take the form of Eq. (2). $x_i(t)$ and $x_j(t)$ are separable in all of the models that are used in this paper. It is important to note that the equations for the resilience [17], information flow [18] and signal propagation time [19], properties of complex networks all require the governing equation of the complex network to be of the separated form. Table 1 presents several governing differential equations that are common in biological, neurological and social complex networks [19].

$$\frac{dx_i}{dt} = M_0(x_i) + \sum_{j\to i} A_{ij} M_1(x_i) M_2(x_j) \tag{2}$$

Typically, data are available in the form of a series of snapshots containing the state of the complex network—that is, the activities of nodes—at different time-frames. Here, as an example, we generate a dataset by simulating epidemic dynamics on a simple unweighted three-node network shown in Fig. 1. The adjacency matrix for this network is shown in Eq. (3). The simulation starts with initial values 0, 0 and 1 for the activities of the three nodes, respectively. At each step, the activities of the next snapshot are calculated by adding the activities of the previous snapshot to the product of Δt and the derivative given by the epidemic dynamics equation in Table 1.

$$\boldsymbol{A} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix} \tag{3}$$

The scale and unit of the variables in these equations depend on the physical interpretation of the system. For instance, if we take 100 snapshots of a social network over a period of one week, and define $t = 0$ as the beginning of that week and $t = 1$ as the end, then Δt would be equal to 0.01 in our equations and about 100 minutes in the real world. Similarly, the values of the activities could be arbitrarily scaled as it would only change the coefficients of the terms in the differential equation. For example, if the activity x_i denotes the number of messages sent by node i in a social network during the interval Δt, then $x_i = 1$ could be defined as 50 messages sent during that interval, or however else we desire.

If the data do not contain the actual values of the derivatives and the derivatives are calculated by dividing Δx_i by Δt—which is the case for most datasets—then the number of snapshots should be sufficient in a way that $\Delta x_i/\Delta t$ resembles dx_i/dt. The value of the derivative dx_2/dt of the second node in our example and the components of the derivative—that is, the self-dynamics and the effect of neighbors—are depicted in Figs. 2, 3 and 4 with different numbers of frames sam-

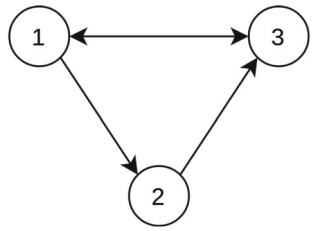

Fig. 1 Example of a simple unweighted three-node network

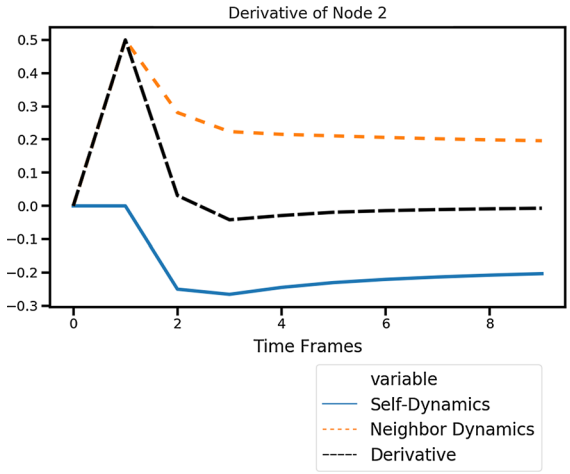

Fig. 2 Values of the derivative (and its components) for the activity of node 2 in the simulation of epidemic dynamics on the graph of Fig. 1 with 10 snapshots. The number of snapshots is not sufficient, and thus, the plot diverges from actual dynamics

pled from the same duration. The simulations with 1000 frames and 100 frames have almost identical plots; however, if the number of frames falls to 10, the plot diverges from the actual dynamics.

It should be noted that we are not after solving the differential equation to obtain the mathematical expression for activity $x_i(t)$ itself, our goal is only to discover the differential equation that governs the system, that is, to find the mathematical expression for dx_i/dt. This is due to the fact that many properties of complex networks—including but not limited to impact [15], stability [15], resilience [17], information flow [18] and propagation time [19]—are defined based on the responses of nodes to perturbations in other nodes, which means they deal with the variations and fluctuations in activities rather than the values of activities themselves; thus, they can be derived directly from the underlying governing differential equation and do not require the mathematical expression for $x_i(t)$ to be known. In general, solving such differential equations

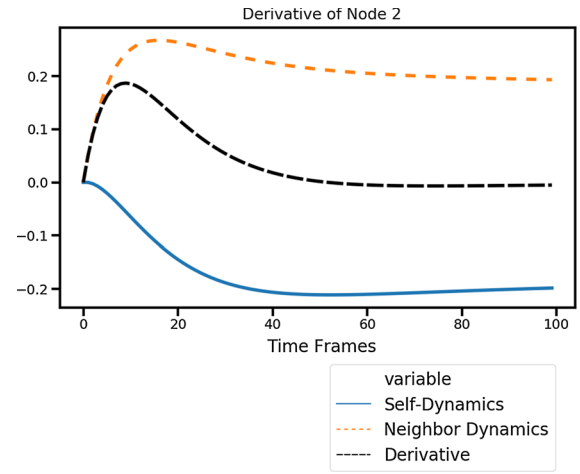

Fig. 3 Values of the derivative (and its components) for the activity of node 2 in the simulation of epidemic dynamics on the graph of Fig. 1 with 100 snapshots. The number of snapshots is sufficient, and thus, the plot resembles actual dynamics

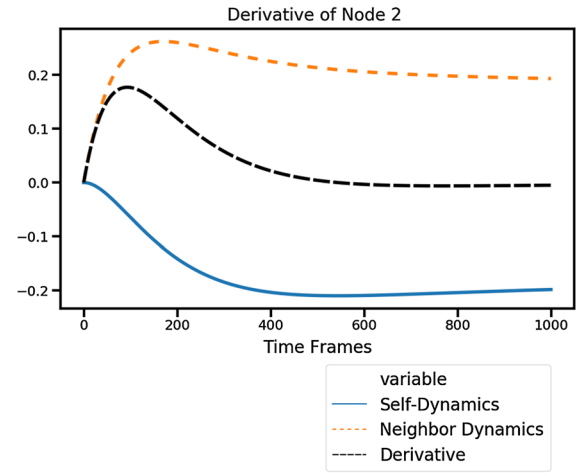

Fig. 4 Values of the derivative (and its components) for the activity of node 2 in the simulation of epidemic dynamics on the graph of Fig. 1 with 1000 snapshots

can be extremely difficult, and in most cases an explicit formula may not even exist.

