General optimization framework for accurate and efficient reconstruction of symmetric complex networks from dynamical data

Chuang Ma,[1] Ying-Cheng Lai,[2] Xiang Li,[3] and Hai-Feng Zhang[4,*]
[1]*School of Internet, Anhui University, Hefei 230601, China*
[2]*School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, Arizona 85287, USA*
[3]*The Institute of Complex Networks and Intelligent Systems, Shanghai Research Institute for Intelligent Autonomous Systems, Tongji University, Shanghai 201210, China*
[4]*School of Mathematical Science, Anhui University, Hefei 230601, China*

(Received 19 September 2022; revised 26 February 2023; accepted 18 August 2023; published 8 September 2023)

The challenging problem of network reconstruction from dynamical data can in general be formulated as an optimization task of solving multiple linear equations. Existing approaches are of the two types: Point-by-point (PBP) and global methods. The local PBP method is computationally efficient, but the accuracies of its solutions are somehow low, while a global method has the opposite traits: High accuracy and high computational cost. Taking advantage of the network symmetry, we develop a novel framework integrating the advantages of both the PBP and global methods while avoiding their shortcomings: i.e., high reconstruction accuracy is guaranteed, but the computational cost is orders of magnitude lower than that of the global methods in the literature. The mathematical principle underlying our framework is block coordinate descent (BCD) for solving optimization problems, where the various blocks are determined by the network symmetry. The reconstruction framework is validated by numerical examples with a variety of network structures (i.e., sparse and dense networks) and dynamical processes. Our success is a demonstration that the general principle of exploiting symmetry can be extended to tackling the challenging inverse problem or reverse engineering of complex networks. Since solving a large number of linear equations is key to a plethora of problems in science and engineering, our BCD-based network reconstruction framework will find broader applications.

DOI: 10.1103/PhysRevE.108.034304

I. INTRODUCTION

In applications of complex network theories, an essential requirement is that the network structure is known to be able to study the pertinent dynamical processes on the network, predict the future state of the system, and even harness the network dynamics through the articulation of some optimal control strategies, and so on. However, in real-world situations, the detailed structure or topology of a complex network is often unknown, or only the dynamical signals (time series) from different locations of the network are available, rendering fundamental the problem of inferring the network structure from data [1,2]. In the past two decades, various methods have been developed to address this challenging "inverse" problem in modern network science and engineering [2–10].

A natural reconstruction approach is to utilize the similarity or correlation among dynamic signals of individual nodes to infer the network structure. In this context, various metrics such as the Pearson correlation coefficient, Jaccard coefficient, mutual information, and Granger causality analysis can be used to capture the interactions between nodes. The ultimate network structure is then derived by applying a selected threshold. Furthermore, if the node signal data adhere to a Gaussian distribution, the network structure can be ascertained through the covariance matrix [11]. Another approach to network reconstruction involves identifying neighboring nodes in the network by regressing the state of a given node against the states of other nodes [12]; among these methods, linear regression is the most frequently utilized technique.

A common operational principle underlying most existing data-based network reconstruction methods is based on the "point-by-point" (PBP) strategy whereby the local network structure at each and every available node is inferred, and then all the resulting local connections (links) are combined (e.g., intersection, union, or average) to remove spurious connections so as to generate a set of mutually consistent links among the nodes. Mathematically, this entails solving multiple sets of linear equations, because the local inference problem often can be cast as finding the solutions of a set of linear equations. For example, if the network to be inferred is sparse, local reconstruction can be formulated as a sparse optimization (or compressive sensing) problem [13,14], which is essentially a problem of solving a set of linear algebraic equations. This method is applied to the network reconstruction of a diverse array of dynamical processes: Coupled oscillatory dynamics [3,4,15], evolutionary game dynamics [13,16], epidemic spreading dynamics [17–22], transportation dynamics [2,16,23], and collective dynamics [24,25].

The existing PBP methods, while demonstrated to be reasonably effective in specific contexts, have failed to take

*haifengzhang1978@gmail.com

advantage of an essential and common feature of complex networks: Symmetry. In undirected networks, symmetry is natural because the network adjacency matrix is symmetric. In fact, real-world networks typically possess a large number of symmetries that shape not only the structural properties of the network such as the spectrum and redundancy but also the dynamical processes on it [26–34]. In particular, in the coupled dynamic network, the symmetries of the network can lead to phase relations, resonances, and synchronous or cycling chaos [35–37]. Since the symmetry constraints were not taken into account in the existing PBP-based methods, significant errors can arise in solving the linear equations at each node and in combining the local solutions. Intuitively, an alternative is to develop some sort of global method in which the reconstruction is not done locally at the nodal level but is done at the global scale of the whole network, i.e., solving all the equations at all nodes in the network at the same time, taking into account the network symmetry [38]. Apparently, this strategy is practically workable only for small networks. Compared with the symmetry constraint method, the error generated by the PBP method depends on the amount of data used for network reconstruction. If the data are insufficient, the PBP method will yield errors and inevitably lead to low reconstruction accuracy. While how large the network can be for the global method to be effective depends on many factors such as the total number of nodes, the phase-space dimension of the nodal dynamical processes, the available computational resources, etc., it is not unreasonable to anticipate the global method to fail in real-world applications that involve large complex networks. To develop a computationally feasible and efficient framework for data-based network reconstruction, taking advantage of the network symmetry so as to achieve high accuracies is highly desired.

One should note that the symmetry constraint considered in this work requires that not only is the network structure symmetric but also the matrix formed by the solution of this set of linear equations is symmetric. A counterexample is the reconstruction of a gene regulatory network. The network dynamics are nonlinear, and in the vicinity of a steady state, one can apply small perturbations to the system to generate a set of linear equations characterizing the connections in the local neighborhood of each node [10,39,40]. Even though its network structure (adjacency matrix) is symmetric, our method is not applicable for this case because the matrix formed by solving this set of linear equations may not be symmetric.

In this paper, we develop a framework to solve multiple linear equations subject to various symmetry constraints, which integrates the advantages of both the PBP and global methods while avoiding their shortcomings, i.e., when solving linear equation at each node, we regard the solution of the linear equation of other nodes as constraints (network symmetry) and obtain the solution through an iterative process. Our framework can be proved to be convergent to the global method by the block coordinate descent (BCD) method, so it not only guarantees the high accuracy of network reconstruction but also is computationally efficient (with computational load far less than that with any global reconstruction method). Differing from previous methods based on sparse optimization, our reconstruction framework is effective for both sparse and dense networks. Because the mathematical underpinnings of our framework are the block coordinate descent (BCD) method for solving optimization problems, it is called the BCD framework. In brief, BCD is an iterative algorithm for optimization that successively minimizes the objective function in each block coordinate while leaving the other coordinates fixed. Exploiting the network symmetry to determine the blocks, we arrive at an optimization framework for efficiently solving the complicated set of linear equations resulting from the inverse problem of data-based network reconstruction. Since the optimization process uses the symmetry constraint when each of the nodes (blocks) is optimized, the method is global and enjoys the high accuracy—the intrinsic advantage of any global optimization method. But since each optimization is done on only a block (node), the computational load can be significantly reduced as compared with the standard global method. Our mathematical formulation and numerical experiments with a variety of network structures and dynamical processes demonstrate that the BCD method has the reconstruction accuracy of the traditional global method but with a computational load comparable to that of the local optimization method. These advantages are brought upon by the network symmetry, providing another example demonstrating the power of using symmetry to tackle difficult problems. While the development of our BCD optimization framework is motivated by and tested on data-based reconstruction of complex networks with diverse network structures and dynamical processes, the universal core of the optimization problem is to solve a set of sophisticated linear equations, which is key to many problems in different fields of science and engineering. We expect that our BCD framework will find broader applications beyond network reconstruction.

II. A BRIEF REVIEW OF POINT-BY-POINT AND GLOBAL METHODS

To place our work in a proper context, we review the basics of the previous PBP and global methods in the language of linear-equation solving. Such a description facilitates an explanation and understanding of our BCD based reconstruction framework.

The network reconstruction problem from some dynamics often can be cast as finding the solutions of a set of linear equations. A typical example is a complex network of n nodes with coupled nonlinear oscillatory dynamics system [1] (see Appendix A 1 for a more detailed description), which can be described by

$$\dot{\boldsymbol{x}}_i = \boldsymbol{f}_i(\boldsymbol{x}_i) + \sum_{j=1}^{n} a_{i,j}\boldsymbol{g}_{i,j}(\boldsymbol{x}_i, \boldsymbol{x}_j), \tag{1}$$

where $\boldsymbol{x}_i \in \mathbb{R}^D$ is the state vector of the ith node, and the functions $\boldsymbol{f}_i$: $\mathbb{R}^D \to \mathbb{R}^D$ and $\boldsymbol{g}_{i,j}$: $\mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}^D$ govern the intrinsic and interaction dynamics of node i, respectively, and $a_{i,j}$ is an element of the adjacency matrix $\mathcal{A}$: $a_{i,j} = 1$ if there is a physical interaction from node i to j, and $a_{i,j} = 0$ otherwise. The network reconstruction task entails obtaining solutions of the adjacency matrix $\mathcal{A}$.

For the dth component of the state vector at node i, the dynamical equation is

$$\dot{x}_i^{(d)}(\tau_m) = f_i^{(d)}(x_i(\tau_m)) + \sum_{j=1}^{n} a_{i,j} g_{i,j}^{(d)}(x_i(\tau_m), x_j(\tau_m)). \quad (2)$$

In general, $\dot{x}_i^{(d)}(\tau_m)$ and $x_i^{(d)}(\tau_m)$ can be sampled as time-series data. When $\boldsymbol{f}_i$ and $\boldsymbol{g}_{i,j}$ are known, it is clear that Eq. (2) can be rewritten as the following linear equations:

$$\mathcal{C}^i \cdot \boldsymbol{A}^i = \boldsymbol{y}^i, \quad (3)$$

where $\mathcal{C}^i \in \mathbb{R}^{m_i \times n}$ and $\boldsymbol{y}^i \in \mathbb{R}^{m_i}$, which can be calculated by $x_i^{(d)}(\tau_m)$ of sampling m_i times, i.e., $m = 1, 2, \ldots, m_i$, and m_i represents the length of time series or the number of equations in the linear equations: $m_i = M, i = 1, 2, \ldots, n$ with M being the length of the time series recording of the nodal state. The quantity to be solved is $\boldsymbol{A}^i = [a_{i,1}, \ldots, a_{i,i}, \ldots, a_{i,n}]^T$, which represents the ith row of the adjacency matrix $\mathcal{A}$, where $a_{i,j} > 0$ if node j is a neighbor of node i and $a_{i,j} = 0$ otherwise. The neighbors of node i can be inferred when $\boldsymbol{A}^i$ is solved from the set of linear equations in Eq. (3), and the whole network can be reconstructed by inferring the neighbors of all nodes.

