Computing partial synchronization manifolds of delay-coupled systems

L. Su*, W. Michiels*, E. Steur**,*** and H. Nijmeijer***
*Department of Computer Science, KU Leuven
**Institute for Complex Molecular Systems, Eindhoven University of Technology
***Department of Mechanical Engineering, Eindhoven University of Technology

Summary. Due to weak couplings, large time-delays, etc., a network of interconnected dynamical systems may exhibit a phenomenon of incomplete synchronization where some but not all systems behave synchronously. The phenomenon is called *partial synchronization* or *cluster synchronization*. To describe the pattern of partial synchronization, so-called partial synchronization manifolds are used, which are linear invariant subspaces of $\mathcal{C}$, the state space of the network of systems. Here, we focus on partial synchronization manifolds in networks of identical systems interacting via linear diffusive coupling with inclusion of time-delays. Based on a recently proposed existence criterion for partial synchronization manifolds in terms of the block structure of a reordered adjacency matrix, we present an improved algorithm for computing partial synchronization manifolds, particularly, for networks with invasive delayed coupling whose coupling term does not vanish when relevant systems are synchronized. It is shown that the computational cost is largely reduced using the improved algorithm when computing the partial synchronization manifolds of networks with invasive coupling.

Introduction

Synchronization of coupled dynamical systems has been observed by the scientific community since centuries ago. In the 17th century, the Dutch scientist Christian Huygens recorded the synchronous motion of two pendulum clocks attached to one beam [5], see Figure 1. During the recent decades, much attention has been driven to this topic by the need to understand complex systems. Examples of synchronization can be found in fields from nature to engineering. Fireflies flash at the same time instants [3]; geese fly in flock during migration [7]; and robots operate in cooperation [6], [8]. The most trivial form of synchronous behavior is full synchronization, which typically refers to the state where all systems of a network behave identically [12]. However, full synchronization is not always achievable. Networks may exhibit some kind of incomplete synchronization due to weak interconnections between systems, large time-delays, etc. This type of incomplete synchronization is called *partial synchronization*, which refers to the phenomenon that some but not all systems in networks synchronize [12]. Partial synchronization is often observed in large networks, for example, coherent activity in parts of human brain neuron system, [4].

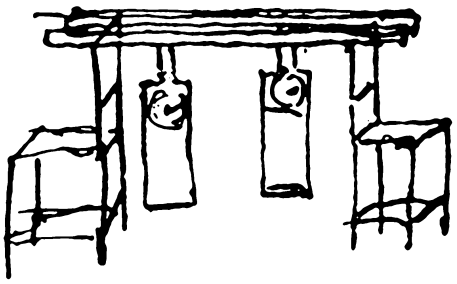

Figure 1: Original Huygens drawing, [5]

Partial synchronization is associated with the existence of *partial synchronization manifolds*, which are linear invariant subspaces of the networked systems' state space. In [10], [9], [11], [12], the existence of partial synchronization manifolds have been studied. Here, we focus on partial synchronization manifolds in networks of identical systems interacting via linear diffusive time-delay coupling described by a weighted graph. In [12], a number of equivalent existence criteria for partial synchronization manifolds for such networks are presented. One of these criteria exploits the block structure of a reordered adjacency matrix to compute partial synchronization manifolds. An algorithm based on it is available in [12]. However, the algorithm can be improved for the case of networks with invasive coupling (whose coupling term does not vanish when the systems are synchronized). The row sums of the original adjacency matrix can be used as a priori to limit the number of generated partitions for the criteria to check. Based on this idea, an improved algorithm to compute all partial synchronization manifolds is developed in this paper.
The structure of the paper is as follows. The three subsequent sections introduce the network of systems, partial synchronization manifolds and existence criteria of partial synchronization manifolds, respectively. These sections are based on [12]. The section after them shows the improved algorithm for computing the partial synchronization manifolds. The next section presents an example to demonstrate the algorithm. The final section includes the concluding remarks.