As an example, we briefly present how the "propagation time" property of a complex network can be calculated solely based on the structure of the network and the governing differential equation. Propagation time, which is formally defined in Eq. 4, represents the time when the activity of node i has reached an η-fraction of its final response to the signal from node j—that is, perturbation Δx_j. A commonly used value for η is 1/2. The formula for the propagation time property of com-

plex networks, derived by Hens et al. [19], is shown in Eq. 5 where $\mathcal{L}(j \to i)$ is a predictive metric that is proportional to the actual propagation time $T(j \to i)$, $\Pi(j \to i)$ is a shortest path from source node j to target node i, p iterates over the nodes present in the shortest path except j itself, $S_i = \sum_{j \to i} A_{ij}$ is the weighted in-degree of node i and θ is solely determined by the system's dynamics based on Eq. (6). The parameter $\Gamma(0)$ in Eq. (6) is the leading exponent of the Hahn series expansion in Eq. (7), in which Y and R are defined by Eq. (8) and Eq. (9), respectively, where functions M_0 and M_1 are the same functions introduced in Eq. 2. The Hahn series expansion depicted in Eq. (7) is a generalization of Taylor series expansion to include real exponents; thus, $\Gamma(n)$ represents a sequence of ascending real numbers.

$$\Delta x_i(T(j \to i)) = \eta \Delta x_i(t \to \infty) \tag{4}$$

$$T(j \to i) \propto \mathcal{L}(j \to i) = \min_{\Pi(j \to i)} \left\{ \sum_{p \in \Pi(j \to i), p \neq j} S_p^{\theta} \right\} \tag{5}$$

$$\theta = -2 - \Gamma(0) \tag{6}$$

$$Y(R^{-1}(x)) = \sum_{n=0}^{\infty} C_n x^{\Gamma(n)} \tag{7}$$

$$Y(x) = \left(\frac{d(M_1 R)}{dx} \right)^{-1} \tag{8}$$

$$R(x) = -\frac{M_1(x)}{M_0(X)} \tag{9}$$

3 Proposed methodology

Our algorithm takes the network structure and the time-series of the activity of each node in the network as input, and consists of two steps that are repeatedly executed until convergence. The first step guesses the exponents of the terms in the governing differential equation using a genetic algorithm (GA). The second step then determines the best possible coefficients for the terms given by the previous step based on an approach similar to the SINDy algorithm.

In this way, the GA step of our algorithm has to deal with only the exponents of the terms, which is half of the parameters compared to a purely genetic approach where both the exponents and the coefficients of the governing differential equation are evolved, and the coefficients of the terms are discovered in the SINDy step which is quite efficient. This fact is the key to the efficiency of our algorithm.

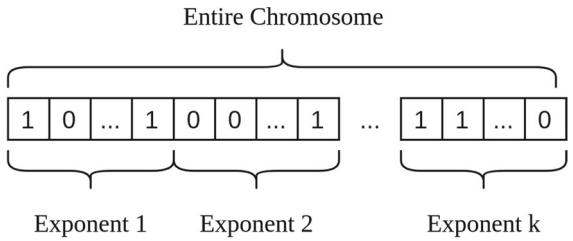

Fig. 5 Structure of the chromosome in the GA step of the PICND algorithm

3.1 The GA step

The goal of this step is to determine the exponents of the terms in the governing differential equation using a simple genetic algorithm where a population P of individuals produce offsprings O in each iteration of the algorithm. Each individual in the population corresponds to a specific differential equation. Offsprings will take the place of other individuals in the population if they have a higher fitness—that is, a more suitable differential equation.

The structure of the chromosome of each individual is shown in Fig. 5. The chromosome consists of k sections of c bits, where k is the maximum number of terms in the differential equation, and the exponent of each term is calculated using the binary number in the corresponding section of the chromosome. Keep in mind that some of the coefficients of the terms may be zero, therefore the actual number of terms may be less than k. The search space for each exponent is constrained to the range $(l_{min}, l_{max}]$ and is divided into 2^c parts; consequently, if the value of the binary number in i-th section is b_i, it means the exponent of the i-th term in the differential equation is equal to $(b_i + 1)(l_{max} - l_{min})/2^c$. The chromosomes of the individuals in the initial population are completely random.

The fitness of each individual in the population is determined based on the mean-squared error (MSE) of the differential equation of that individual compared to the input data, as seen in Eq. (10). However, the coefficients of the terms in the differential equations are required in order to calculate the MSE; therefore, this operation and all subsequent operations of the GA step that depend on the fitness value are postponed until

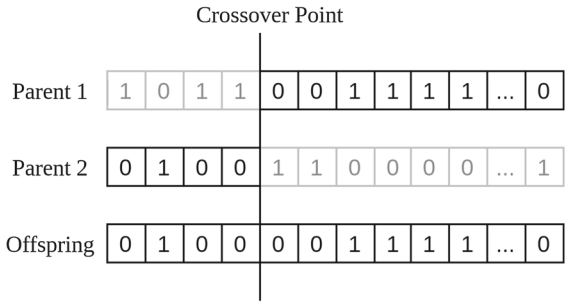

Fig. 6 An example of the crossover operation

the next step—that is, the SINDy step—is executed and the coefficients are obtained.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(\hat{\dot{x}}_i - \dot{x}_i)^2 \tag{10}$$

Each offspring is produced through a crossover between the chromosomes of two individuals in the population, which are randomly selected based on their fitness; the higher their fitness, the more likely they are to be selected. During the crossover operation, a crossover point in the range $(1, k * c)$ is randomly selected, and the chromosome of the offspring is acquired by concatenating the portion of the chromosome of the second parent that comes before the crossover point, and the portion of the chromosome of the first parent that comes after the crossover point. An example of this operation is shown in Fig. 6. A crossover between an individual and itself results in the same individual; therefore, if the same individual is selected as both of the participants, crossover will be retried.

After the crossover phase, each bit in the chromosome of the offspring has a chance to be mutated with probability p_m, in which case the value of the bit is flipped from 1 to 0 or 0 to 1.

3.2 The SINDy step

The goal of this step is to determine the coefficients of the terms in the governing differential equation for each individual in the population. First, we describe the SINDy algorithm, and then, we explain our modifications to the SINDy algorithm.

SINDy organizes the input time-series into the $\boldsymbol{X}$ matrix presented in Eq. (11). In this matrix, $x_i(t_j)$ denotes the value of the i-th variable—in our case, the activity of node i—at time t_j. Thus, the i-th column of this matrix corresponds to the time-series of variable x_i, and the j-th row corresponds to the state of the system at time t_j. Time progresses from top to bottom, and SINDy assumes $t_{j+1} - t_j = \Delta t$ is constant for all j; therefore, we have to interpolate the state of the system if that is not the case. Typically, $\boldsymbol{X}$ is normalized by subtracting the mean of each column from the entries of that column and then dividing the entries by the standard deviation of the column.

$$\boldsymbol{X} = \begin{bmatrix} x_1(t_1) & x_2(t_1) & \dots & x_n(t_1) \\ x_1(t_2) & x_2(t_2) & \dots & x_n(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ x_1(t_m) & x_2(t_m) & \dots & x_n(t_m) \end{bmatrix} \tag{11}$$

The $\dot{\boldsymbol{X}}$ matrix can be calculated from the $\boldsymbol{X}$ matrix by subtracting the previous row from each row and dividing the result by Δt, as shown in Eq. (12). In this matrix $\dot{x}_i(t_j) = (x_i(t_{j+1}) - x_i(t_j))/\Delta t$ denotes the derivative of variable x_i at time t_j. This method of calculating the derivatives could potentially introduce a great deal of error unless Δt has a small value in the context of the dynamical system, for instance a few milliseconds for physical systems and a few minutes for social networks. For certain systems, sensors can measure derivatives of the variables as well—for instance, speed and acceleration—in which case the actual measurements for the derivatives should be used in $\dot{\boldsymbol{X}}$ instead.