A. Point-by-point method

To reconstruct the network structure from continuous or discrete dynamics, one infers the neighbor of each node i in the network. Similar to the network reconstruction from coupled nonlinear oscillatory dynamics, the local reconstruction task is equivalent to solving the following linear equations [1–4,10,13–18,20,21,38–48]:

$$\mathcal{C}^i \cdot \boldsymbol{A}^i = \boldsymbol{y}^i + \boldsymbol{\varepsilon}^i, \quad (4)$$

where $\boldsymbol{\varepsilon}^i \in \mathbb{R}^{m_i}$ represents the noise in the time series measurements from node i. The whole network can be reconstructed by inferring the neighbors of all the nodes, a task that requires solving n sets of m_i linear equations, one for each node:

$$\begin{aligned} \mathcal{C}^1 \cdot \boldsymbol{A}^1 &= \boldsymbol{y}^1 + \boldsymbol{\varepsilon}^1, \\ \mathcal{C}^2 \cdot \boldsymbol{A}^2 &= \boldsymbol{y}^2 + \boldsymbol{\varepsilon}^2, \\ &\vdots \\ \mathcal{C}^n \cdot \boldsymbol{A}^n &= \boldsymbol{y}^n + \boldsymbol{\varepsilon}^n. \end{aligned} \quad (5)$$

1. *Point-by-point method without sparsity constraint*

The essence of the PBP method is to treat the linear equations in Eq. (5) as independent, which are solved one by one. For example, each equation can be solved by the least-squares method:

$$\min \ \|\mathcal{C}^i \cdot \boldsymbol{A}^i - \boldsymbol{y}^i\|_2^2, \quad (6)$$

whose solution is

$$\boldsymbol{A}^i = [(\mathcal{C}^i)^T \cdot \mathcal{C}^i]^{-1} \cdot (\mathcal{C}^i)^T \cdot \boldsymbol{y}^i, \quad (7)$$

where the matrix $(\mathcal{C}^i)^T \cdot \mathcal{C}^i$ can be invertible only for $m_i \geqslant n$. To this end, a two-norm constraint (i.e., ridge regression) is often introduced to ensure that the matrix inverse in Eq. (7) exists:

$$\min \ \|\mathcal{C}^i \cdot \boldsymbol{A}^i - \boldsymbol{y}^i\|_2^2 + \beta\|\boldsymbol{A}^i\|_2^2. \quad (8)$$

With this constraint, the solution of Eq. (8) can be written as

$$\boldsymbol{A}^i = [(\mathcal{C}^i)^T \cdot \mathcal{C}^i + \beta\mathcal{I}]^{-1} \cdot (\mathcal{C}^i)^T \cdot \boldsymbol{y}^i, \quad (9)$$

where $\mathcal{I}$ is the identify matrix and β is a hyperparameter. For $\beta = 0$, Eq. (9) reduces to Eq. (7). Otherwise, $(\mathcal{C}^i)^T \cdot \mathcal{C}^i + \beta\mathcal{I}$ is invertible for $\beta > 0$.

For an nth-order matrix, the computational time required for solving the inverse of the matrix is $O(n^3)$, and the time to solve $(\mathcal{C}^i)^T \cdot \mathcal{C}^i$ is $O(Mn^2)$ for $m_i = M, i = 1, 2, \ldots, n$. Therefore, the total time required for solving Eq. (5) is $O(n(Mn^2 + n^3)) = O(Mn^3 + n^4)$. In general, M has a linear dependence on n [$M = O(n)$]. Overall, the required computational time for reconstructing a general, nonsparse network is $O(n^4)$. Since each equation of Eqs. (5) is solved independently, the computations can be done in parallel. Therefore, we have an effective runtime of $O(n^4/p)$ by assuming that there are p processors.

2. *PBP-ADMM method for sparse networks*

Many real-world networks are sparse, i.e., the number of neighbors of each node is much smaller than the size of the network, rendering applicable compressed sensing based sparse optimization method for solving each linear equation in Eqs. (5) with a small amount of data. In general, Lasso regression can be used [16,49]:

$$\min \ \tfrac{1}{2}\|\mathcal{C}^i \cdot \boldsymbol{A}^i - \boldsymbol{y}^i\|_2^2 + \lambda\|\boldsymbol{A}^i\|_1, \quad (10)$$

where λ is a hyperparameter. The classical method to solve the linear equations defined by Eq. (10) is the alternating direction method of multipliers (ADMM) [50]. In particular, the optimization problem (10) is equivalent to the follow equation:

$$\min \tfrac{1}{2}\|\mathcal{C}^i \cdot \boldsymbol{A}^i - \boldsymbol{y}^i\|_2^2 + \lambda\|\boldsymbol{Z}^i\|_1 \quad \text{s.t. } \boldsymbol{A}^i - \boldsymbol{Z}^i = \boldsymbol{0}, \quad (11)$$

where $\boldsymbol{Z}^i \in \mathbb{R}^n$ and whose augmented Lagrangian function is given as

$$\begin{aligned} L_\rho(\boldsymbol{A}^i, \boldsymbol{Z}^i, \boldsymbol{V}^i) = {} & \frac{1}{2}\|\mathcal{C}^i \cdot \boldsymbol{A}^i - \boldsymbol{y}^i\|_2^2 + \lambda\|\boldsymbol{Z}^i\|_1 \\ & + (\boldsymbol{V}^i)^T \cdot (\boldsymbol{A}^i - \boldsymbol{Z}^i) + \frac{\rho}{2}\|\boldsymbol{A}^i - \boldsymbol{Z}^i\|_2^2, \end{aligned} \quad (12)$$

where $\boldsymbol{V}^i \in \mathbb{R}^n$ is the vector of n Lagrange multipliers. The ADMM algorithm gives

$$\begin{aligned} (\boldsymbol{A}^i)^{k+1} &= \arg\min_{\boldsymbol{A}^i} L_\rho(\boldsymbol{A}^i, (\boldsymbol{Z}^i)^k, (\boldsymbol{V}^i)^k), \\ (\boldsymbol{Z}^i)^{k+1} &= \arg\min_{\boldsymbol{Z}^i} L_\rho((\boldsymbol{A}^i)^{k+1}, \boldsymbol{Z}^i, (\boldsymbol{V}^i)^k), \\ (\boldsymbol{V}^i)^{k+1} &= (\boldsymbol{V}^i)^k + \rho((\boldsymbol{A}^i)^{k+1} - (\boldsymbol{Z}^i)^{k+1}), \end{aligned} \quad (13)$$

where k represents the kth iteration in ADMM algorithm. The main steps are as follows: First, by solving $(\boldsymbol{A}^i)^{k+1} = \arg\min_{\boldsymbol{A}^i} L_\rho(\boldsymbol{A}^i, (\boldsymbol{Z}^i)^k, (\boldsymbol{V}^i)^k)$, one gets

$$\begin{aligned} (\boldsymbol{A}^i)^{k+1} = {} & [(\mathcal{C}^i)^T \cdot \mathcal{C}^i + \rho\mathcal{I}]^{-1} \\ & \cdot [(\mathcal{C}^i)^T \cdot \boldsymbol{y}^i + \rho(\boldsymbol{Z}^i)^k - (\boldsymbol{V}^i)^k]. \end{aligned} \quad (14)$$

Next, by solving $(\boldsymbol{Z}^i)^{k+1} = \arg\min_{\boldsymbol{Z}^i} L_\rho((\boldsymbol{A}^i)^{k+1}, \boldsymbol{Z}^i, (\boldsymbol{V}^i)^k)$, one obtains

$$\begin{aligned}(\boldsymbol{Z}^i)^{k+1} &= S_{\frac{\lambda}{\rho}}\left((\boldsymbol{A}^i)^{k+1} + \frac{(\boldsymbol{V}^i)^k}{\rho}\right)\\ &= \left[\left((\boldsymbol{A}^i)^{k+1} + \frac{(\boldsymbol{V}^i)^k}{\rho}\right) - \frac{\lambda}{\rho}\right]_+\\ &\quad - \left[\left(-(\boldsymbol{A}^i)^{k+1} - \frac{(\boldsymbol{V}^i)^k}{\rho}\right) - \frac{\lambda}{\rho}\right]_+,\end{aligned} \tag{15}$$

where $(x-y)_+ = [(x-y) + |x-y|]/2$. Equation (13) can then be rewritten as

$$\begin{aligned}(\boldsymbol{A}^i)^{k+1} &= [(\mathcal{C}^i)^T \cdot \mathcal{C}^i + \rho\mathcal{I}]^{-1} \cdot [(\mathcal{C}^i)^T \cdot \boldsymbol{y}^i + \rho(\boldsymbol{Z}^i)^k - (\boldsymbol{V}^i)^k],\\ (\boldsymbol{Z}^i)^{k+1} &= \left(\boldsymbol{S}_1^k + \left|\boldsymbol{S}_1^k\right|\right)/2 - \left(\boldsymbol{S}_2^k + \left|\boldsymbol{S}_2^k\right|\right)/2,\\ (\boldsymbol{V}^i)^{k+1} &= (\boldsymbol{V}^i)^k + \rho((\boldsymbol{A}^i)^{k+1} - (\boldsymbol{Z}^i)^{k+1}),\end{aligned} \tag{16}$$

with

$$\begin{aligned}\boldsymbol{S}_1^k &= (\boldsymbol{A}^i)^{k+1} + \frac{(\boldsymbol{V}^i)^k}{\rho} - \frac{\lambda}{\rho},\\ \boldsymbol{S}_2^k &= -(\boldsymbol{A}^i)^{k+1} - \frac{(\boldsymbol{V}^i)^k}{\rho} - \frac{\lambda}{\rho}.\end{aligned} \tag{17}$$

The neighbors of each node can be determined by iterating Eq. (16) until it reaches a steady state [i.e., $\|(\boldsymbol{A}^i)^{k+1} - (\boldsymbol{A}^i)^k\|_2^2 \leqslant \epsilon = 0.001$] or the maximum number of iterations (1000 is given). Once the neighborhoods of all nodes are determined, the whole network structure can be obtained.

The required computational time is determined by two factors. First, it is necessary to calculate $[(\mathcal{C}^i)^T \cdot \mathcal{C}^i + \rho\mathcal{I}]^{-1}$ and $(\mathcal{C}^i)^T \cdot \boldsymbol{y}^i$ for each node i. Both are invariant quantities that can be solved before iteration, for which the time requirement is $O(n^3)$ $[M = O(n)]$. Second, the time needed for the iteration process Eq. (16) for each node is $O(t_{\max} n^2)$ with $t_{\max}$ being the maximum number of iterations. As a result, the total computational time of the PBP-ADMM algorithm is $O(t_{\max} n^3 + n^4)$. Similarly, the local network structure at each node can be reconstructed independently, the computations can be done in parallel. Namely, when there are p processors, we have an effective runtime of $O(t_{\max} n^3/p + n^4/p)$.

B. Global method

A tacit assumption underlying the PBP or PBP-ADMM methods is that the resulting linear equations are independent of each other. This may lead to errors, especially for symmetric networks. For example, for an undirected network with a symmetric adjacency matrix: $a_{i,j} = a_{j,i}$ for $i, j = 1, 2, \ldots, n$, the PBP or PBP-ADMM methods can generate solutions that violate this symmetric relation: $a_{i,j} \neq a_{j,i}$ as $a_{i,j}$ is solved from node i but $a_{j,i}$ is obtained from node j. In fact, for undirected networks, Eq. (5) is a set of multiple linear equations with symmetric properties. Failure to take into account the symmetry in the equations will result in reduced solution accuracies. An approach is to develop a global method with or without the sparsity constraint by treating Eq. (5) as a whole.