Network of systems

In this paper, we adopt the settings of [12] by considering the networks of identical systems with linear diffusive time-delay coupling. The networks are represented by directed weighted graphs $\mathcal{G} = (\mathcal{V}, \mathcal{E}, A)$, where

- $\mathcal{V}$ is a finite set of nodes with cardinality $|\mathcal{V}| = N$;
- $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ is the *ordered* set of edges, where the edge (i, j) points from node i to node j;
- $A = \left(a_{ij}\right) \in \mathbb{R}^{N\times N}$ is the weighted adjacency matrix, where $a_{ij} > 0$ represents the weight of edge (i, j) when $(i, j) \in \mathcal{E}$, and $a_{ij} = 0$ when $(i, j) \notin \mathcal{E}$.

Figure 2 shows an example of one such graph $\mathcal{G}$ with the set of nodes

$$\mathcal{V} = \{1, 2, 3, 4, 5\},$$

the set of edges

$$\mathcal{E} = \{(1, 3), (1, 5), (2, 1), (2, 4), (3, 2), (4, 3), (4, 5), (5, 2), (5, 3)\}$$

and the adjacency matrix

$$A = \begin{pmatrix} 0 & 0 & 2 & 0 & 2 \\ 1 & 0 & 0 & 2 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 2 \\ 0 & 1 & 1 & 0 & 0 \end{pmatrix}.$$

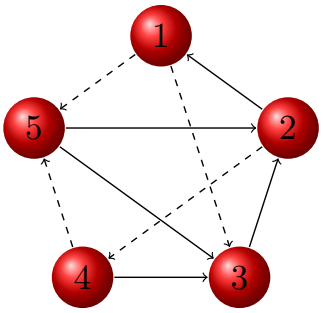

Figure 2: Example Network. Solid edges have weight of 1, dashed edges have weight of 2.

Every node of $\mathcal{G}$ consists of a dynamical system described by

$$\begin{cases} \dot{x}_i(t) = f(x_i(t), u_i(t)) \\ y_i(t) = h(x_i(t)) \end{cases} \tag{1}$$

where $i \in \mathcal{V}$, state $x_i(t) \in \mathbb{R}^n$, input(s) $u_i(t) \in \mathbb{R}^m$, output(s) $y_i(t) \in \mathbb{R}^m$, function $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ and function $h : \mathbb{R}^n \to \mathbb{R}^m$. The systems (1) interact via either one of the following two types of coupling:

$$u_i(t) = \sum_{j\in\mathcal{N}_i} a_{ij}[y_j(t - \tau) - y_i(t)] \tag{2}$$

or

$$u_i(t) = \sum_{j\in\mathcal{N}_i} a_{ij}[y_j(t - \tau) - y_i(t - \tau)]. \tag{3}$$

Here the set $\mathcal{N}_i$ is the *neighbor set* of node $i \in \mathcal{V}$, i.e., $\mathcal{N}_i := \{j \in \mathcal{V}|(i, j) \in \mathcal{E}\}$, a_{ij} are the entries of the weighted adjacency matrix A and τ is the time-delay. These two types of couplings have a fundamental difference: Coupling (2) is called *invasive* coupling as the coupling does not vanish when system i and its neighbors are synchronized; Coupling (3) is called *non-invasive* coupling as the coupling vanishes ($u_i(t) \equiv 0$) when system i and its neighbors are synchronized, [12]. From the time being, we mainly focus on newtorks with invasive coupling.

Note that the graph considered here is simple and strongly connected. For a simple graph $\mathcal{G}$, every pair of its nodes is joined by at most one edge and it does not contain self-loops. For a strongly connected graph $\mathcal{G}$, for each pair of nodes $u, v \in \mathcal{V}$, there exists a directed path from u to v and also a directed path from v to u, [2].

A solution is a partially synchronous solution of the coupled systems (1), (2) or (1), (3) if there exist $i, j \in \mathcal{V}$ with $i \neq j$ such that

$$x_i(t) = x_j(t), \quad \forall t \geq t_0, \tag{4}$$

whenever $x_i(t) = x_j(t)$ for $t \in [t_0 - \tau,\ t_0]$. It is important to note that finding all possible partially synchronous solutions is the first step in the study of partial synchronization in the network. The study of stability of these partially synchronous solutions, which is necessary for partial synchronization in any real-world application, is not considered here.