$$\dot{\boldsymbol{X}} = \begin{bmatrix} \dot{x}_1(t_1) & \dot{x}_2(t_1) & \dots & \dot{x}_n(t_1) \\ \dot{x}_1(t_2) & \dot{x}_2(t_2) & \dots & \dot{x}_n(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ \dot{x}_1(t_{m-1}) & \dot{x}_2(t_{m-1}) & \dots & \dot{x}_n(t_{m-1}) \end{bmatrix} \tag{12}$$

SINDy then constructs a library of nonlinear functions of the $\boldsymbol{X}$ matrix, which are organized into the $\Theta(\boldsymbol{X})$ matrix as shown in Eq. (13). In Eq. (13), $\boldsymbol{X}^{P_k}$ contains all of the polynomials of order k; as an example, $\boldsymbol{X}^{P_2}$ is shown in Eq. (14). The list of nonlinear functions in this library is given by the user, and depends on the problem domain. It is noteworthy that the last row of $\boldsymbol{X}$ is omitted before the construction of $\Theta(\boldsymbol{X})$ due to the fact that $\Theta(\boldsymbol{X})$ is used in Eq. (16) and therefore must have the same number of rows as $\dot{\boldsymbol{X}}$, and because the derivative at t_m is unknown and therefore not present in the $\dot{\boldsymbol{X}}$ matrix, $\dot{\boldsymbol{X}}$ has only $m - 1$ rows.

$$\Theta(\boldsymbol{X}) = \begin{bmatrix} | & | & | & | & & | \\ \mathbf{1} & \boldsymbol{X} & \boldsymbol{X}^{P_2} & \boldsymbol{X}^{P_3} & \dots & sin(\boldsymbol{X}) & \dots \\ | & | & | & | & & | \end{bmatrix} \tag{13}$$

$$\boldsymbol{X}^{P_2} = \begin{bmatrix} x_1^2(t_1) & \dots & x_n^2(t_1) & x_1(t_1)x_2(t_1) & \dots \\ x_1^2(t_2) & \dots & x_n^2(t_2) & x_1(t_2)x_2(t_2) & \dots \\ \vdots & \ddots & \vdots & \vdots & \ddots \\ x_1^2(t_m) & \dots & x_n^2(t_m) & x_1(t_m)x_2(t_m) & \dots \end{bmatrix} \quad (14)$$

Figure 7 depicts the schematic of the SINDy algorithm. The goal of SINDy is to discover the coefficient matrix $\boldsymbol{\Xi}$ shown in Eq. (15) where $\boldsymbol{\xi}_i$ denotes the coefficient vector for the nonlinear functions of variable x_i, in a manner that is as sparse as possible, which means most entries in the matrix are zero. To achieve this, SINDy first finds a least-squares (LS) solution for $\boldsymbol{\Xi}$ based on Eq. (16), and then sets all of entries of $\boldsymbol{\Xi}$ that have an absolute value less than a hyperparameter λ to zero, and subsequently finds another least-squares solution with the remaining elements of $\boldsymbol{\Xi}$, and repeats this process until no other entries of the $\boldsymbol{\Xi}$ matrix are set to zero. Reference [24] discusses the conditions and the rate of convergence for the SINDy algorithm in detail.

A hyperparameter optimization problem should be solved in order to obtain the optimum value for λ and select the best model. As a separate least-squares solution should be found for each candidate λ, this model selection phase significantly slows down the SINDy algorithm. The authors of the original paper discuss the best methods for model selection in a subsequent publication [25].

$$\boldsymbol{\Xi} = \left[\boldsymbol{\xi}_1\ \boldsymbol{\xi}_2 \dots \boldsymbol{\xi}_n\right] \quad (15)$$

$$\dot{\boldsymbol{X}} = \Theta(\boldsymbol{X})\boldsymbol{\Xi} \quad (16)$$

We slightly modify the SINDy algorithm for our purposes. First, we define vector $\boldsymbol{X}_i$ presented in Eq. (17) as the time-series for the activities of node i, which is basically the i-th column of matrix $\boldsymbol{X}$, and vector $\dot{\boldsymbol{X}}_i$ presented in Eq. (18) as the derivatives of the activities of node i, which is the same as the i-th column of matrix $\dot{\boldsymbol{X}}$.

$$\boldsymbol{X}_i = \begin{bmatrix} x_i(t_1) \\ x_i(t_2) \\ \vdots \\ x_i(t_m) \end{bmatrix} \quad (17)$$

$$\dot{\boldsymbol{X}}_i = \begin{bmatrix} \dot{x}_i(t_1) \\ \dot{x}_i(t_2) \\ \vdots \\ \dot{x}_i(t_{m-1}) \end{bmatrix} \quad (18)$$

Then, we create a separate library of nonlinear functions $\Theta(\boldsymbol{X}_i)$ for each of the nodes as shown in Eq. (19), where $p_1, \dots p_5$ are the exponents of the terms in the governing differential equation of the individual determined by the GA step.

The reason behind customizing the library for each node is that we want the dynamics of all of the nodes in the network to obey the same governing differential equation, in other words, we want $\boldsymbol{\xi}_i$ to be equal to the same $\boldsymbol{\xi}$ for all i. In order to achieve this, we came up with the idea to stack all columns of the matrix $\dot{\boldsymbol{X}}$ on top of each other, and stack n copies of the matrix $\Theta(\boldsymbol{X})$ on top of each other, and find a single least-squares solution for all of the nodes at once. However, based on the form of Eqs (1) and (2) and the fact that the activities of nodes are located at different columns, we can see that even if all of the nodes have exactly the same dynamics and the same edges, if we used SINDy to determine their coefficient vectors using a shared library of nonlinear functions, their coefficient vectors would be permutations of each other, not equal. Consequently, a single $\boldsymbol{\xi}$ cannot be discovered with our stacking method if n copies of the same library of nonlinear functions are stacked on top of each other, and thus, the library should be customized for each node and aligned with the libraries of other nodes.

Note that it is not possible to use permutations of the columns of the matrix $\Theta(\boldsymbol{X})$ with our stacking method due to the fact that nodes may have a different number of neighbors, and the edges that connect them may have different weights; therefore, each nonlinear function that is a part of the G function in Eq. (1) should be included in the library in the form of a summation, just like the third, fourth and fifth columns of Eq. (19).