1. Global method without sparsity constraint

The problem to solve Eq. (5) as a whole can be formulated as

$$\min \sum_i \|\mathcal{C}^i \cdot \boldsymbol{A}^i - \boldsymbol{y}^i\|_2^2, \tag{18}$$

with the symmetry $a_{i,j} = a_{j,i}$, where the number of unknown quantities is $n(n+1)/2$. Adding the two-norm constraint to Eq. (18) to ensure that the matrix is invertible leads to

$$\min \sum_i \|\mathcal{C}^i \cdot \boldsymbol{A}^i - \boldsymbol{y}^i\|_2^2 + \frac{\beta}{2}\|\mathcal{A}\|_2^2. \tag{19}$$

Globally, Eq. (19) can be written as

$$\min \quad \|\widehat{\mathcal{C}} \cdot \widehat{\boldsymbol{A}} - \widehat{\boldsymbol{y}}\|_2^2 + \beta\|\widehat{\boldsymbol{A}}\|_2^2, \tag{20}$$

where the vector $\widehat{\boldsymbol{A}} = [a_{1,1}, \ldots, a_{1,n}, a_{2,2}, \ldots, a_{2,n}, \ldots, a_{n-1,n}, a_{n,n}]^T$ has $n(n+1)/2$ components, $\widehat{\mathcal{C}} \in \mathbb{R}^{[M*n]\times[n(n+1)/2]}$ and $\widehat{\boldsymbol{y}} \in \mathbb{R}^{M*n}$ are the global matrix and vector, respectively. The solution of Eq. (20) can be written as

$$\widehat{\boldsymbol{A}} = [(\widehat{\mathcal{C}})^T \cdot \widehat{\mathcal{C}} + \beta\mathcal{I}]^{-1} \cdot (\widehat{\mathcal{C}})^T \cdot \widehat{\boldsymbol{y}}. \tag{21}$$

Because $\widehat{\mathcal{C}}$ is a matrix with Mn rows and $n(n+1)/2$ columns, the time required for calculating $(\widehat{\mathcal{C}})^T \cdot \widehat{\mathcal{C}}$ is $O(n^2(Mn)n^2) = O(Mn^5)$. Note that the dimension of the matrix $[(\widehat{\mathcal{C}})^T \cdot \widehat{\mathcal{C}} + \beta\mathcal{I}]$ is $[n(n+1)/2] \times [n(n+1)/2]$. Since the time required to calculate the inverse of an $n \times n$ matrix is $O(n^3)$, the time needed for directly calculating the inverse of $[(\widehat{\mathcal{C}})^T \cdot \widehat{\mathcal{C}} + \beta\mathcal{I}]$ is $O(n^6)$. The total computational time required to obtain the global solution is thus $O(Mn^5 + n^6)$, which is $O(n^6)$ if $M = O(n)$.

2. Global-ADMM method for sparse networks

For sparse networks, Eq. (18) is subject to the sparsity constraint and becomes

$$\min \sum_i \|\mathcal{C}^i \cdot \boldsymbol{A}^i - \boldsymbol{y}^i\|_2^2 + \lambda\|\mathcal{A}\|_1, \tag{22}$$

which can be written in the following global form:

$$\min \|\widehat{\mathcal{C}} \cdot \widehat{\boldsymbol{A}} - \widehat{\boldsymbol{y}}\|_2^2 + 2\lambda\|\widehat{\boldsymbol{A}}\|_1, \tag{23}$$

where the matrix $\widehat{\mathcal{C}}$ and the vectors $\widehat{\boldsymbol{A}}$ and $\widehat{\boldsymbol{y}}$ have the same forms as those in Eq. (20). Equation (23) can be rewritten as

$$\begin{aligned}\min \quad & \tfrac{1}{2}\|\widehat{\mathcal{C}} \cdot \widehat{\boldsymbol{A}} - \widehat{\boldsymbol{y}}\|_2^2 + \lambda\|\widehat{\boldsymbol{A}}\|_1\\ \text{s.t.} \quad & \widehat{\boldsymbol{A}} - \widehat{\boldsymbol{Z}} = 0,\end{aligned} \tag{24}$$

whose augmented Lagrangian function is

$$\begin{aligned}L_\rho(\widehat{\boldsymbol{A}}, \widehat{\boldsymbol{Z}}, \widehat{\boldsymbol{V}}) &= \frac{1}{2}\|\widehat{\mathcal{C}} \cdot \widehat{\boldsymbol{A}} - \widehat{\boldsymbol{y}}\|_2^2 + 2\lambda\|\widehat{\boldsymbol{Z}}\|_1\\ &\quad + (\widehat{\boldsymbol{V}})^T(\widehat{\boldsymbol{A}} - \widehat{\boldsymbol{Z}}) + \frac{\rho}{2}\|\widehat{\boldsymbol{A}} - \widehat{\boldsymbol{Z}}\|_2^2,\end{aligned} \tag{25}$$

where $\widehat{\boldsymbol{V}} \in \mathbb{R}^{n(n+1)/2}$ is the vector of $n(n+1)/2$ Lagrange multipliers. Application of the ADMM algorithm leads to

$$\begin{aligned}(\widehat{\boldsymbol{A}})^{k+1} &= [(\widehat{\mathcal{C}})^T \cdot \widehat{\mathcal{C}} + \rho\mathcal{I}]^{-1} \cdot [(\widehat{\mathcal{C}})^T \cdot \widehat{\boldsymbol{y}} + \rho(\widehat{\boldsymbol{Z}})^k - (\widehat{\boldsymbol{V}})^k],\\ (\widehat{\boldsymbol{Z}})^{k+1} &= \tfrac{S_1^k + |S_1^k|}{2} - \tfrac{S_2^k + |S_2^k|}{2},\\ (\widehat{\boldsymbol{V}})^{k+1} &= (\widehat{\boldsymbol{V}})^k + \rho((\widehat{\boldsymbol{A}})^{k+1} - (\widehat{\boldsymbol{Z}})^{k+1}),\end{aligned} \tag{26}$$

with

$$\begin{aligned} \boldsymbol{S}_1^k &= (\widehat{\boldsymbol{A}})^{k+1} + \frac{(\widehat{\boldsymbol{V}})^k}{\rho} - \frac{\lambda}{\rho}, \\ \boldsymbol{S}_2^k &= -(\widehat{\boldsymbol{A}})^{k+1} - \frac{(\widehat{\boldsymbol{V}})^k}{\rho} - \frac{\lambda}{\rho}. \end{aligned} \tag{27}$$

The connectivity of each node can be inferred by iterating Eq. (26) until convergence is achieved. The matrix inverse $[(\widehat{\mathcal{C}})^T \cdot \widehat{\mathcal{C}} + \rho\mathcal{I}]^{-1}$ and the vector $(\widehat{\mathcal{C}})^T \cdot \widehat{\boldsymbol{y}}$ can be calculated before the iterations with the computational time $O(n^6)$ for $M = O(n)$. The iteration process requires computational time $O(t_{\max} n^4)$. The total computational time required of the global ADMM algorithm under the sparsity constraint is thus $O(t_{\max} n^4 + n^6)$.

III. METHOD BASED ON BLOCK COORDINATE DESCENT

The PBP method and its variant PBP-ADMM method suffer one significant drawback of low accuracy, because Eq. (5) is solved point by point without taking into account the network symmetry. In fact, the PBP or PBP-ADMM solutions typically violate the symmetry. While synthesizing all the local equations into a set of global equations overcomes this difficulty as the symmetry is naturally represented in the equations (i.e., GLO and GLO-ADMM methods), the amount of computation required is often infeasible because of the need to solve equations with a very large number of unknown variables at the same time.

The fact that, for the solutions of Eq. (5), the global method has higher accuracy and the PBP method is computationally efficient has motivated us to integrate the two methods by exploiting the natural symmetry of the network. In particular, when solving each linear equation in Eq. (5), we regard the other linear equations as constraints (network symmetry) and obtain the solution through an iterative process. This leads to our general BCD framework. The basic idea is that when solving the ith equation (i.e., $a_{i,1}, \ldots, a_{i,i}, \ldots, a_{i,n}$) of Eqs. (5), the solution of other equations about $a_{1,i}, \ldots, a_{i,i}, \ldots, a_{n,i}$ can be taken as the constraint condition (i.e., symmetry constraint $a_{i,j} = a_{j,i}$), and iterating this process leads to an optimal solution.

A. General block coordinate descent method without sparsity constraint

More specifically, when the values of $\boldsymbol{A}^j$ for $j = 1, \ldots, i-1, i+1, \ldots, n$ are given in advance, the problem of solving $(a_{i,1}, \ldots, a_{i,i}, \ldots, a_{i,n})$ without the sparsity constraint can be written as solving the following optimization problem:

$$\min \ \|\mathcal{C}^i \cdot \boldsymbol{A}^i - \boldsymbol{y}^i\|_2^2 + \alpha\|\widetilde{\mathcal{C}}^i \cdot \boldsymbol{A}^i - \widetilde{\boldsymbol{y}}^i\|_2^2 + \beta\|\boldsymbol{A}^i\|_2^2, \tag{28}$$

where the solution of $\widetilde{\mathcal{C}}^i \cdot \boldsymbol{A}^i = \widetilde{\boldsymbol{y}}^i$ is $a_{1,i}, \ldots, a_{i,i}, \ldots, a_{n,i}$ in $\boldsymbol{A}^j$ for $j = 1, \ldots, i-1, i+1, \ldots, n$, then $\|\widetilde{\mathcal{C}}^i \cdot \boldsymbol{A}^i - \widetilde{\boldsymbol{y}}^i\|_2^2$ indicates the solutions of other linear equations about $\boldsymbol{A}^i$ as constraints (network symmetry $a_{i,j} = a_{j,i}$) when the ith linear equation of Eq. (5) is solved, and the parameter $\alpha \in [0, 1]$ is a hyperparameter. Here, we set

$$\begin{aligned} \widetilde{\mathcal{C}}^i &\equiv \left[\left(\mathscr{C}_i^1\right)^T, \ldots, \left(\mathscr{C}_i^{i-1}\right)^T, \boldsymbol{0}, \left(\mathscr{C}_i^{i+1}\right)^T, \ldots, \left(\mathscr{C}_i^n\right)^T\right]^T, \\ \widetilde{\boldsymbol{y}}^i &\equiv \left[\left(\boldsymbol{Y}_i^1\right)^T, \ldots, \left(\boldsymbol{Y}_i^{i-1}\right)^T, 0, \left(\boldsymbol{Y}_i^{i+1}\right)^T, \ldots, \left(\boldsymbol{Y}_i^n\right)^T\right]^T, \end{aligned} \tag{29}$$

where

$$\mathscr{C}_i^j = \begin{bmatrix} 0 & \cdots & 0 & C_{1,i}^j & 0 & \cdots & 0 \\ 0 & \cdots & 0 & C_{2,i}^j & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & 0 & C_{m_j,i}^j & 0 & \cdots & 0 \end{bmatrix}, \tag{30}$$

and

$$\boldsymbol{Y}_i^j = \begin{bmatrix} y_1^j - \sum_{l \neq i} C_{1,l}^i a_{j,l} \\ \vdots \\ y_{m_j}^j - \sum_{l \neq i} C_{m_j,l}^i a_{j,l} \end{bmatrix}, \tag{31}$$

with $C_{k,i}^j$ being the element of the kth row and the ith column in matrix $\mathcal{C}^j$, and y_i^j is the ith component of vector $\boldsymbol{y}^j$.

The above settings can ensure the convergence of BCD method to global method when $\alpha = 1$, the detailed process is given Theorem 1 at the end of this section. The hyperparameter α represents the degree of dependence on other linear equations when solving the ith linear equation of Eq. (5). If α is large, the solution at the current time step is close to that at the preceding time step, implying slow convergence. A way to improve the convergence rate is to reduce the dependence on the solutions at the preceding time step, i.e., reduce the value of α. For $\alpha = 0$, each linear equation of Eqs. (5) is solved independently, and the BCD method reduces to the PBP method.