Partial synchronization manifolds

Consider a $N \times N$ permutation matrix Π. We associate Π with an equivalence relation $\sim$ on the set of nodes $\mathcal{V}$, which is such that $i \sim j$ if the ij^{th} entry of Π is equal to 1. This equivalence relation defines a partition $\mathcal{P}$ of $\mathcal{V}$. We denote the number of parts of the partition $\mathcal{P}$ by K. We will refer to the parts of $\mathcal{P}$ as the clusters of the network. Furthermore, note that $K = \dim\ker(I_N - \Pi)$. For example, for a graph $\mathcal{G}$ with $\mathcal{V} = \{1,2,3,4,5\}$, the partition $\mathcal{P} = \{\mathcal{P}_1, \mathcal{P}_2\} = \{\{1,3,4\}\{2,5\}\}$ can be presented by the following permutation matrix

$$\Pi = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}.$$

Let $\mathcal{C}([-\tau, 0], \mathbb{R}^{Nn})$ be the space of continuous functions that map the interval $[-\tau, 0] \subset \mathbb{R}$ into $\mathbb{R}^{Nn}$. Denoting the state of the coupled network by $x_t \in \mathcal{C}([-\tau, 0], \mathbb{R}^{Nn})$,

$$x_t(\theta) := \begin{pmatrix} x_{t,1}(\theta) \\ x_{t,2}(\theta) \\ \vdots \\ x_{t,N}(\theta) \end{pmatrix} = \begin{pmatrix} x_1(t+\theta) \\ x_2(t+\theta) \\ \vdots \\ x_N(t+\theta) \end{pmatrix}, \quad \theta \in [-\tau, 0],$$

i.e., the function segment $x(t+\theta),\ \theta \in [-\tau,\ 0]$, conditions of the form (4) can then be expressed as

$$x_t(\theta) = (\Pi \otimes I_n) x_t(\theta), \quad \forall \theta \in [-\tau, 0], \quad \forall t \geq t_0, \tag{5}$$

or $x_t \in \mathcal{M}(\Pi)$, $\forall t \geq t_0$ whenever $x_{t_0} \in \mathcal{M}(\Pi)$, where

$$\mathcal{M}(\Pi) := \{\phi \in \mathcal{C}([-\tau, 0], \mathbb{R}^{Nn}) \,|\, \phi(\theta) = \text{col}(\phi_1(\theta), \phi_2(\theta), \ldots, \phi_N(\theta)), \\ \phi_i(\theta) \in \mathbb{R}^n,\ i = 1, \ldots, N,\ \phi(\theta) \in \ker(I_{Nn} - \Pi \otimes I_n)\, \forall \theta \in [-\tau, 0]\}$$

is the set of partially synchronous states induced by the permutation matrix Π. This brings us to the following definition.

Definition 1 *[12] The set $\mathcal{M}(\Pi)$ with permutation matrix Π for which $1 < K < N$ is a partial synchronization manifold for the coupled systems* (1)*,* (2)*, or* (1)*,* (3)*, if and only if it is positively invariant under the dynamics* (1)*,* (2)*, or* (1)*,* (3)*, respectively.*

Existence of partial synchronization manifolds

Given a partition $\mathcal{P}$ or corresponding permutation matrix Π, relabel the nodes of the network by clusters, i.e., the first k_1 nodes belong to cluster 1, the second k_2 nodes belong to cluster 2, and so on,

$$\tilde{\mathcal{P}} = \{\{1, \ldots, k_1\}, \{k_1 + 1, \ldots, k_1 + k_2\}, \ldots\},$$

where $k_\ell, \ell = 1, \ldots, K$ are the dimensions of the parts of $\mathcal{P}$.
Mathematically, this relabeling can be done using another permutation matrix R, which we refer to as the *reordering matrix*. For any $N \times N$ permutation matrix Π with $1 < K < N$, there always exists an $N \times N$ reordering matrix R such that