The nonlinear functions in the library $\Theta(\boldsymbol{X}_i)$ are designed to take into account all of the dynamic models similar to ones described in sect. 4.1; however, this library can be easily extended to cover other arbitrary dynamic models.

$$\Theta(\boldsymbol{X}_i) = \begin{bmatrix} | & | & | & | \\ \mathbf{1}\ \boldsymbol{X}_i^{p_1} & \sum_{j\to i} A_{ij}\boldsymbol{X}_i^{p_2}\boldsymbol{X}_j^{p_3} & \sum_{j\to i} A_{ij}\boldsymbol{X}_j^{p_4} & \\ | & | & | & | \\ & & | & \\ & & \sum_{j\to i} A_{ij}\left(1 - \frac{1}{\boldsymbol{X}_j^{p_5}}\right) & \\ & & | & \end{bmatrix} \quad (19)$$

Finally, we stack $\dot{\boldsymbol{X}}_i$s on top of each other to obtain the matrix $\dot{\boldsymbol{X}}$ seen in Eq. (20), and stack $\Theta(\boldsymbol{X}_i)$s on top of each other to obtain matrix $\Theta(\boldsymbol{X})$ seen in Eq. (21), and find a least-squares solution for $\boldsymbol{\Xi}$ based on

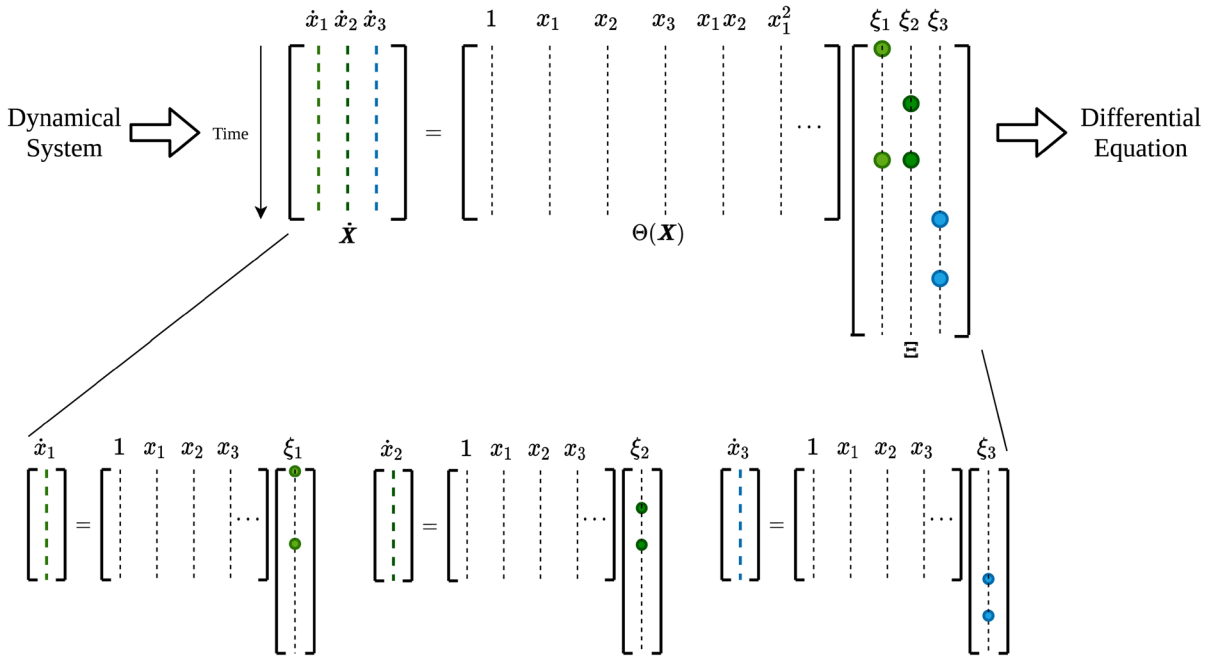

Fig. 7 Schematic of the SINDy algorithm

Eq. (16) to obtain a single coefficient vector $\boldsymbol{\xi}$, which is passed on to the GA step in order to calculate the fitness of the corresponding individual in the population. We do not need to perform the sparse regression in SINDy here because our library is small and consists of only five columns, which significantly speeds up our algorithm by skipping the hyperparameter optimization phase; however, if our aim is to take into account many other possible dynamic models, and our library becomes larger as a result, we would need to perform the sparse regression as well.

$$\dot{\boldsymbol{X}} = \begin{bmatrix} \dot{\boldsymbol{X}}_1 \\ \dot{\boldsymbol{X}}_2 \\ \vdots \\ \dot{\boldsymbol{X}}_n \end{bmatrix} \tag{20}$$

$$\Theta(\boldsymbol{X}) = \begin{bmatrix} — \ \Theta(\boldsymbol{X}_1) \ — \\ — \ \Theta(\boldsymbol{X}_2) \ — \\ \vdots \\ — \ \Theta(\boldsymbol{X}_n) \ — \end{bmatrix} \tag{21}$$

If the number of nodes is huge, it is possible to use only a subset of nodes in the stacking method, preferably randomly sampled in each iteration; however, this negatively impacts the convergence rate of the algorithm.

3.3 Combining the steps

The aforementioned steps are repeatedly executed until the termination condition is satisfied, which happens when there are no changes in the fitness of the fittest individual in the population for n_i consecutive iterations. Algorithm 1 shows a high-level pseudocode for the PICND algorithm, and Fig. 8 depicts its schematic.

3.4 Drawbacks of PICND

In some cases, it is possible that PICND is not able to find the dynamics of the network. In general, PICND will fail if the input data does not adhere to the complex networks model discussed in Sect. 2, for instance, if the dynamics of the nodes are radically different. In this case, an interesting research direction would be to somehow cluster the nodes with the same dynamics together and find a complex subnetwork, or find a contracted network where the contraction of some of the edges in the network could decrease the difference between the dynamics of the nodes in the network.

Another situation where PICND may fail is when the dynamics of the network varies over time, which may occur in case of changes to the rules, regulations

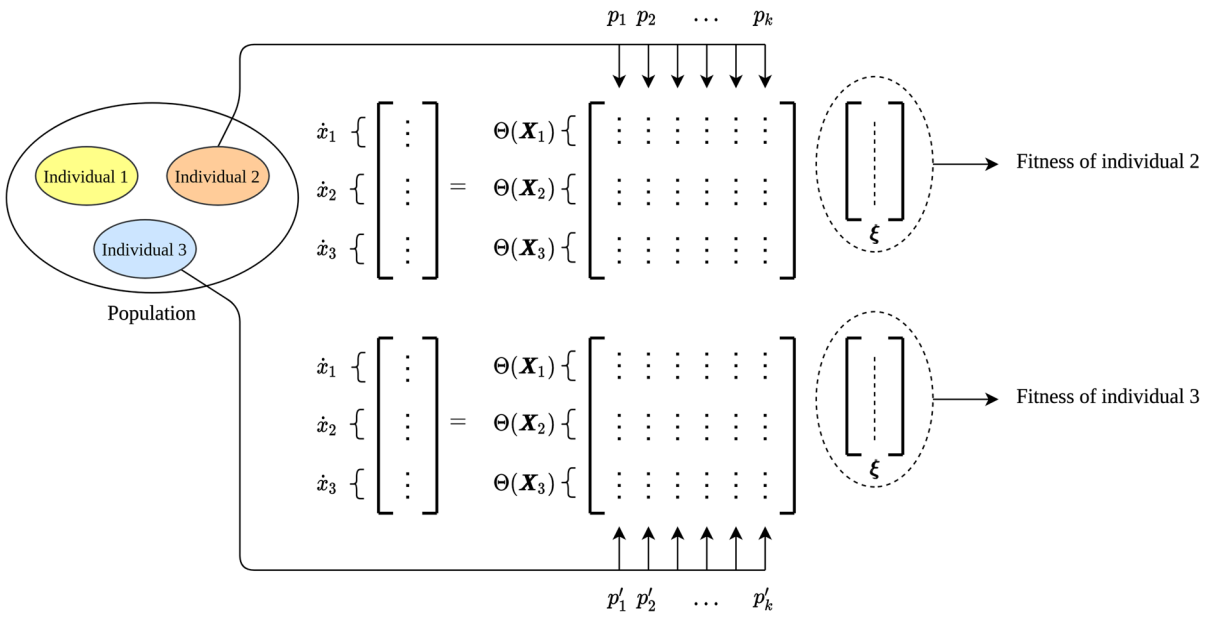

Fig. 8 Schematic of the PICND algorithm. Exponents of the terms in the differential equation—shown as $p_1, \ldots, p_k$ and $p'_1, \ldots, p'_k$—are derived from each individual's chromosome and fed into the library of nonlinear functions in our modification to the SINDy algorithm. Then, a single coefficient vector is calculated for each individual based on our stacking method. Finally, the coefficient vector of each individual along with the exponents derived from its chromosome is used to derive its fitness, which is then used in the selection process that determines the population of the subsequent iteration