With Eq. (28), the solution of $\boldsymbol{A}^i$ is obtained as

$$\begin{aligned} \boldsymbol{A}^i = {} & [(\mathcal{C}_i)^T \cdot \mathcal{C}_i + \alpha(\widetilde{\mathcal{C}}_i)^T \cdot \widetilde{\mathcal{C}}_i + \beta\mathcal{I}]^{-1} \\ & \cdot [(\mathcal{C}_i)^T \cdot \boldsymbol{y}^i + \alpha(\widetilde{\mathcal{C}}_i)^T \cdot \widetilde{\boldsymbol{y}}^i]. \end{aligned} \tag{32}$$

We set $\mathcal{F}^i \equiv (\mathcal{C}^i)^T \cdot \mathcal{C}^i$ and $\boldsymbol{G}^i \equiv (\mathcal{C}^i)^T \cdot \boldsymbol{y}^i$ and, then analyze the terms in Eq. (32) one by one. First, the term $(\widetilde{\mathcal{C}}_i)^T \cdot \widetilde{\mathcal{C}}_i$ can be calculated as

$$(\widetilde{\mathscr{C}}_i)^T \cdot \widetilde{\mathscr{C}}_i = \mathrm{diag}\left(\left[F_{i,i}^1, \ldots, F_{i,i}^{i-1}, 0, F_{i,i}^{i+1}, \ldots, F_{i,i}^n\right]^T\right), \tag{33}$$

with $F_{k,i}^j$ being the element of the kth row and the ith column in matrix $\mathcal{F}^i$, diag$(\boldsymbol{x})$ represents a diagonal matrix if $\boldsymbol{x}$ is a vector, or diag$(\mathcal{X})$ denotes a vector composed of the diagonal elements of the matrix $\mathcal{X}$. Second, the term $(\widetilde{\mathscr{C}}_i)^T \cdot \widetilde{\boldsymbol{y}}^i$ can be calculated as

$$\left(\widetilde{\mathscr{C}}_i\right)^T \cdot \widetilde{\boldsymbol{y}}^i = \sum_{j\neq i}\left(\mathscr{C}_i^j\right)^T \cdot \boldsymbol{Y}_i^j$$

$$= \begin{bmatrix} G_i^1 \\ \vdots \\ G_i^{i-1} \\ 0 \\ G_i^{i+1} \\ \vdots \\ G_i^N \end{bmatrix} - \text{diag}\left(\begin{bmatrix} \boldsymbol{F}^1(i,:) \\ \vdots \\ \boldsymbol{F}^{i-1}(i,:) \\ \boldsymbol{F}^i(i,:) \\ \boldsymbol{F}^{i+1}(i,:) \\ \vdots \\ F^N(i,:) \end{bmatrix}\begin{bmatrix} \left(\boldsymbol{A}_i^1\right)^T \\ \vdots \\ \left(\boldsymbol{A}_i^{i-1}\right)^T \\ \boldsymbol{0} \\ \left(\boldsymbol{A}_i^{i+1}\right)^T \\ \vdots \\ \left(\boldsymbol{A}_i^N\right)^T \end{bmatrix}\right), \tag{34}$$

where G_i^j is the ith component of vector $\boldsymbol{G}^j$, $\boldsymbol{F}^j(i,:)$ [or $\boldsymbol{F}^j(:,i)$] is the ith row (or ith column) of matrix $\mathcal{F}^j$, and $\boldsymbol{A}_i^j = [a_{j,1},\dots,a_{j,i-1},0,a_{j,i+1},\dots,a_{j,n}]^T$. Furthermore, letting

$$\boldsymbol{B}^i \equiv \left[G_i^1,\dots,G_i^{i-1},0,G_i^{i+1},\dots,G_i^n\right]^T,$$
$$\Gamma^i \equiv [\boldsymbol{F}^1(:,i),\dots,\boldsymbol{F}^n(:,i)]^T,$$
$$\Delta^i \equiv \left[\boldsymbol{A}_i^1,\dots,\boldsymbol{A}_i^{i-1},\boldsymbol{0},\boldsymbol{A}_i^{i+1},\dots,\boldsymbol{A}_i^n\right]^T,$$

we can simplify Eq. (34) as

$$(\widetilde{\mathscr{C}}_i)^T \cdot \widetilde{\boldsymbol{y}}^i = \boldsymbol{B}^i - \text{diag}(\Gamma^i \cdot \Delta^i). \tag{35}$$

Finally, the solution in Eq. (28) can be written as

$$\boldsymbol{A}^i = \Psi^i \cdot [\boldsymbol{G}^i + \alpha(\boldsymbol{B}^i - \text{diag}(\Gamma^i \cdot \Delta^i))], \tag{36}$$

where

$$\Psi^i \equiv \Big[\mathcal{F}^i + \beta\mathcal{I} + \alpha \cdot \text{diag}\left(\left[F_{i,i}^1,\dots,F_{i,i}^{i-1},0,F_{i,i}^{i+1},\dots,F_{i,i}^n\right]^T\right)\Big]^{-1}.$$

In Eq. (36), only quantity Δ^i contains the elements of $\mathcal{A}$. However, the other quantities (i.e., Ψ^i, Γ^i, $\boldsymbol{G}^i$, and $\boldsymbol{B}^i$) do not contain the elements of $\mathcal{A}$, so they can be calculated in advance.

In general, solving $\mathcal{A}$ is an iterative process. Given the initial value $\mathcal{A}$, we calculate $\boldsymbol{A}^i$ for each node i according to Eq. (36) until the solution converges. A key advantage of our BCD-based method is that the matrices Ψ^i and Γ^i as well as the vectors $\boldsymbol{G}^i$ and $\boldsymbol{B}^i$ in Eq. (36) can be calculated before the iteration process, making it highly efficient. In fact, the main reason behind the high efficiency of our BCD method is that the key matrix Ψ^i can be obtained in advance of the iterative process.

The computational load of our BCD method is determined by two tasks. First, the time required for calculating Ψ^i, Γ^i, $\boldsymbol{G}^i$, and $\boldsymbol{B}^i$ for all nodes is $O(n^4)$ [assuming $M = O(n)$]. Obviously, each node can be processed independently, so the time required in parallel is $O(n^4/p)$ when there are p processes. Second, the iteration process requires computational time on the order of $O(t_{\max}n^3)$. As a result, the total time required of our BCD method is $O(t_{\max}n^3 + n^4)$ [or $O(t_{\max}n^3 + n^4/p)$ in parallel]. Compared with the PBP method, there is an extra time on the order of $O(t_{\max}n^3)$ from the iteration process. However, the total time required of our BCD algorithm is practically negligible in comparison with the time required for a global algorithm. Remarkably, the final solution of our BCD method in general converges to the solution of the global method, thereby guaranteeing high accuracy.

Theorem 1. The convergence of the BCD method to the global method when $\alpha = 1$, i.e., the iterative Eq. (28) can converge to the solution of Eq. (19), when $\widetilde{\mathcal{C}}^i$ and $\widetilde{\boldsymbol{y}}^i$ are given in Eq. (29) and $\alpha = 1$.

Proof. As shown in Sec. II B 1, for general complex networks without any sparsity constraint, solving Eq. (5) as a whole is equivalent to obtaining solutions to the following optimization problem (global method):

$$\min \sum_i \|\mathcal{C}^i \cdot \boldsymbol{A}^i - \boldsymbol{y}^i\|_2^2 + \frac{\beta}{2}\|\mathcal{A}\|_2^2, \tag{37}$$

which has the same form as Eq. (19).

For Eq. (37), we can use the block coordinate descent (BCD) method to solve it. The basic idea is to calculate the values of $a_{i,1},\dots,a_{i,i},\dots,a_{i,n}$ by fixing the values of $a_{j,1},\dots,a_{j,i-1},a_{j,i+1},\dots,a_{j,n}$ for $j = 1,\dots,i-1,i+1,\dots,n$. When the iterative process converges, the resulting solution is the exact solution of Eq. (37). A theoretical analysis of the iterative process and its convergence is given in Appendix B. More specifically, if the values of $a_{j,1},\dots,a_{j,i-1},a_{j,i+1},\dots,a_{j,n}$ for $j = 1,\dots,i-1,i+1,\dots,n$ are given in advance, the problem of solving $(a_{i,1},\dots,a_{i,i},\dots,a_{i,n})$ from Eq. (28) is equivalent to solving the following optimization problem:

$$\min \|\mathcal{C}^i \cdot \boldsymbol{A}^i - \boldsymbol{y}^i\|_2^2 + \sum_{j\neq i}\left\|a_{i,j}\boldsymbol{C}^j(:,i) - \boldsymbol{Y}_i^j\right\|_2^2 + \beta\|\boldsymbol{A}^i\|_2^2, \tag{38}$$

where $\boldsymbol{C}^j(:,i)$ is the ith column of matrix $\mathcal{C}^j$, and the vector $\boldsymbol{Y}_i^j$ is given by

$$\boldsymbol{Y}_i^j = \begin{bmatrix} y_1^j - \sum_{l\neq i} C_{1,l}^i a_{j,l} \\ \vdots \\ y_{m_j}^j - \sum_{l\neq i} C_{m_j,l}^i a_{j,l} \end{bmatrix}.$$

Obviously, when $\widetilde{\mathcal{C}}^i$ and $\widetilde{\boldsymbol{y}}^i$ are set as shown in Eq. (29), Eq. (38) can be calculated as

$$\min \|\mathcal{C}^i \cdot \boldsymbol{A}^i - \boldsymbol{y}^i\|_2^2 + \|\widetilde{\mathcal{C}}^i \cdot \boldsymbol{A}^i - \widetilde{\boldsymbol{y}}^i\|_2^2 + \beta\|\boldsymbol{A}^i\|_2^2. \tag{39}$$

Therefore, Eq. (39) is consistent with Eq. (28) when $\alpha = 1$, namely, the BCD method converges to the global method. ■

B. Block coordinate descent method for sparse networks

With the sparsity constraint, the solutions of Eq. (22) is equivalent to

$$\begin{aligned} &\min \ \textstyle\sum_i \|\mathcal{C}^i \cdot \boldsymbol{A}^i - \boldsymbol{y}^i\|_2^2 + \lambda\|\mathcal{Z}\|_1 \\ &\text{s.t.} \quad \mathcal{A} - \mathcal{Z} = \boldsymbol{0}. \end{aligned} \tag{40}$$

The augmented Lagrangian function is

$$L_\rho(\mathcal{A},\mathcal{Z},\mathcal{V}) = \sum_i \|\mathcal{C}^i\cdot \boldsymbol{A}^i - \boldsymbol{y}^i\|_2^2 + \lambda\|\mathcal{Z}\|_1 + \sum_i (\boldsymbol{V}^i)^T\cdot(\boldsymbol{A}^i - \boldsymbol{Z}^i) + \frac{\rho}{2}\|\mathcal{A}-\mathcal{Z}\|_2^2, \quad (41)$$

where $\mathcal{A}$, $\mathcal{Z}$, and $\mathcal{V}$ are symmetric matrices. Application of the ADMM algorithm gives

$$\begin{aligned}(\mathcal{A})^{k+1} &= \arg\min\nolimits_{\mathcal{A}} L_\rho(\mathcal{A},(\mathcal{Z})^k,(\mathcal{V})^k),\\ (\mathcal{Z})^{k+1} &= \arg\min\nolimits_{\mathcal{Z}} L_\rho((\mathcal{A})^{k+1},\mathcal{Z},(\mathcal{V})^k), \quad (42)\\ (\mathcal{V})^{k+1} &= (\mathcal{V})^k + \rho((\mathcal{A})^{k+1}-(\mathcal{Z})^{k+1}).\end{aligned}$$

The most complex problem in solving Eq. (42) is directly to solve $\mathcal{A}$, therefore, the BCD method will be used to improve efficiency and ensure accuracy, which we call the BCD-ADMM method. Our BCD-ADMM method consists of two steps.