$$R^\top \Pi R = \begin{pmatrix} \Pi_C(k_1) & & & \\ & \Pi_C(k_2) & & \\ & & \ddots & \\ & & & \Pi_C(k_K) \end{pmatrix}, \quad \sum_{\ell=1}^{K} k_\ell = N, \tag{6}$$

i.e. $R^\top \Pi R$ is a block diagonal matrix with K blocks $\Pi_C(k_\ell)$, each of which is a $k_\ell \times k_\ell$-dimensional cyclic permutation matrix.
Using R, we can construct the *reordered adjacency matrix*

$$R^\top A R = \left(\begin{array}{c|c|c|c} A_{11} & A_{12} & \cdots & A_{1K} \\ \hline A_{21} & A_{22} & \cdots & A_{2K} \\ \hline \vdots & \ddots & \ddots & \vdots \\ \hline A_{K1} & A_{K2} & \cdots & A_{KK} \end{array}\right), \quad A_{ij} \in \mathbb{R}^{k_i \times k_j}. \tag{7}$$

We are now ready to give an explicit condition for existence of a partial synchronization manifold.

Theorem 1 *[12] Given an adjacency matrix A and a permutation matrix Π of the same dimension. Associate with Π a reordering matrix R such that $R^\top \Pi R$ is of the form* (6)*. Assume system* (1) *is left-invertible (the system input-output map is injective), then the following statements are equivalent:*

1) *$\mathcal{M}(\Pi)$ is a partial synchronization manifold for (1) and (2), respectively (1) and (3);*
2) *all blocks, respectively all off-diagonal blocks, of the reordered adjacency matrix* (7)*, partitioned in blocks of size $k_i \times k_j$ according to* (6)*, have constant row-sums.*

In practice, networks may have physically distinct communication channels which have different time-delays and/or interaction weights. Here, the latter case is discussed. The multiple time delays case is discussed in Appendix A.
Suppose a network $\mathcal{G}$ has r characteristic scalar interaction weights $\omega_\ell, \ell = 1, \ldots, r$, we can decompose it as

$$\mathcal{G}(\mathcal{V}, \mathcal{E}, A) = \mathcal{G}_1(\mathcal{V}, \mathcal{E}_1, \omega_1 A_1) \oplus \cdots \oplus \mathcal{G}_r(\mathcal{V}, \mathcal{E}_r, \omega_r A_r), \tag{8}$$

where A_ℓ are adjacency matrices with non-negative integer entries and $\mathcal{E}_\ell$ is the set of edges corresponding to A_ℓ. Here $\oplus$ denotes a sum on graphs with the same set of nodes $\mathcal{V}$, which joins the sets of edges and sums the adjacency matrices. Therefore,

$$\mathcal{G}_1(\mathcal{V}, \mathcal{E}_1, \omega_1 A_1) \oplus \cdots \oplus \mathcal{G}_r(\mathcal{V}, \mathcal{E}_r, \omega_r A_r) = \mathcal{G}\left(\mathcal{V}, \cup_{\ell=1}^r \mathcal{E}_\ell, \sum\nolimits_{\ell=1}^r \omega_\ell A_\ell\right),$$

which means

$$A = \sum_{\ell=1}^r \omega_\ell A_\ell \tag{9}$$

and

$$\mathcal{E} = \cup_{\ell=1}^r \mathcal{E}_\ell. \tag{10}$$

Here, the numbers ω_ℓ, $\ell = 1, \ldots, r$ are called basis weights.
Clearly, satisfying the row sum criteria in item 2 of Theorem 1 for all adjacency matrices A_ℓ simultaneously is a *sufficient* condition for $M(\Pi)$ to be a partial synchronization manifold. However, this condition becomes both *sufficient* and *necessary* when the basis weights are rationally independent. The basis weights are rationally independent if and only if the following implication holds for any integer numbers $q_1, \ldots, q_r$

$$\sum_{\ell=1}^r \omega_\ell q_\ell = 0 \Rightarrow q_\ell = 0 \quad \forall \ell = 1, \ldots, r. \tag{11}$$

Then, we have the following condition for the aforementioned graphs.