Algorithm 1 PICND

```
population ← RandomPopulation()
for individual in population do
    individual.coefficients ← SINDyStep(individual.exponents)
    individual.fitness ← CalculateFitness(individual.exponents, individual.coefficients)
end for
while not terminationCondition do
    offsprings ← Mutation(Crossover(population))
    for individual in offsprings do
        individual.coefficients ← SINDyStep(individual.exponents)
        individual.fitness ← CalculateFitness(individual.exponents, individual.coefficients)
    end for
    population ← SelectKFittest(population + offsprings)
end while
```

and policies of social networks or financial markets, although this problem can be mitigated by constraining the algorithm to short periods of time. However, even in these cases, PICND can be useful, for instance it can be used in a sliding window manner to detect specific windows of time where dominant epidemic dynamics are present in a social network.

As previously mentioned, the library of nonlinear functions used here can only discover dynamic models similar to ones described in sect. 4.1; however, it can be easily modified to include other common dynamic models such as neuronal dynamics and mutualistic dynamics [19] presented in Table 1. Indeed, Mangan et al. [26] have shown that the SINDy algorithm can determine more general forms of dynamics with slight modifications to the library of nonlinear functions. In doing so, in order to avoid convergence problems in the genetic algorithm, it is important to design the interpretation of the chromosome in a way that no two different chromosomes describe the same equation. For instance, if the self-dynamics part of the equation is in the form of $x_i^{k_1} + x_i^{k_2}$ where $k_1 \neq k_2$, we can take k_1 to be equal to the value of the first section of the chromosome v_1, and take k_2 to be equal to the value of the second section v_2. In this case, $v_1 = 1, v_2 = 2$ and $v_1 = 2, v_1 = 1$ both describe the same dynamics but result in different chromosomes. A workaround for this problem is to take

k_1 to be equal to v_1 and k_2 to be equal to $v_1 + v_2 + \epsilon$, thus ensuring k_2 is always greater than k_1.

Even more complex dynamics can be included with minor tweaks. In the case that node-independent nonlinear functions such as $\sin(t)$ are suspected to exist in the governing differential equation, the corresponding vector can be added to the library of every node. For coupled systems such as FitzHugh–Nagumo neuronal network model [27] and delay-coupled complex neuronal networks [28] where the F part of Eq. (1) for $\dot{x}_i$—and similarly $\dot{y}_i$—can depend on both x_i and y_i, linear and nonlinear functions of y_i can be added to the library of the node with activity x_i.

4 Experimental setup

In our experiments, we use the same methodology as Hens et al. [19], where we simulate each dynamic model on one or more empirical networks that are relevant to that specific dynamic model. We simulate four of the six common dynamic models introduced in Table 1, as shown in Fig. 9. All of the models are simulated for 100 snapshots with $\Delta t = 0.01$. Then, in each of these cases, we compare the accuracy and efficiency of the PICND algorithm in the task of identifying the governing differential equation, with a purely genetic approach. We repeat each experiment three times and record the average value of the discovered exponents, discovered coefficients, MSE and number of iterations.

4.1 Dynamic models

4.1.1 Regulatory dynamic models (R1 and R2)

These dynamic models represent gene regulation using the Michaelis–Menten model with different exponents for the self-dynamics and the regulating Hill function [29]. Equations (22) and (23) show the governing differential equations for the R1 and R2 dynamics models, respectively.

$$\frac{dx_i}{dt} = -x_i + \sum_{j \to i} A_{ij} \frac{x_j}{1 + x_j} \tag{22}$$

$$\frac{dx_i}{dt} = -x_i^{0.4} + \sum_{j \to i} A_{ij} \frac{x_j^{0.2}}{1 + x_j^{0.2}} \tag{23}$$

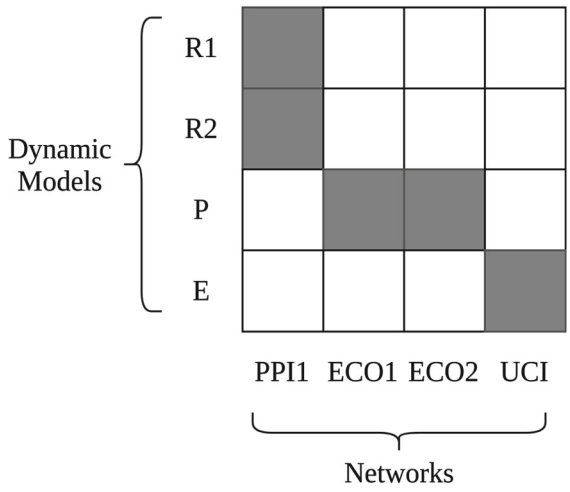

Fig. 9 Summary of simulations. The cells with gray color indicate the dynamic model on the corresponding row was simulated on the network on the corresponding column

4.1.2 Population dynamic model (P)

This dynamic model represents population dynamics through birth-death processes [30]. Eq. (24) is the governing differential equation for this dynamic model.

$$\frac{dx_i}{dt} = -x_i^{0.5} + \sum_{j \to i} A_{ij} x_j^{0.2} \tag{24}$$

4.1.3 Epidemic dynamic model (E)

This dynamic model captures the dynamics of epidemic spreading occurring in real-world networks such as the spread of viral messages in social networks [31]. Equation (25) is the governing differential equation for this dynamic model.

$$\frac{dx_i}{dt} = -x_i + \sum_{j \to i} A_{ij} (1 - x_i) x_j \tag{25}$$

4.2 Networks

4.2.1 Yeast protein–protein interaction network (PPI1)

This network consists of 6574 nodes in total, representing the chemical interactions between proteins in yeast [32]. Due to the large size of the network, the simulation is performed on an induced subgraph of 20 nodes selected via random walk (RW) sampling,