Step 1: Solving $(\mathcal{A})^{k+1} = \arg\min_{\mathcal{A}} L_\rho(\mathcal{A},(\mathcal{Z})^k,(\mathcal{V})^k)$.

Similar to solving the BCD method, solving $\mathcal{A}$ is an iterative process. In particular, given the initial value $\mathcal{A}$, we calculate $\boldsymbol{A}^i$ for each node i according to the following formula until the solution converges:

$$\min \|\mathcal{C}^i\cdot\boldsymbol{A}^i - \boldsymbol{y}^i\|_2^2 + \alpha\|\widetilde{\mathcal{C}^i}\cdot\boldsymbol{A}^i - \widetilde{\boldsymbol{y}^i}\|_2^2 + 2(\boldsymbol{V}^i)^T\cdot(\boldsymbol{A}^i-\boldsymbol{Z}^i) + \rho\|\boldsymbol{A}^i - \boldsymbol{Z}^i\|_2^2. \quad (43)$$

The solution of Eq. (43) is

$$\boldsymbol{A}^i = \tilde{\Psi}^i[\boldsymbol{G}^i + \alpha(\boldsymbol{B}^i - \mathrm{diag}(\Gamma^i\cdot\Delta^i)) + \rho\boldsymbol{Z}^i - \boldsymbol{V}^i], \quad (44)$$

with

$$\tilde{\Psi}^i = \left[\mathcal{F}^i + \alpha\cdot\mathrm{diag}\left(\left[F_{i,i}^1,\ldots,F_{i,i}^{i-1},0,F_{i,i}^{i+1},\ldots,F_{i,i}^n\right]^T\right) + \rho\mathcal{I}\right]^{-1}. \quad (45)$$

Step 2: Solving $(\mathcal{Z})^{k+1} = \arg\min_{\mathcal{Z}} L_\rho((\mathcal{A})^{k+1},\mathcal{Z},\mathcal{V}^k)$.

Solving the optimization problem

$$\min \quad \lambda\|\mathcal{Z}\|_1 + \sum_i(\boldsymbol{V}^i)^T\cdot(\boldsymbol{A}^i - \boldsymbol{Z}^i) + \frac{\rho}{2}\|\mathcal{A}-\mathcal{Z}\|_2^2 \quad (46)$$

we obtain

$$\mathcal{Z} = S_{\lambda/\rho}\left(\mathcal{A}+\frac{\mathcal{V}}{\rho}\right) = \left[\left(\mathcal{A}+\frac{\mathcal{V}}{\rho}\right)-\frac{\lambda}{\rho}\mathcal{I}\right]_+ - \left[\left(-\mathcal{A}-\frac{\mathcal{V}}{\rho}\right)-\frac{\lambda}{\rho}\mathcal{I}\right]_+. \quad (47)$$

Solving for $\mathcal{A}$ is thus an iterative process. Given the initial condition for $\mathcal{A}$, we iterate Eq. (42) until it converges, where the number of iterations required is $t_{\max}$. The quantity $(\mathcal{A})^{k+1} = \arg\min_{\mathcal{A}} L_\rho(\mathcal{A},(\mathcal{Z})^k,(\mathcal{V})^k)$ can then be calculated using Eq. (44) using $\tilde{t}_{\max}$ rounds of iteration.

In executing the ADMM algorithm, it is not necessary to demand high accuracy in the iterative process for matrix $\mathcal{A}$ [50]. We can then set $\tilde{t}_{\max} = 1$. The quantities $\tilde{\Psi}^i$, $\boldsymbol{G}^i$, $\boldsymbol{B}^i$, and Γ^i in Eq. (44) can be calculated prior to the iteration process because their calculation is independent of that of $\mathcal{A}$, making it highly efficient. Overall, the computational load of the BCD-ADMM algorithm comes from two components. First, for each node i, the time required to calculate $\tilde{\Psi}^i$, $\boldsymbol{G}^i$, $\boldsymbol{B}^i$, and Γ^i is $O(n^4)$ for $M = O(n)$. Second, the time required of the iteration process is $O(t_{\max}\tilde{t}_{\max}n^3)$. For $\tilde{t}_{\max} = 1$, the total computational time is $O(t_{\max}n^3 + n^4)$ [or $O(t_{\max}n^3 + n^4/p)$ in parallel], which is similar to that of the PBP-ADMM algorithm and considerably lower than that of the global ADMM method.

Taken together, for both the BCD and BCD-ADMM algorithms, the essence is to divide the multiple linear equations into a series of equations, so high efficiency is guaranteed. As the equations are solved one by one, the solutions of the other equations are treated as constraints [e.g., Eqs. (28) and (43)] by incorporating the network symmetry. We can prove that the solutions are in fact the global solution.

IV. NUMERICAL VALIDATION

We test and evaluate the accuracy and efficiency of our BCD or BCD-ADMM method using representative networks and dynamics.

A. Evaluation metrics

We use the following criterion to reconstruct a network from data: If the calculated element $a_{i,j}$ of the adjacency matrix exceeds 0.5, an edge connecting nodes i and j exists; otherwise, such an edge does not exist if $a_{i,j} < 0.5$. The reconstruction accuracy can be measured by the F1 index defined as

$$\mathrm{F1} = \frac{2\times\mathrm{Precision}\times\mathrm{Recall}}{\mathrm{Precision}+\mathrm{Recall}}, \quad (48)$$

where $0 \leqslant \mathrm{F1} \leqslant 1$, $\mathrm{Precision} = \frac{\mathrm{TP}}{\mathrm{TP+FP}}$, $\mathrm{Recall} = \frac{\mathrm{TP}}{\mathrm{TP+FN}}$, with TP, FP, and FN denoting the rates of true positive, false positive, and false negative, respectively, a larger value of F1 indicates a higher reconstruction. accuracy. Perfect reconstruction is achieved for F1 = 1.

B. Experimental results

Figure 1 presents the results of reconstructing the Zachary karate club network with coupled nonlinear oscillatory dynamics. The sparse network has 34 nodes and 78 edges, as shown in Fig. 1(a). (The details of the coupled nonlinear oscillatory dynamics model are given in Appendix A 1). Figure 1(b) shows the reconstruction results from the BCD-ADMM method, where the values of $a_{i,j}$ for the existent edges (blue dots) and nonexistent edges (yellow dots) can be unambiguously distinguished, but the PBP-ADMM method fails this task. Figure 1(c) shows the F1 scores from the BCD-ADMM method for $\alpha = 1$ and $\alpha = 0.05$, as well as the F1 score from the PBP-ADMM method. It can be seen that, for $\alpha = 0.05$, the F1 score reaches its perfect unity value even when the iteration time t is small. For $t > 1000$, the F1 score from the BCD-ADMM method with $\alpha = 1$ reaches the unity value. In contrast, the F1 score from the PBP-ADMM method can never exceed 0.9. Note that the convergence speed and accuracy in solving each linear equation is different with the PBP-ADMM method. For example, the convergence speed is fast and the accuracy is high when solving the linear equations associated with the small degree nodes, but the opposites

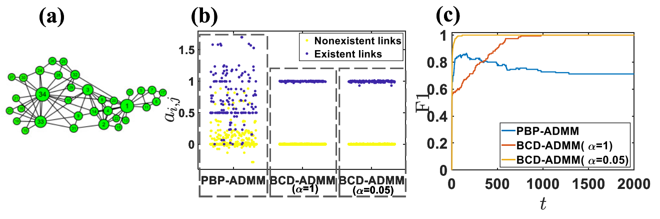

FIG. 1. Reconstruction of the Zachary karate club network with coupled nonlinear oscillatory dynamics. (a) The actual network (the ground truth). (b) The estimated values of the elements of the network adjacency matrix $a_{i,j}$ from the PBP-ADMM (left dashed boxes), BCD-ADMM with $\alpha = 0.05$ (central dashed boxes), and BCD-ADMM with $\alpha = 1$ (right dashed boxes). The horizontal axes in each dashed box represent the IDs of all edges, which are randomly arranged. The length of the time series is $M = 12$, where M is the number of accessible time instants in the coupled nonlinear oscillatory dynamics. The parameter setting is $\tilde{t}_{\max} = 1$ and $t_{\max} = 2000$. (c) F1 score versus the iteration length t for three cases: BCD-ADMM with $\alpha = 0.05$, BCD-ADMM with $\alpha = 1$, and PBP-ADMM.

hold when solving the linear equations with large degree nodes. As a result, the F1 score as the overall reconstruction accuracy for the PBP-ADMM method increases first, then decreases, and finally reaches a constant value.

Figure 1 illustrates the high accuracy of our BCD-ADMM method. The next question is how does the BCD (or BCD-ADMM) method compare with the global (or global ADMM) method in terms of the accuracy? Representative results are shown in Fig. 2, where the differences between $\mathcal{A}_1(t)$ (the reconstructed adjacency matrix from the BCD or BCD-ADMM method) and $\mathcal{A}_2$ (the reconstructed matrix from the global or

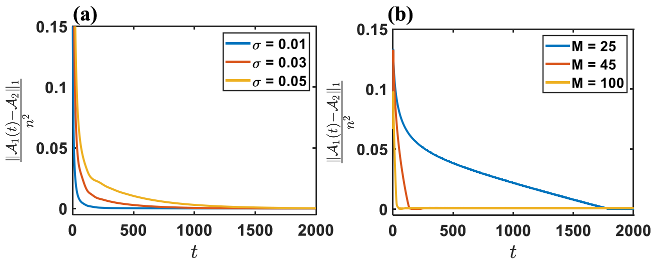

FIG. 2. Performance comparison between the BCD (or BCD-ADMM) and global (or global ADMM) methods. Shown are the differences between the adjacency matrix $\mathcal{A}_1(t)$ reconstructed from the BCD ($\alpha = 1$) or BCD-ADMM ($\alpha = 1$) method using t iterations and the reconstructed matrix $\mathcal{A}_2$ from the global or global ADMM method versus the iteration time t, respectively. (a) Performance comparison of reconstructing a resistance network with current transportation dynamics (see Appendix A 3) between the BCD and global methods, where the network is fully connected (dense network) with $n = 200$ nodes and the link weights follow a uniform distribution in [1.0,3.0]. The parameter setting is the length of the time series $M = 240$, $\xi_i \sim N(0, \sigma^2)$, and $\beta = 0$.(b) Performance comparison of reconstructing an Erdös-Rényi (ER) random network [51] of size $n = 200$ and average degree $\langle k\rangle = 14$ (sparse network) with coupled nonlinear oscillatory dynamics between the BCD-ADMM and global ADMM methods.