Theorem 2 *[12] Consider a graph $\mathcal{G}$ that decomposes as* (8) *and suppose that the numbers $\omega_1, \ldots, \omega_r$ are rationally independent. Given a permutation matrix Π of appropriate dimension and associate with that Π a reordering matrix R such that $R^\top \Pi R$ is of the form* (6)*. Assume system* (1) *is left-invertible, then the following statements are equivalent:*

1) *$\mathcal{M}(\Pi)$ is a partial synchronization manifold for (1) and (2), respectively (1) and (3);*
2) *for each $\ell = 1, \ldots, r$, all blocks, respectively all off-diagonal blocks, of each block-structured matrices $R^\top A_l R$, partitioned in blocks of size $k_i \times k_j$ according to* (6)*, have constant row-sums.*

We remark that a number of equivalent existence conditions of partial synchronization manifolds are presented in [12].

An improved algorithm for computing all partial synchronization manifolds

An algorithm for identifying all partial synchronization manifolds has been presented in [12], which consists of two ingredients: generating possible partitions, and checking the viability of a partition (i.e., checking whether or not it corresponds to a partial synchronization manifold). For the latter, the algorithm checks the row-sum criteria of Theorem 1, or 2. If all elements of A are integers or if A can be decomposed as $A = \sum_{i=1}^r \omega_\ell A_\ell$, with A_ℓ containing integers and $(\omega_1, \ldots, \omega_p)$ rationally independent, only row-sum tests on matrices with *integer elements* need to be made. In this case, the detection algorithm can be carried out in exact arithmetic, without making conditions more stringent.
However, this algorithm scales badly with the network size due to its combinatorial nature (as we shall see, the number of possible partitions of N systems grows with N in an exponential-like way). To restrict the computational cost, the following is done:

- each row-sum test on a reordered adjacency matrix is aborted as soon as a block with non-constant row sums has been detected;

- for the case of coupling (2), a necessary condition for a viable partition is that each block of the (full) reordered adjacency matrix has constant row sums, which indicates that, for any two nodes to be in the same part of a viable partition, it is also necessary that their corresponding rows in the original adjacency matrix have the same row sum. Therefore, we propose to start the algorithm by marking the nodes according to the row sums of the original adjacency matrix. This will be used to restrict the number of possible partitions to be generated.

In the latter way, we improve the algorithm as originally proposed in [12] for computing partial synchronization manifolds of networks interacting via *invasive* coupling. Thus, the first ingredient of the improved algorithm will contain two parts: 1) marking the nodes according to the row sums of the adjacency matrix; 2) generating possible partitions exploiting the marking. The second ingredient (check partition variability by row sum criteria) remains the same as for the original algorithm, therefore we will not present it here. To illustrate the proposed improvement for networks with invasive coupling, we will briefly describe the two parts of the first ingredient and show an example in the next section.
The improved algorithm has been implemented in MATLAB and can be downloaded from
`http://twr.cs.kuleuven.be/research/software/delay-control/manifolds/`

Mark nodes by row sums

As mentioned, the nodes are marked according to the row sums of their corresponding rows in the adjacency matrix. This is done by constructing a vector $A_R = [R_1 \cdots R_N]^\top \in \mathbb{R}^N$ which indicates the (in)equality of the row sums of the adjacency matrix A. Here, $R_i = R_j$ if and only if the i-th row and j-th row of A have the same row sum. A_R will be then used to determine which nodes can be put into the same clusters during the generation of all possible partitions. In what follows, we introduce the procedure to construct A_R for both cases where the adjacency matrix A is treated as a single matrix and where A is decomposed into multiple matrices according to (9).

When the adjacency matrix A is treated as one single matrix, the process to acquire A_R is straightforward. If the adjacency matrix A only contains integer elements, A_R can be determined by simply calculating the row sums of A:

$$A_R = A\mathbf{1}_N,$$

where $\mathbf{1}_N$ is the N-dimensional vector with all entries equal to 1. If the adjacency matrix A contains non-integer element(s), A_R can be obtained by comparing the row sums of A using a predefined tolerance.
Denote the number of distinct elements of A_R by d. If $d = N$, no partial synchronization manifolds exist. The algorithm will stop and return the result stating no partial synchronization manifolds exist.