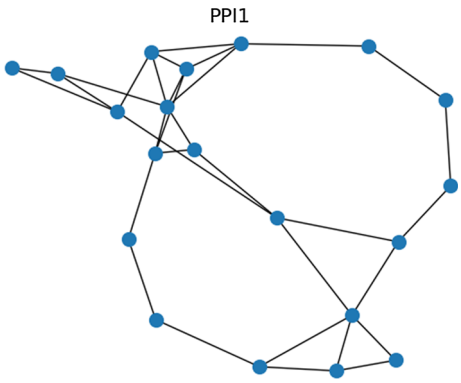

Fig. 10 Network sampled from the yeast protein–protein interaction network

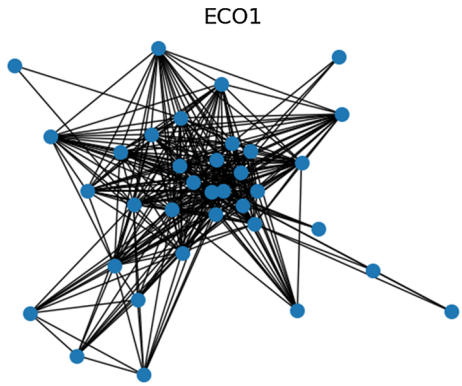

Fig. 11 Pollinator–pollinator indirect interaction network (ECO1)

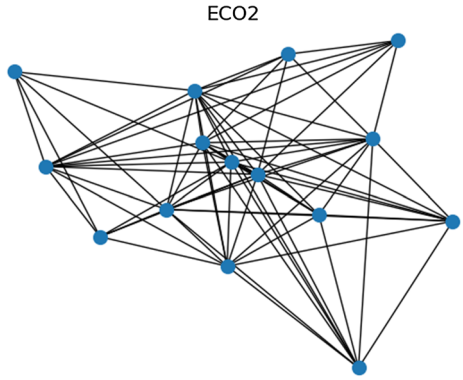

Fig. 12 Plant–plant indirect interaction network (ECO2)

which best preserves graph patterns and characteristics among common sampling techniques for large graphs [33]. Figure 10 shows the network resulting from this sampling operation.

4.2.2 Indirect interaction networks derived from a plant–pollinator interaction network (ECO1 and ECO2)

The plant–pollinator interaction network from Dicks et al. [34] represents the interactions between 16 species of plants and 36 species of pollinators in Shelfanger, Norfolk, UK. This network is a bipartite graph linking plants to their pollinators. When two different plants are visited by the same pollinator, they increase each other's population indirectly by increasing the population of their shared pollinator. Similarly, pollinators sharing the same plants benefit each other indirectly. Therefore, two networks can be constructed based on the plant–pollinator network, namely the pollinator–pollinator and plant–plant indirect interaction networks.

Equation (26) shows how the pollinator–pollinator indirect interaction network (ECO1) is constructed, and Eq. (27) does the same for the plant–plant network (ECO2). In these equations, the adjacency matrices of plant–pollinator, pollinator–pollinator and plant–plant networks are represented by M, B and A, respectively. This method can potentially lead to networks that have more than one connected component, in which case the largest connected component is used instead. The resulting indirect interaction networks are shown in Figs. 11 and 12.

$$B_{kl} = \sum_{i=1}^{16} \frac{M_{ik} M_{il}}{\sum_s M_{is}} \tag{26}$$

$$A_{ij} = \sum_{k=1}^{36} \frac{M_{ik} M_{jk}}{\sum_s M_{sk}} \tag{27}$$

4.2.3 UCIOnline social network (UCI)

This network is derived from the data set of the University of California Irvine online instant messaging service, which captures the messages exchanged between 1899 individuals during a 7-month period [35]. In the constructed network, there is an edge between the nodes representing two individuals in the social network if and only if they have exchanged at least one message during this period; therefore, this is an undirected and unweighted network. Similar to the PPI1

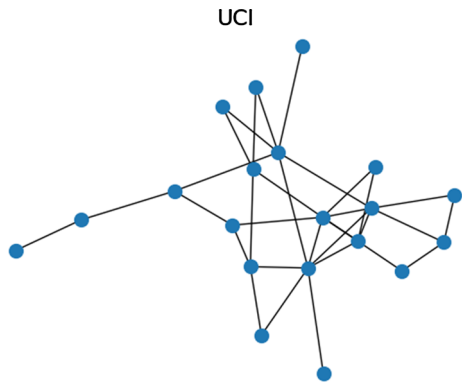

Fig. 13 Network sampled from the UCIOnline social network

networks, due to the large size of the network, the simulation is performed on an induced subgraph of 20 nodes selected via random walk (RW) sampling. Figure 13 depicts the network resulting from this sampling process.

4.3 Algorithms

In order to properly compare both algorithms with respect to their efficiency, instead of using the usual termination condition—that is, terminating the algorithm after the fitness of the fittest individual in the population does not improve for n_i consecutive iterations—we let both algorithms run for the same amount of time—120 seconds—on the same hardware—Intel Core i7-4700HQ CPU 2.40GHz.

4.3.1 Parameter identification of complex network dynamics (PICND)

This is our proposed method presented in sect. 3. The values used for the parameters of the PICND algorithm are presented in Table 2.

4.3.2 Purely genetic approach (GA)

This method solely utilizes a genetic algorithm to derive the governing differential equation, meaning that both the coefficients and the exponents of the differential equation are included in the chromosome and are discovered by a genetic algorithm. We set the parameters of the GA algorithm to exactly the same values as the PICND algorithm, with the exception of the number of the sections in the chromosome $k = 10$ in order to allow the chromosome to include both the exponents and the coefficients of the terms in the governing differential equation. In addition, we restrict the range of the coefficients to $[-1, 1]$ for the GA algorithm.

5 Results

The results of our experiments are summarized in Tables 3–7. In these tables, both the exponents and the coefficients are rounded to two decimal places. Even though both of the algorithms run for the same amount of time, and even though we narrow down the search space of the coefficients to the range $[-1, 1]$ for the GA algorithm while providing no hints about the coefficients to the PICND algorithm, GA completely fails to find the correct dynamics in all cases, while PICND correctly discovers the governing differential equation accurate to two decimal places in each and every case. We also observe that the MSE of the fittest individual in the PICND algorithm is always several orders of magnitude less than the MSE of the fittest individual in the GA algorithm.

Given enough execution time, GA would be able to search the entire space of possibilities and thus accurately discover the underlying dynamics as well; however, it is evident from these experiments that GA would require much more time than PICND to achieve the same level of accuracy.

Figures 14–18 display the descent of the mean-squared error over the iterations of the PICND algorithm as well as the GA algorithm on a logarithmic scale. From these figures, it is evident that although each iteration of the GA algorithm is more efficient than the PICND algorithm, and thus GA can go through more iterations in the same amount of time, overall the error of the PICND algorithm decreases much more rapidly than the error of the GA algorithm; therefore, PICND can achieve significantly better results in the same amount of time. This fact showcases how efficient PICND is compared to a purely genetic approach.