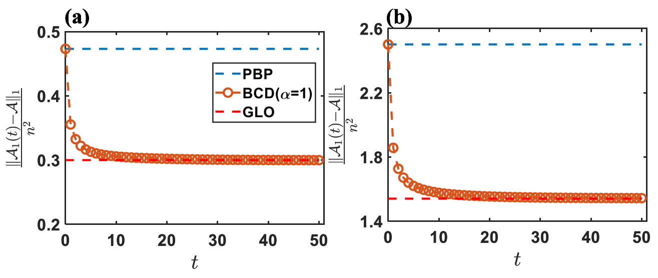

FIG. 3. Monotonic behavior of the accuracy of the BCD method with time. Shown is the reconstruction error $\|\mathcal{A}_1(t) - \mathcal{A}\|_1/n^2$ versus the length t of iteration, based on the time series from current transportation dynamics, where $\mathcal{A}_1(t)$ is the reconstructed adjacency matrix and $\mathcal{A}$ is the ground truth. The network and other parameter values are the same as those in Fig. 2. Panels (a) and (b) are for the noise amplitude $\sigma = 0.01$ and $\sigma = 0.05$, respectively.

global ADMM method) as a function of the iteration time are shown. It can be seen that the differences approach zero, indicating that our BCD and BCD-ADMM methods converges to the solution of the global and global ADMM methods, respectively, and thereby validating the theoretical analysis in Theorem 1.

For general networks without any sparsity constraint, the BCD method enables us to control the accuracy by the number of iterations. In particular, comparing the computational load of the BCD method $O(t_{\max}n^3 + n^4)$ with that of the PBP method $O(n^4)$, we see that an extra term of the BCD method comes from the iteration, where the time needed to carry out each iteration step is $O(n^3)$ but this is one order of magnitude smaller than the computational load of the PBP method. In fact, it can be proven (in Appendix B) that the accuracy of the BCD method increases with time monotonically. Numerical evidence supporting this result is shown in Fig. 3. Since the computational loads of the BCD and PBP methods are of the same order of magnitude, this feature of the BCD method suggests an effective way to obtain a highly accurate reconstruction result: Performing the PBP method first and setting the outcome as the initial condition for the BCD method. The final reconstruction accuracy can then be controlled by the number of iterations of the BCD method, as the computational load required to carry out each iteration is only $O(n^3)$.

As most real-world networks meet the sparsity condition, we study the performance of the BCD-ADMM method in comparison with the PBP-ADMM or the global ADMM method. Representative results are shown in Fig. 4 for a dynamical scale-free network, where it can be seen that the convergence speed of the BCD-ADMM method improves as the value of α decreases but at the expense of decreased accuracy. It is thus useful to set $\alpha = 1/M$. Figure 4 indicates that the BCD-ADMM method gives not only higher reconstruction accuracy than that of the PBP-ADMM method but also faster convergence.

To further test the general advantages of the BCD-ADMM method, we compare its performance with that of the PBP-ADMM and global methods for different kinds of synthetic networks and dynamical processes. Figure 5 shows the F1 score versus the length M of the time series for two types of networks (scale-free and random) and two kinds of dynamical

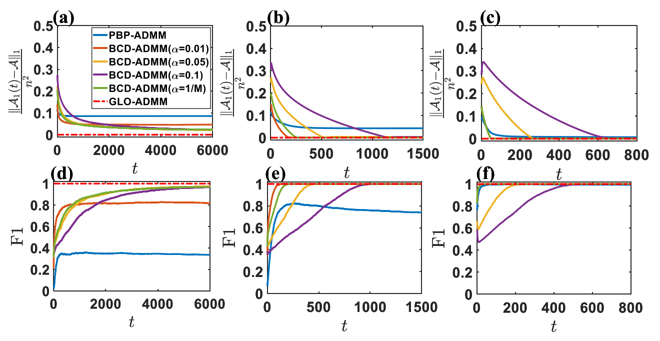

FIG. 4. Performance comparison of BCD-ADMM, PBP-ADMM, or global ADMM methods for sparse complex networks. The time series data are generated from a coupled nonlinear oscillatory dynamics, scale-free network [52] of size $n = 200$ and average degree $\langle k\rangle = 14$. (a)–(c) Reconstruction error $\|\mathcal{A}_1(t) - \mathcal{A}\|_1/n^2$ versus the length t of iteration for the length of the time series $M = 25$, $M = 45$, and $M = 100$, respectively. (d)–(f) The corresponding F1 scores.

processes (coupled oscillatory and evolutionary-game dynamics, the details of the dynamical network model are given in Appendix A). It can be seen that the value of F1 increases with M for all three methods, but the critical values of M required to achieve error-free reconstruction with the BCD-ADMM and global ADMM methods are approximately the same but much smaller than that with the PBP-ADMM

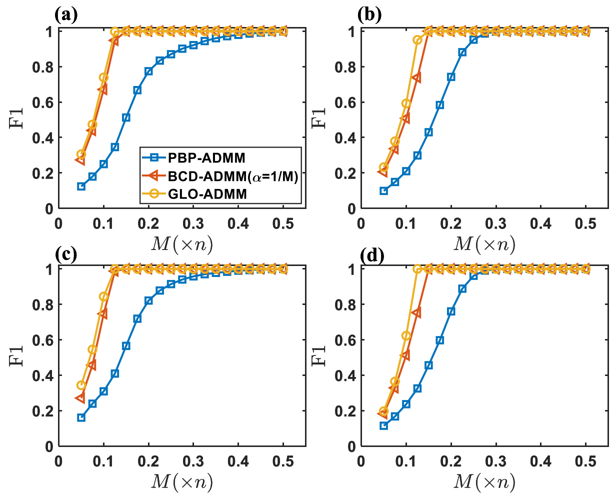

FIG. 5. Effects of the length of the time series on reconstruction. Presented is performance comparison of BCD-ADMM, PBP-ADMM, and global ADMM methods for two types of complex networks and two kinds of dynamical processes in terms of the F1 score. The left and right columns are for a scale-free and a random network, respectively, whereas the top and bottom rows are for coupled nonlinear oscillatory and evolutionary-game dynamics, respectively. Both networks have $n = 200$ nodes and average degree $\langle k\rangle = 14$. Each data point is the result of averaging over five independent realizations. In all cases, BCD-ADMM and global ADMM require similar and relatively short time series to achieve zero reconstruction error, but the PBP-ADMM methods require much longer time series.

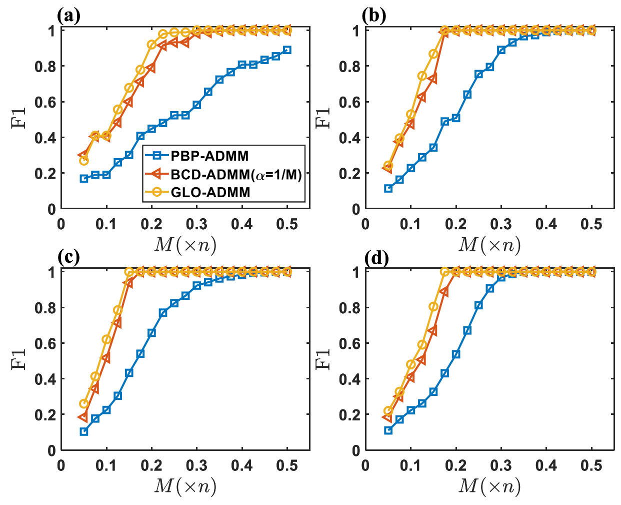

FIG. 6. Performance comparison of BCD-ADMM, PBP-ADMM, and global ADMM methods for real world networks. Shown are the F1 scores versus the length M of time series for the three methods for (a) Zachary karate club network, (b) Dolphins network, (c) Polbooks network, (d) US college football network. Each data point is the result of averaging over five independent realizations. The time series data are generated from a coupled nonlinear oscillatory dynamics. Behaviors similar to those for synthetic networks in Fig. 5 are observed.

methods. Since the computational load of the global ADMM method is much higher than that of the BCD-ADMM method, the results in Fig. 5 suggest that, among the three methods, the BCD-ADMM method is the best in terms of accuracy and computational load. The same conclusion can be drawn based on the performance comparison results from four real-world networks: The Zachary karate club network, Dolphins network, Polbooks network, and US college football network, as shown in Fig. 6.

V. CONCLUSIONS

Reconstructing complex networks from data is a challenging inverse problem in network science. Various methods have been proposed to deal with reconstruction problems arising from different applications, but a common mathematical framework is lacking. Here, we view the problem of network reconstruction as one that solves multiple linear equations and exploit the symmetry in the network to develop a generally applicable, accurate, and efficient solution method based on the principle of BCD (block coordinate descent). Table I summarizes the main results in comparison with the previous PBP (point-by-point) and global methods. For sparse networks, the performance of all three types of methods can be further enhanced through ADMM (alternating direction method of multipliers). Our mathematical analysis and numerical tests indicate that, among the three types of methods, our BCD-based method has the best performance in terms of accuracy and computational load. In particular, the PBP or PBP-ADMM method [3,13] has low computational cost but the accuracy is also low. The global or global-ADMM method [38] has high accuracy but the computational cost is

TABLE I. Comparison of accuracy and efficiency of different algorithms.

		Time complexity	Accuracy	Convergence speed
Without sparsity constraint	PBP	$O(n^4)$	Low	
	Global	$O(n^6)$	High	
	BCD	$O(t_{\max}n^3 + n^4)$	Same as global	
With sparsity constraint	PBP-ADMM	$O(t_{\max}n^3 + n^4)$	Low	Fast
	Global-ADMM	$O(t_{\max}n^4 + n^6)$	High	
	BCD-ADMM ($\alpha = 1$)	$O(t_{\max}n^3 + n^4)$	Same as global-ADMM	Slow
	BCD-ADMM ($\alpha = 1/M$)	$O(t_{\max}n^3 + n^4)$	Higher than PBP-ADMM and close to global-ADMM	Close to PBP-ADMM

overwhelming, making it infeasible for large networks. Figure 7(a) shows that global methods become computationally infeasible when the network size reaches 400. Our articulated BCD or BCD-ADMM method has the advantages of the PBP and global methods but overcomes their shortcomings. In fact, computationally, our BCD or BCD-ADMM method is as accurate as the global method, but the efficiency is far higher than that of the global method. Figure 7(b) presents the results of reconstructing the Power network [53], and the network has 1723 nodes and 4117 edges. The results in Fig. 7(b) indicate that the GLO-ADMM method has become completely infeasible owing to its huge time complexity. However, the BCD-ADMM method achieves accurate reconstruction when the length of time-series data $M = 25$, whereas the PBP-ADMM algorithm needs $M = 50$ to achieve the same level of accuracy. For complex networks with symmetry, among the three types of methods, our BCD-based one stands out as a viable choice for solving the reconstruction problem because the method fully exploits the advantages brought upon by the symmetric properties. As described in the introduction, our framework cannot be applied to the situation where the matrix variables that need to be solved to represent the network structure are not symmetric even if the network structure is symmetric. Some symmetric information may be useful, for example, $a_{i,j} = a_{j,i} = 0$. How to use this potential symmetric information to expand our framework is worth investigating. In addition, solving a large number of linear equations is a universal problem arising from many different fields in science and engineering, and we expect our method to have broader applications beyond network reconstruction.