When the adjacency matrix A is decomposed in the form of (9), some extra effort is needed to acquire A_R.
First, using the same method above, we can obtain the vector $A_{\ell,R}$ indicating the (in)equality of the row sums of each matrix A_ℓ ($\ell = 1, 2, \ldots, r$):

$$A_{\ell,R} = [R_{\ell,1} \ \cdots \ R_{\ell,N}]^\top.$$

Again, $R_{\ell,i} = R_{\ell,j}$ if and only if the i-th and j-th rows of the matrix A_ℓ have the same row sum.
Second, find the common row sum indicator A_R for all adjacency matrices. In A_R, $R_i = R_j$ if and only the i-th and j-th rows of all matrices $A_\ell, \ell = 1, \ldots, r$ have the same row sum.
Start with the 1st row of the matrices. For the 1st row, construct the following vectors for $l = 1, \ldots, r$:

$$S_{\ell,1} = \begin{bmatrix} s_{\ell,1} \\ s_{\ell,2} \\ \vdots \\ s_{\ell,N} \end{bmatrix} \text{ with } s_{\ell,i} = \begin{cases} 1 & \text{for } i = 1 \\ 1 & \text{for } i = 2, \ldots, N \text{ if } R_{l,i} = R_{l,1} \\ 0 & \text{otherwise} \end{cases}.$$

Then, these r vectors are combined into one vector C_1 by element-wise multiplication

$$C_1 = \begin{bmatrix} 1 \\ \prod_{\ell=1}^{r} s_{\ell,2} \\ \vdots \\ \prod_{\ell=1}^{r} s_{\ell,N} \end{bmatrix} = \begin{bmatrix} 1 \\ c_{1,2} \\ \vdots \\ c_{1,N} \end{bmatrix}.$$

Here, the i-th row and the 1th row in every adjacency matrix $(A_1, A_2, \ldots, A_r)$ have the same row sum if and only if $c_{1,i} = 1$ for $i = 2, \ldots, N$.

For rows $i = 2, \ldots, N$, the following procedure is used to obtain the vectors C_i.

Algorithm 1: Calculate C_i for A_R

Input: $A_{\ell,R}, \ell = 1, \ldots, r$ and C_1
Output: $C_i, i = 1, \ldots, d$
$d \leftarrow 1$;
$q \leftarrow 2$;
while $q \leq N$ **do**
  if $\sum_{i=1}^{d} c_{i,q} \neq 0$ **then**
    $q \leftarrow q + 1$;
    continue;
  else
    $d \leftarrow d + 1$;
    construct the following vectors for $\ell = 1, \ldots, r$:

$$S_{\ell,d} = \begin{bmatrix} s_{\ell,1} \\ s_{l,2} \\ \vdots \\ s_{\ell,N} \end{bmatrix} \text{ with } s_{l,i} = \begin{cases} 1 & \text{for } i = q \\ 1 & \text{for } i = 1, \ldots, q-1, q+1, \ldots, N \text{ if } R_{l,i} = R_{l,q}; \\ 0 & \text{otherwise} \end{cases}$$

    combine these r vectors into one vector C_d by element-wise multiplication:

$$C_d = \begin{bmatrix} \prod_{\ell=1}^{r} s_{\ell,1} \\ \vdots \\ \prod_{\ell=1}^{r} s_{\ell,q-1} \\ 1 \\ \prod_{\ell=1}^{r} s_{\ell,q+1} \\ \vdots \\ \prod_{\ell=1}^{r} s_{\ell,N} \end{bmatrix} = \begin{bmatrix} c_{d,1} \\ \vdots \\ c_{d,q-1} \\ 1 \\ c_{d,q+1} \\ \vdots \\ c_{d,N} \end{bmatrix};$$

    $q \leftarrow q + 1$;
  end
end

Collect all $C_i, i = 1, \ldots, d$ in one matrix C

$$C = \begin{bmatrix} C_1 & C_2 & \cdots & C_d \end{bmatrix}, \text{ where } d \leq N.$$

Note that for $d = N$, no partial synchronization manifolds exist. The algorithm will stop and return the result stating no partial synchronization manifolds exist.
Finally, multiply C with the vector $v = [1\ 2\ \cdots\ d]^\top$ to obtain A_R:

$$A_R = Cv = [R_1\ \cdots R_N]^\top.$$

A_R will then be exploited in next step to limit the number of partitions.
We remark that the procedure presented here can be also used for networks which are decomposed due to multiple time delays, see Appendix A for details.