6 Conclusion

We have shown that the proposed methodology in this paper, namely the PICND algorithm, is an efficient method for discovering the equations of the underlying dynamics of complex dynamical networks such as social networks. As previously discussed, PICND does

Table 2 Values used for the PICND algorithm parameters

Parameter	Description of parameter	Value
$\|P\|$	Number of individuals in the population	100
$\|O\|$	Number of offsprings produced in each iteration	10
k	Number of the sections in the chromosome	5
c	Number of bits per section in the chromosome	12
$(l_{min}, l_{max}]$	Range of the search space for each exponent	(0, 2]
p_m	Mutation probability	0.1

Table 3 Comparison of the accuracy of PICND vs. GA for R1 dynamic model on PPI1 network

Algorithm	Discovered diff. Eq.	MSE	Iterations
PICND	$-1.00x_i^{1.00} + 1.00\sum_{j\to i} A_{ij}\frac{x_j^{1.00}}{1+x_j^{1.00}}$	2.44×10^{-11}	1501
GA	$0.25 - 0.92x_i^{1.00} - 0.01\sum_{j\to i} A_{ij}x_i^{0.07}x_j^{0.13} + 0.16\sum_{j\to i} A_{ij}x_j^{0.02} + 0.85\sum_{j\to i} A_{ij}\frac{x_j^{0.34}}{1+x_j^{0.34x}}$	7.09×10^{2}	4555

Table 4 Comparison of the accuracy of PICND vs. GA for R2 dynamic model on PPI1 network

Algorithm	Discovered diff. Eq.	MSE	Iterations
PICND	$-1.00x_i^{0.40} + 1.00\sum_{j\to i} A_{ij}\frac{x_j^{0.20}}{1+x_j^{0.20}}$	7.43×10^{-10}	1785
GA	$-0.26 - 0.95x_i^{0.43} - 0.03\sum_{j\to i} A_{ij}x_i^{0.01}x_j^{0.12} + 0.88\sum_{j\to i} A_{ij}x_j^{0.04} - 0.826\sum_{j\to i} A_{ij}\frac{x_j^{0.86}}{1+x_j^{0.86}}$	8.98×10^{-1}	6378

Table 5 Comparison of the accuracy of PICND vs. GA for P dynamic model on ECO1 network

Algorithm	Discovered diff. Eq.	MSE	Iterations
PICND	$-1.00x_i^{0.50} + 1.00\sum_{j\to i} A_{ij}x_j^{0.20}$	1.39×10^{-7}	276
GA	$-1.00 - 0.84x_i^{0.34} - 0.01\sum_{j\to i} A_{ij}x_i^{0.16}x_j^{0.04} + 0.56\sum_{j\to i} A_{ij}x_j^{0.25} + 0.75\sum_{j\to i} A_{ij}\frac{x_j^{0.77}}{1+x_j^{0.77}}$	6.03×10^{-1}	693

Table 6 Comparison of the accuracy of PICND vs. GA for P dynamic model on ECO2 network

Algorithm	Discovered diff. Eq.	MSE	Iterations
PICND	$-1.00x_i^{0.50} + 1.00\sum_{j\to i} A_{ij}x_j^{0.20}$	6.25×10^{-11}	2515
GA	$-0.22 - 0.94x_i^{0.56} - 0.12\sum_{j\to i} A_{ij}x_i^{0.02}x_j^{0.10} + 0.58\sum_{j\to i} A_{ij}x_j^{0.27} + 0.76\sum_{j\to i} A_{ij}\frac{x_j^{1.90}}{1+x_j^{1.90}}$	8.21×10^{-2}	3025

Table 7 Comparison of the accuracy of PICND vs. GA for E dynamic model on UCI network

Algorithm	Discovered diff. Eq.	MSE	Iterations
PICND	$-1.00x_i^{1.00} - 1.00\sum_{j\to i} A_{ij}x_i^{1.00}x_j^{1.00} + 1.00\sum_{j\to i} A_{ij}x_j^{1.00}$	3.78×10^{-11}	1489
GA	$-0.29 + 0.61x_i^{0.92} + 0.26\sum_{j\to i} A_{ij}x_i^{0.61}x_j^{1.63} - 0.20\sum_{j\to i} A_{ij}x_j^{0.73} - 0.06\sum_{j\to i} A_{ij}\frac{x_j^{1.66}}{1+x_j^{1.66}}$	7.66×10^{-4}	4721

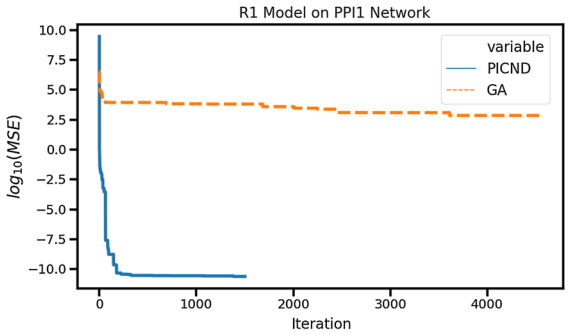

Fig. 14 Comparison of error over iterations of both algorithms with R1 model on PPI1 network

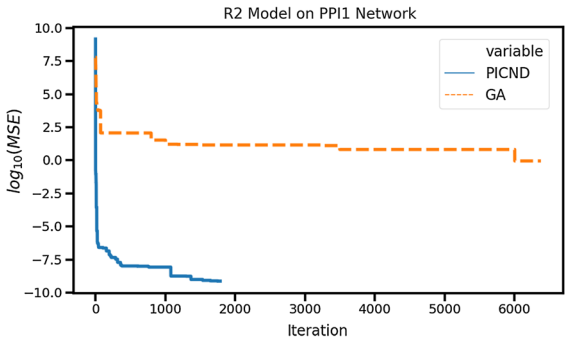

Fig. 15 Comparison of error over iterations of both algorithms with R2 model on PPI1 network

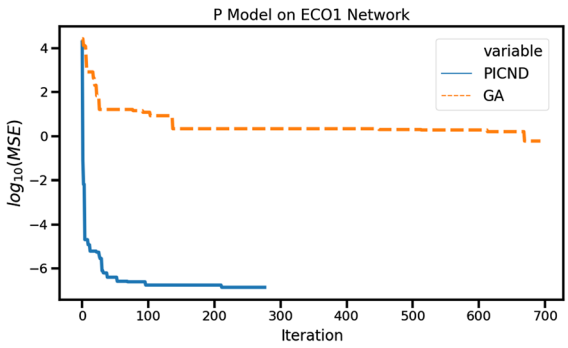

Fig. 16 Comparison of error over iterations of both algorithms with P model on ECO1 network

not have the drawbacks of other symbolic regression methods. First, unlike the method developed by Barzel et al., it does not depend on the observation of the effect of an external perturbation to the system. Additionally, because the domain of the problem is restricted to complex networks as opposed to dynamical systems in general, and because the list of candidate dynamic models to look for is known beforehand, unlike the SINDy algorithm, the creation of the library of non-

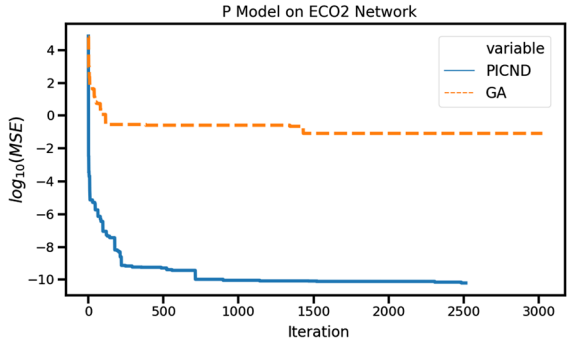

Fig. 17 Comparison of error over iterations of both algorithms with P model on ECO2 network

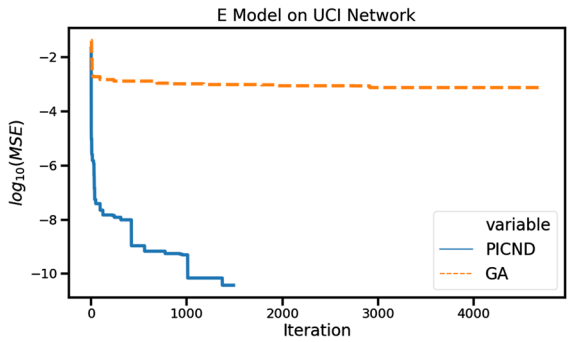

Fig. 18 Comparison of error over iterations of both algorithms with E model on UCI network

linear functions does not require manual intervention; therefore, PICND can be utilized to automatically infer the dynamics of a given data set, provided that these dynamics are similar to one of the candidate dynamic models. Finally, a purely genetic approach requires us to provide restrictions on the search space of the coefficients as well as the exponents of the terms in the governing differential equation, while PICND only requires a restriction on the search space of the exponents.