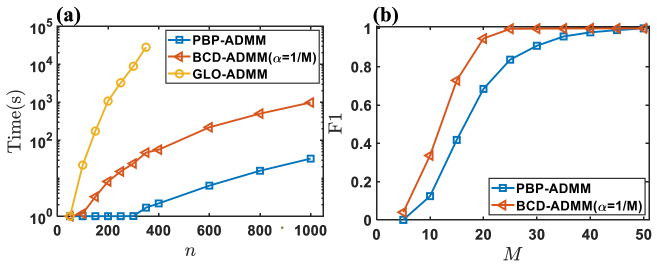

FIG. 7. (a) Runtime comparison of BCD-ADMM, PBP-ADMM, and global ADMM methods. Shown is the runtime versus the network size n. The time series data are generated from an evolutionary-game dynamics on a scale-free network with average degree $\langle k\rangle = 14$. The length of the time series is set to $M = 0.4n$, and the runtime is limited to 5×10^5 seconds. (b) Performance comparison BCD-ADMM and PBP-ADMM methods for Power network size $n = 1723$. The time series data are generated from a current transportation dynamics.

ACKNOWLEDGMENTS

This work was partly supported by the National Natural Science Foundation of China (Grants No. 12005001, No. 61973001, and No. 71731004), and the Natural Science Foundation of Anhui Province China (Grant No. 2008085QF299), and the support from the Fundamental Research Funds for the Central Universities, China. The work at Arizona State University was supported by AFOSR under Grant No. FA9550-21-1-0438.

APPENDIX A: NETWORK RECONSTRUCTION WITH DIFFERENT TYPES OF DYNAMICAL PROCESSES

1. Coupled nonlinear oscillatory dynamics

A complex network of n nodes with coupled nonlinear oscillatory dynamics is described by [1]

$$\dot{\boldsymbol{x}}_i = \boldsymbol{f}_i(\boldsymbol{x}_i) + \sum_{j=1}^{n} a_{i,j}\boldsymbol{g}_{i,j}(\boldsymbol{x}_i, \boldsymbol{x}_j), \tag{A1}$$

where $\boldsymbol{x}_i = [x_i^{(1)}, x_i^{(2)}, \dots, x_i^{(D)}]^T \in \mathbb{R}^D$ is the state vector of the ith node, and the functions $\boldsymbol{f}_i$: $\mathbb{R}^D \to \mathbb{R}^D$ and $\boldsymbol{g}_{i,j}$: $\mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}^D$ govern the intrinsic and interaction dynamics of node i, respectively, $a_{i,j}$ is an element of the adjacency matrix $\mathcal{A}$ of an undirected network: $a_{i,j} = a_{j,i} = 1$ if there is an edge connecting nodes i and j, and $a_{i,j} = a_{j,i} = 0$ otherwise. The network reconstruction task entails obtaining solutions of the adjacency matrix $\mathcal{A}$.

For the dth component of the state vector at node i, the dynamical equation is

$$\dot{x}_i^{(d)}(\tau_m) = f_i^{(d)}(x_i(\tau_m)) + \sum_{j=1}^{n} a_{i,j} g_{i,j}^{(d)}(x_i(\tau_m), x_j(\tau_m)). \tag{A2}$$

By sampling the state vector at a certain time and at the nearby time, one can set $\tau_m = (t_{m-1} + t_m)/2$ to obtain the estimates of the state variable and its derivative at time τ_m:

$$x_i^{(d)}(\tau_m) = \frac{x_i^{(d)}(t_{m-1}) + x_i^{(d)}(t_m)}{2}, \tag{A3}$$

$$\dot{x}_i^{(d)}(\tau_m) = \frac{x_i^{(d)}(t_m) - x_i^{(d)}(t_{m-1})}{t_m - (t_{m-1})}. \tag{A4}$$

If there are M such times, we have the following set of linear equations:

$$\dot{x}_{i,m}^{(d)} = f_{i,m}^{(d)} + \sum_{j=1}^{N} a_{i,j} g_{i,j,m}^{(d)} \; (m = 1, 2, \ldots, M), \tag{A5}$$

which can be conveniently written in the matrix form:

$$\mathcal{C}^i \cdot \boldsymbol{A}^i = \boldsymbol{y}^i, \tag{A6}$$

where

$$\boldsymbol{A}^i \equiv [a_{i,1}, a_{i,2}, \ldots, a_{i,n}]^T,$$

$$\boldsymbol{y}^i \equiv \left[0, \dot{x}_{i,1}^{(d)} - f_{i,1}^{(d)}, \dot{x}_{i,2}^{(d)} - f_{i,2}^{(d)}, \ldots, \dot{x}_{i,m}^{(d)} - f_{i,m}^{(d)}\right]^T, \tag{A7}$$

$$\mathcal{C}^i \equiv \begin{bmatrix} 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \\ g_{i,1,1}^{(d)} & \cdots & g_{i,i-1,1}^{(d)} & 0 & g_{i,i+1,1}^{(d)} & \cdots & g_{i,N,1}^{(d)} \\ g_{i,1,2}^{(d)} & \cdots & g_{i,i-1,2}^{(d)} & 0 & g_{i,i+1,2}^{(d)} & \cdots & g_{i,N,2}^{(d)} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ g_{i,1,M}^{(d)} & \cdots & g_{i,i-1,M}^{(d)} & 0 & g_{i,i+1,M}^{(d)} & \cdots & g_{i,N,M}^{(d)} \end{bmatrix}. \tag{A8}$$

In our numerical examples, we use the following chaotic Rössler system as the individual nodal dynamical process:

$$\begin{aligned} \dot{x}_i &= -y_i - z_i + \textstyle\sum_{j=1}^{n} a_{i,j} f(x_i, x_j), \\ \dot{y}_i &= x_i + a_i y_i, \\ \dot{z}_i &= b_i + z_i(x_i - c_i), \end{aligned} \tag{A9}$$

where $a_i = 0.15$, $b_i = 0.2$, $c_i = 10$, and $f(x_i, x_j) = x_i - x_j$.

2. Evolutionary-game dynamics

Many complex systems in biology, social sciences, and economics can be modeled by evolutionary-game dynamics. In a typical evolutionary game, at any time one agent can choose one of two strategies: Cooperation (C) or defection (D), which can be represented by the vectors $\boldsymbol{S}(C) = [1, 0]^{\mathrm{T}}$ and $\boldsymbol{S}(D) = [0, 1]^{\mathrm{T}}$, respectively. The payoffs of the two players in a game are determined by their strategies and the payoff matrix of the specific game. In our study, we use the snowdrift game with the 2×2 payoff matrix

$$\mathbb{P} = \begin{pmatrix} 1 & 1-b \\ 1+b & 0 \end{pmatrix},$$

where the parameter b ($0 < b < 1$) characterizes the temptation to defect. (We set $b = 0.7$ in our numerical experiments.)

At each time step, all agents play the game with their neighbors and gain payoffs. The payoff of agent i at time step t is [13]

$$P_i(t) = \sum_{j=1}^{n} a_{i,j} \boldsymbol{S}_i^{\mathrm{T}}(t) \cdot \mathbb{P} \cdot \boldsymbol{S}_j(t). \tag{A10}$$

After obtaining the payoff, each agent updates its strategy (i.e., C or D) at the next round to try to maximize its payoff based on some updating rule. We use the Fermi rule for strategy updating, where node i adopts the strategy of neighbor j with the probability

$$\eta[\boldsymbol{S}_i(t+1) \leftarrow \boldsymbol{S}_j(t)] = \frac{1}{1 + \exp\{[P_i(t) - P_j(t)]/\kappa\}}, \tag{A11}$$

where $\kappa = 0.1$ characterizes the irrationality of agents.

After M rounds of the snowdrift game, the strategy and payoff of each agent are collected. This leads to a set of M linear equations:

$$P_i(t_m) = \sum_{j=1}^{n} a_{i,j} \boldsymbol{S}_i^{\mathrm{T}}(t_m) \cdot \mathbb{P} \cdot \boldsymbol{S}_j(t_m) \; (m = 1, 2, \ldots, M), \tag{A12}$$

which can be expressed in the matrix form $\mathcal{C}^i \cdot \boldsymbol{A}^i = \boldsymbol{y}^i$ with

$$\boldsymbol{A}^i = [a_{i,1}, a_{i,2}, \ldots, a_{i,n}]^{\mathrm{T}},$$

$$\boldsymbol{y}^i = [0, P_i(t_1), P_i(t_2), \ldots, P_i(t_M)]^T,$$

$$\mathcal{C}^i = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & \cdots & 0 \\ L_{i,1,1} & \cdots & L_{i,i-1,1} & 0 & L_{i,i+1,1} & \cdots & L_{i,N,1} \\ L_{i,1,2} & \cdots & L_{i,i-1,2} & 0 & L_{i,i+1,2} & \cdots & L_{i,N,2} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ L_{i,1,M} & \cdots & L_{i,i-1,M} & 0 & L_{i,i+1,M} & \cdots & L_{i,N,M} \end{bmatrix}, \tag{A13}$$

where $L_{i,j,m} = \boldsymbol{S}_i^{\mathrm{T}}(t_m) \cdot \mathbb{P} \cdot \boldsymbol{S}_j(t_m)$.

3. Current transportation dynamics

We consider a network of resistors on which current transportation dynamics occur. The resistance between nodes i and j is $r_{i,j}$, and $r_{i,j} = +\infty$ if nodes i and j are not directly connected. For node i, if the resistances of the links connected to it and the relevant voltages are known, the current at i can be calculated according to Kirchhoff's law [16] as

$$\sum_{j=1}^{n} \frac{a_{i,j}}{r_{i,j}} (V_i - V_j) = I_i. \tag{A14}$$

For alternating current, the voltage of node i at time t is

$$V_i(t) = \bar{V} \sin[(\omega + \Delta\omega_i)t], \tag{A15}$$

where $\bar{V}$ is the peak of voltage, $\omega = 10^3$ is the frequency, and $\Delta\omega_i \in [0, 20]$ is the perturbation.

Collecting the current and voltage of each node at M time steps leads to the following set of linear equations:

$$\sum_{j=1}^{n} \frac{a_{i,j}}{r_{i,j}}[V_i(t_m) - V_j(t_m)] = I_i(t_m)\ (m = 1, 2, \ldots, M), \quad \text{(A16)}$$

which can be expressed in the standard matrix form $\mathcal{C}^i \cdot \boldsymbol{A}^i = \boldsymbol{y}^i$ with

$$\boldsymbol{A}^i = \left[\frac{a_{i,1}}{r_{i,1}}, \frac{a_{i,2}}{r_{i,2}}, \ldots, \frac{a_{i,n}}{r_{i,n}}\right]^{\mathrm{T}},$$

$$\boldsymbol{y}^i = [0, I_i(t_1), I_i(t_2), \ldots, I_i(t_M)]^T,$$

$$\mathcal{C}^i = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & \cdots & 0 \\ L_{i,1,1} & \cdots & L_{i,i-1,1} & 0 & L_{i,i+1,1} & \cdots & L_{i,N,1} \\ L_{i,1,2} & \cdots & L_{i,i-1,2} & 0 & L_{i,i+1,2} & \cdots & L_{i,N,2} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ L_{i,1,M} & \cdots & L_{i,i-1,M} & 0 & L_{i,i+1,M} & \cdots & L_{i,N,M} \end{bmatrix}, \quad \text{(A17)}$$

where $L_{i,j,m} = V_i(t_m) - V_j(t_m)$. To better mimic a real situation, we assume that the recorded data are corrupted by Gaussian white noise: $V_i(t_m)+\xi_i^{(V)}$ and $I_i(t_m)+\xi_i^{(I)}$, where $\xi_i^{(V)}$ and $\xi_i^{(I)}$ are Gaussian random processes of zero mean and variance σ^2.