Generate possible partitions

Each partition of a network with N systems is presented by a N-digit code $d_1 \cdots d_i \cdots d_N$, following the convention in [12]. Table 1 graphically explains the generation of all possible partition using this coding system in the original algorithm. System 1 always has code 0. When adding a second system, there will be two possible partitions: one coded by 00 when System 1 and 2 belong to the same cluster, the other one coded by 01 when System 2 belongs to a new cluster. Now we add a third system. Based on every possible partition of System 1-2, we can generate the partitions of System 1-3. Starting with partition 00, we can put the third system into the same cluster of System 1 and 2 or a new cluster, coded by 000 or 001 respectively. Similarly, for partition 01, we have three possibilities: 010 (System 1 and 3 in the same cluster), 011 (System 2 and 3 in the same cluster) or 012 (System 3 in a new cluster). Continuing this procedure, we can generate all possible partitions for networks with N systems. Note that the number of partitions generated in this way is known as Bell number. In combinatorial mathematics, Bell numbers $\{B_0, B_1, \ldots\}$ introduced in [1] count the numbers of partitions of a set. The N-th Bell number B_N is the number of partitions of a set of size N. Starting with $B_0 = 1$, the Bell numbers can be generated by using the following recurrence relation, [13]:

$$B_n = \sum_{k=1}^{n-1} \binom{n-1}{k} B_k, \text{ for } n \geq 1,$$

where $\binom{n-1}{k}$ define the binomial coefficients. It can be shown that the Bell numbers increase in a exponential-like way with N. The first few Bell numbers are $1, 1, 2, 5, 15, 52, 203, 877, 4140, \ldots$

To reduce the number of partitions, we impose the row sum requirements by exploiting the vector A_R during partition generation for networks with coupling (2). System 1 still gets code 0. However, when adding subsequent systems, the possibilities will be restricted by the row sum requirements. A_R is used to determine where to put the newly added systems, see below.

Algorithm 2: Generate possible partitions

Input: $A_R = [R_1 \ \cdots \ R_N]^\top$
Output: set of parition codes for systems $\{1, \ldots, N\}$
let $D_1 = \{0\}$ be the set of partition codes for system $\{1\}$ (d_1=0);
for *i=2,...,N* **do**
  set the set of partition codes for systems $\{1, \ldots, i\}$ empty;
  foreach *element in the set of partition codes for systems* $\{1, \ldots, i-1\}$ **do**
    retrieve $d_1, \ldots, d_{i-1}$;
    compute the set $D_i = \underbrace{\{d_j \mid R_j = R_i,\ j = 1, \ldots, i-1\}}_{\text{System i allowed in the same clsuters of other systems due to same row sums}} \cup \underbrace{\{\max(d_1, d_2, \ldots, d_{i-1}) + 1\}}_{\text{System i in a new clsuter}}$;
    append every element of D_i to the current code $d_1 \cdots d_{i-1}$ and add all corresponding codes $d_1 \cdots d_i$ in the set of partition codes for systems $\{1, \ldots, i\}$;
  end
end

System 1	Systems 1,2	Systems 1,2,3	Systems 1,2,3,4	$\cdots$
0	00	000	0000	$\cdots$
			0001	
		001	0010	
			0011	
			0012	
				$\ddots$
	01	010	0100	
			0101	
			0102	
		011	0110	
			0111	
			0112	
		012	0120	
			0121	
			0122	
			0123	
				$\ddots$

Table 1: Generating the set of all partitions of a number of systems using the original algorithm, [12].