We have also established that the PICND algorithm is much faster than a purely genetic approach. Perhaps a more clever and complicated evolution strategy would speed up the GA algorithm, such as the NEAT strategy [36] which has recently been used for data-driven discovery of free-form governing differential equations [37]; however, in that case we could apply the same strategy to PICND, which would speed up the process unless the SINDy step becomes a bottleneck, which is unlikely due to the efficiency of the SINDy algorithm.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Watts, D.J., Strogatz, S.H.: Collective dynamics of "small-world" networks. Nature **393**(6684), 440–442 (1998)
2. Barabási, A.L., Albert, R.: Emergence of scaling in random networks. Science **286**, 509–512 (1999)
3. Pastor-Satorras, R., Vázquez, A., Vespignani, A.: Dynamical and correlation properties of the Internet. Phys. Rev. Lett. **87**, 258701 (2001)
4. Newman, M.E.J.: Assortative mixing in networks. Phys. Rev. Lett. **89**, 208701–4 (2002)
5. Strogatz, S.H.: Exploring complex networks. Nature **410**, 268–276 (2001)
6. Drogovtsev, S.N., Mendez, J.F.F.: Evolution of networks: from biological nets to the Internet and WWW. Oxford University Press, Oxford (2003)
7. Pastor-Satorras, R., Vespignani, A.: Evolution and structure of the Internet: A statistical physics approach. Cambridge University Press, Cambridge, U.K. (2004)
8. Palla, G., Derényi, I., Farkas, I., Vicsek, T.: Uncovering the overlapping community structure of complex networks in nature and society. Nature **435**, 814–818 (2005)
9. Caldarelli, G.: Scale-free networks: complex webs in nature and technology. Oxford University Press, New York (2007)
10. Newman, M.E.J.: Networks - an introduction. Oxford University Press, New York (2010)
11. Cohen, R., Havlin, S.: Complex networks: Structure, robustness and function. Cambridge University Press, New York, NY (2010)
12. Holter, N.S., Maritan, A., Cieplak, M., Fedoroff, N.V., Banavar, J.R.: Dynamic modeling of gene expression data. Proc Natl Acad Sci US. **98**, 1693–1698 (2001)
13. Dorogovtsev, S.N., Goltsev, A.V.: Critical phenomena in complex networks. Rev. Mod. Phys. **80**, 1275–1335 (2008)
14. Barrat, A., Barthélemy, M., Vespignani, A.: Dynamical Processes on Complex Networks. Cambridge University Press, Cambridge (2008)
15. Barzel, B., Barabási, A.: Universality in network dynamics. Nature Phys. **9**, 673–681 (2013)
16. Yan, L., Chen, W., Fang, X., et al.: Event-triggered synchronization for second-order nodes in complex dynamical network with time-varying coupling matrices. Nonlinear Dyn. **98**, 2227–2245 (2019)
17. Gao, J., Barzel, B., Barabási, A.: Universal resilience patterns in complex networks. Nature **530**, 307–312 (2016)
18. Harush, U., Barzel, B.: Dynamic patterns of information flow in complex networks. Nat. Commun. **8**, 2181 (2017)
19. Hens, C., Harush, U., Haber, S., Cohen, R., Barzel, B.: Spatiotemporal signal propagation in complex networks. Nat. Phys. **15**, 403–412 (2019)
20. Barzel, B., Liu, Y., Barabási, A.: Constructing minimal models for complex system dynamics. Nat. Commun. **6**, 7186 (2015)
21. Brunton, S.L., Proctor, J.L., Kutz, J.N.: Discovering governing equations from data by sparse identification of nonlinear dynamical systems. Proc. Nat. Acad. Sci. **113**(15), 3932–3937 (2016)
22. Schmidt, M., Lipson, H.: Distilling free-form natural laws from experimental data, Science 03 Apr 2009, **324**(5923), 81-85
23. Quaranta, G., Lacarbonara, W., Masri, S.F.: A review on computational intelligence for identification of nonlinear dynamical systems. Nonlinear Dyn. **99**, 1709–1761 (2020)
24. Zhang, L., Schaeffer, H.: On the Convergence of the SINDy Algorithm, Multiscale Model. Simul., **17**(3), 948–972
25. Mangan, N.M., Kutz, J.N., Brunton, S.L., Proctor, J.L.: Model selection for dynamical systems via sparse regression and information criteria. Proc. R. Soc. A. **473**, 20170009
26. Mangan, N.M., Brunton, S.L., Proctor, J.L., Kutz, J.N.: Inferring biological networks by sparse identification of nonlinear dynamics. IEEE Transac. Mol. Biol. Multi-Scale Commun. **2**(1), 52–63 (2016)
27. Sun, X., Liu, Z., Perc, M.: Effects of coupling strength and network topology on signal detection in small-world neuronal networks. Nonlinear Dyn. **96**, 2145–2155 (2019)
28. Roy, M., Poria, S.: Enhancement of synchronized chaotic state in a delay-coupled complex neuronal network. Nonlinear Dyn (2020)
29. Karlebach, G., Shamir, R.: Modelling and analysis of gene regulatory networks. Nat. Rev. Mol. Cell Biol. **9**, 770–780 (2008)
30. Novozhilov, A.S., Karev, G.P., Koonin, E.V.: Biological applications of the theory of birth-and-death processes. Briefings in Bioinf. **7**(1), 70–85 (2006)
31. Pastor-Satorras, R., Castellano, C., Van Mieghem, P., Vespignani, A.: Epidemic processes in complex networks. Rev. Mod. Phys. **87**, 925–958 (2015)
32. Szklarczyk, D., Gable, A.L., Lyon, D., et al.: STRING v11: protein-protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets. Nucleic Acids Res. **47**(D1), D607–D613 (2019)
33. Leskovec, J., Faloutsos, C.: Sampling from large graphs. In: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '06). Association for Computing Machinery, New York, NY, USA, 631–636
34. Dicks, L.V., Corbet, S.A., Pywell, R.F.: Compartmentalization in plant–insect flower visitor webs. Journal of Animal Ecology **71**, 32-43
35. Opsahl, T., Panzarasa, P.: Clustering in weighted networks. Social Networks **31**(2), 155–163
36. Stanley, K.O., Miikkulainen, R.: Evolving neural networks through augmenting topologies. Evol. Comput. **10**(2), 99–127 (2002)
37. Atkinson, S., Subber, W., Wang, L., Khan, G., Hawi, P., Ghanem, R.: Data-driven discovery of free-form governing differential equations, unpublished