APPENDIX B: CONVERGENCE PROOF OF BLOCK COORDINATE DESCENT METHOD

Our BCD or BCD-ADMM method is designed to solve the following two optimization problems:

$$\min \sum_i \|\mathcal{C}^i \cdot \boldsymbol{A}^i - \boldsymbol{y}^i\|_2^2 + \frac{\beta}{2}\|\mathcal{A}\|_2^2 \quad \text{(B1)}$$

and

$$(\mathcal{A})^{k+1} = \arg\min_{\mathcal{A}} L_\rho(\mathcal{A}, (\mathcal{Z})^k, (\mathcal{V})^k), \quad \text{(B2)}$$

which can be expressed as the following optimization problem:

$$\min \quad f(\widehat{\boldsymbol{A}}), \quad \text{(B3)}$$

where the vector $\widehat{\boldsymbol{A}}=[a_{1,1}, \ldots, a_{1,n}, a_{2,2}, \ldots, a_{2,n}, \ldots, a_{n-1,n}, a_{n,n}]^T$, and the problem is a convex optimization problem. Setting the initial values for some variables, i.e., $\widehat{\boldsymbol{A}}_0^{0*} = [0, \ldots, 0, 0, \ldots, 0, \ldots, 0, 0]^T$. Calculating the values of $a_{1,1}, \ldots, a_{1,n}$ in Eq. (B3) by fixing the values of $a_{j,2}, a_{j,3}, \ldots, a_{j,n}$ to the value in $\widehat{\boldsymbol{A}}_0^{0*}$ for $j = 2, 3, \ldots, n$, the optimization problem becomes

$$\min \quad f([a_{1,1}, \ldots, a_{1,n}, 0, \ldots, 0, \ldots, 0, 0]^T), \quad \text{(B4)}$$

whose solution can be denoted as $[a_{1,1}^*, \ldots, a_{1,n}^*]^T$, and the solution of $\widehat{\boldsymbol{A}}$ at this time is $\widehat{\boldsymbol{A}}_1^{1*} = [a_{1,1}^*, \ldots, a_{1,n}^*, 0, \ldots, 0, \ldots, 0, 0]^T$, which satisfies

$$f(\widehat{\boldsymbol{A}}_1^{1*}) \leqslant f(\widehat{\boldsymbol{A}}_0^{0*}).$$

In the same way, $a_{2,1} = a_{1,2}, a_{2,2}, \ldots, a_{2,n}$ for Eq. (B3) can be solved by using the solution $\widehat{\boldsymbol{A}}_1^{1*}$ in the previous step to fix other variable values, and the result of $\widehat{\boldsymbol{A}}$ is $\widehat{\boldsymbol{A}}_2^{1*} = [a_{1,1}^*, a_{2,1}^*, a_{1,3}^*, \ldots, a_{1,n}^*, a_{2,2}^*, \ldots, a_{2,n}^*, 0, \ldots, 0, \ldots, 0, 0]^T$. By iteration, we can obtain $\widehat{\boldsymbol{A}}_1^{1*}, \widehat{\boldsymbol{A}}_2^{1*}, \ldots, \widehat{\boldsymbol{A}}_n^{1*}$, which satisfies

$$f(\widehat{\boldsymbol{A}}_n^{1*}) \leqslant \cdots \leqslant f(\widehat{\boldsymbol{A}}_2^{1*}) \leqslant f(\widehat{\boldsymbol{A}}_1^{1*}) \leqslant f(\widehat{\boldsymbol{A}}_0^{0*}).$$

So far, the first iteration round is over and a better solution $\widehat{\boldsymbol{A}}_n^{1*}$ is obtained compared with the initial solution $\widehat{\boldsymbol{A}}_0^{0*}$. Similarly, a better solution $\widehat{\boldsymbol{A}}_n^{2*}$ can be obtained next round. We can stop the iteration process when $\|\widehat{\boldsymbol{A}}_n^{k+1*} - \widehat{\boldsymbol{A}}_n^{k*}\|_2^2 \leqslant \epsilon$ (e.g., $\epsilon = 0.001$).

Note that Eqs. (B1) and (B2) represent a convex optimization problem and the solutions obtained by the BCD and BCD-ADMM methods are optimal and consistent with those of the global and global ADMM methods, respectively.

[1] M. Timme and J. Casadiego, J. Phys. A: Math. Theor. **47**, 343001 (2014).

[2] W.-X. Wang, Y.-C. Lai, and C. Grebogi, Phys. Rep. **644**, 1 (2016).

[3] S. G. Shandilya and M. Timme, New J. Phys. **13**, 013004 (2011).

[4] J. Ren, W.-X. Wang, B. Li, and Y.-C. Lai, Phys. Rev. Lett. **104**, 058701 (2010).

[5] J. Casadiego, D. Maoutsa, and M. Timme, Phys. Rev. Lett. **121**, 054101 (2018).

[6] J.-G. Young, G. Petri, and T. P. Peixoto, Commun. Phys. **4**, 1 (2021).

[7] G. Stepaniants, B. W. Brunton, and J. N. Kutz, Phys. Rev. E **102**, 042309 (2020).

[8] H. Wang, C. Ma, H.-S. Chen, Y.-C. Lai, and H.-F. Zhang, Nat. Commun. **13**, 3043 (2022).

[9] T.-T. Gao and G. Yan, Nat. Comput. Sci. **2**, 160 (2022).

[10] T. S. Gardner, D. Di Bernardo, D. Lorenz, and J. J. Collins, Science **301**, 102 (2003).

[11] G. B. Giannakis, Y. Shen, and G. V. Karanikolas, Proc. IEEE **106**, 787 (2018).

[12] G. Mateos, S. Segarra, A. G. Marques, and A. Ribeiro, IEEE Signal Process. Mag. **36**, 16 (2019).

[13] W.-X. Wang, Y.-C. Lai, C. Grebogi, and J. Ye, Phys. Rev. X **1**, 021021 (2011).

[14] K. Champion, P. Zheng, A. Y. Aravkin, S. L. Brunton, and J. N. Kutz, IEEE Access **8**, 169259 (2020).

[15] R.-Q. Su, X. Ni, W.-X. Wang, and Y.-C. Lai, Phys. Rev. E **85**, 056220 (2012).
[16] X. Han, Z. Shen, W.-X. Wang, and Z. Di, Phys. Rev. Lett. **114**, 028701 (2015).
[17] Y.-Z. Chen and Y.-C. Lai, Phys. Rev. E **97**, 032317 (2018).
[18] J. Li, Z. Shen, W.-X. Wang, C. Grebogi, and Y.-C. Lai, Phys. Rev. E **95**, 032303 (2017).
[19] C. Ma, H.-S. Chen, Y.-C. Lai, and H.-F. Zhang, Phys. Rev. E **97**, 022301 (2018).
[20] C. Ma, H.-S. Chen, X. Li, Y.-C. Lai, and H.-F. Zhang, J. Appl. Dyn. Syst. **19**, 124 (2020).
[21] Z. Shen, W.-X. Wang, Y. Fan, Z. Di, and Y.-C. Lai, Nat. Commun. **5**, 4323 (2014).
[22] H.-F. Zhang, F. Xu, Z.-K. Bao, and C. Ma, IEEE Trans. Circuits Syst. I **66**, 1608 (2019).
[23] F. Basiri, J. Casadiego, M. Timme, and D. Witthaut, Phys. Rev. E **98**, 012305 (2018).
[24] M. Nitzan, J. Casadiego, and M. Timme, Sci. Adv. **3**, e1600396 (2017).
[25] J. Casadiego, M. Nitzan, S. Hallerberg, and M. Timme, Nat. Commun. **8**, 2192 (2017).
[26] F. Chung, L. Lu, T. G. Dewey, and D. J. Galas, J. Comput. Biol. **10**, 677 (2003).
[27] B. D. MacArthur and R. J. Sánchez-García, Phys. Rev. E **80**, 026117 (2009).
[28] B. D. MacArthur, R. J. Sánchez-García, and J. W. Anderson, Discrete Appl. Math. **156**, 3525 (2008).
[29] V. Nicosia, M. Valencia, M. Chavez, A. Díaz-Guilera, and V. Latora, Phys. Rev. Lett. **110**, 174102 (2013).
[30] L. M. Pecora, F. Sorrentino, A. M. Hagerstrom, T. E. Murphy, and R. Roy, Nat. Commun. **5**, 1 (2014).
[31] R. J. Sánchez-García, Commun. Phys. **3**, 87 (2020).
[32] F. Sorrentino, A. B. Siddique, and L. M. Pecora, Chaos **29**, 011101 (2019).
[33] Y. Xiao, B. D. MacArthur, H. Wang, M. Xiong, and W. Wang, Phys. Rev. E **78**, 046102 (2008).
[34] Y. Xiao, M. Xiong, W. Wang, and H. Wang, Phys. Rev. E **77**, 066108 (2008).
[35] M. Golubitsky and I. Stewart, *The Symmetry Perspective: From Equilibrium to Chaos in Phase Space and Physical Space* (Springer Science & Business Media, Berlin, 2003), Vol. 200.
[36] M. Golubitsky and I. Stewart, Bull. Am. Math. Soc. **43**, 305 (2006).
[37] M. Golubitsky and I. Stewart, Chaos **25**, 097612 (2015).
[38] Z. Hang, P. Dai, S. Jia, and Z. Yu, Chaos Solit. Fractals **139**, 110287 (2020).
[39] J. Tegner, M. K. S. Yeung, J. Hasty, and J. J. Collins, Proc. Natl. Acad. Sci. USA **100**, 5944 (2003).
[40] M. K. S. Yeung, J. Tegnér, and J. J. Collins, Proc. Natl. Acad. Sci. USA **99**, 6163 (2002).
[41] Y.-C. Lai, Chaos **31**, 082101 (2021).
[42] G. Li, X. Wu, J. Liu, J.-A. Lu, and C. Guo, Chaos **25**, 043102 (2015).
[43] J. Liu, G. Mei, X. Wu, and J. Lu, IEEE Trans. Circuits Syst. I **65**, 2970 (2018).
[44] Q.-M. Liu, C. Ma, B.-B. Xiang, H.-S. Chen, and H.-F. Zhang, IEEE Trans. Syst. Man Cybern, Syst. **51**, 4639 (2021).
[45] M. Timme, Phys. Rev. Lett. **98**, 224101 (2007).
[46] W.-X. Wang, R. Yang, Y.-C. Lai, V. Kovanis, and C. Grebogi, Phys. Rev. Lett. **106**, 154101 (2011).
[47] X. Wang, J. Lü, and X. Wu, IEEE Trans. Syst. Man Cybern, Syst. **50**, 2588 (2020).
[48] B.-B. Xiang, Z.-K. Bao, C. Ma, X. Zhang, H.-S. Chen, and H.-F. Zhang, Chaos **28**, 013122 (2018).
[49] X. Han, Z. Shen, W.-X. Wang, Y.-C. Lai, and C. Grebogi, Sci. Rep. **6**, 30241 (2016).
[50] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, Foundations and Trends in Machine learning **3**, 1 (2011).
[51] P. Erdõs and A. Rényi, Publ. Math. Inst. Hung. Acad. Sci **5**, 17 (1960).
[52] A.-L. Barabási and R. Albert, Science **286**, 509 (1999).
[53] R. A. Rossi and N. K. Ahmed, in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, Vol. 29 (Association for the Advancement of Artificial Intelligence, 2015), p. 1.