When the procedure reaches $i = N$, we obtain all possible partitions compliant to the row sum requirements. The viabilities of these partitions will then be checked using the row sum criteria from Theorem 1 or 2.
Remarkably, the number of possible partitions for networks with coupling (2) generated by the improved algorithm can be inferred from the structure of A_R.
Suppose A_R has d distinct elements, i.e., the adjacency matrix A has d distinct row sums. Denote k_i as the number of elements with equal values. The number of partitions $N_{\mathcal{P}}$ generated by the new procedure is

$$N_{\mathcal{P}} = \prod_{i=1}^{d} B_{k_i}. \tag{12}$$

Here B_{k_i} is the number of partitions of a set of size k_i, i.e., the k_i-th Bell number.
This can also be explained in a more intuitive way. Let us group the nodes according to the row sums of A and denote this partition as $\bar{\mathcal{P}} = \{\bar{\mathcal{P}}_1, \ldots, \bar{\mathcal{P}}_d\}$. Then $k_i, i = 1, \ldots, d$ is the number of nodes in $\mathcal{P}_i$. For nodes to be in same part in a viable partition, it is *necessary* for them to be in the same group of $\bar{P}$. Therefore, the allowable partitions can be generated by:

- Generate all partitions of sub-networks of $\bar{\mathcal{P}}_i$ and denote the partition set as $\bar{\mathcal{D}}_i$;
- Obtain the partitions of the whole network by combining these partitions of sub-networks, i.e., $\bar{\mathcal{D}}_1 \times \bar{\mathcal{D}}_2 \times \cdots \times \bar{\mathcal{D}}_d$.

Since $|\bar{\mathcal{D}}_i| = B_{k_i}$, the total number of partitions ($|\bar{\mathcal{D}}_1 \times \bar{\mathcal{D}}_2 \times \cdots \times \bar{\mathcal{D}}_d|$) equals to $\prod_{i=1}^{d} B_{k_i}$. Obviously, the follow inequality always holds for $d \geq 2$

$$B_N > \prod_{i=1}^{d} B_{k_i}, \tag{13}$$

where $N = \sum_{i=1}^{d} k_i$ and $N, k_i \in \mathbb{Z}^+$. This supports the conclusion that in many cases the improve algorithm generates fewer partitions than the original algorithm.

Example

The example here shows the computation of all possible partitions of a network with 2 different weights (invasive coupling type) using the improved algorithm. We use the same adjacency matrices from an example in [12]. Below are the decomposed adjacency matrices of the network.

$$A_1 = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}, A_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Calculate row sums of all matrices

$$A_{1,R} = \begin{bmatrix} 2 & 1 & 3 & 3 & 2 & 1 & 3 & 1 \end{bmatrix}^\top,$$
$$A_{2,R} = \begin{bmatrix} 0 & 1 & 1 & 1 & 3 & 0 & 1 & 0 \end{bmatrix}^\top.$$

Combine row sum condition of all matrices

Row 1

$$S_{2,1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}^\top,$$
$$S_{1,1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}^\top,$$
$$C_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^\top.$$

Row 2

$$S_{1,2} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}^\top,$$
$$S_{2,2} = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}^\top,$$
$$C_2 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^\top.$$

Row 3

$$S_{1,3} = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}^\top,$$
$$S_{2,3} = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}^\top,$$
$$C_3 = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}^\top.$$

Row 4 is skipped as $c_{3,4} = 1$ in C_3.

Row 5

$$S_{1,5} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}^\top,$$
$$S_{2,5} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}^\top,$$
$$C_4 = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}^\top.$$

Row 6

$$S_{1,6} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}^\top,$$
$$S_{2,6} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}^\top,$$
$$C_5 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}^\top.$$

Row 7 is skipped as $c_{3,7} = 1$ in C_3.
Row 8 is skipped as $c_{5,8} = 1$ in C_5.
So,

$$C = \begin{bmatrix} C_1 & C_2 & C_3 & C_4 & C_5 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Multiply C with $v = [1\ 2\ 3\ 4\ 5]^\top$, yields

$$A_R = Cv = \begin{bmatrix} 1 & 2 & 3 & 3 & 4 & 5 & 3 & 5 \end{bmatrix}^\top$$

Therefore, we can conclude:

- In both A_1 and A_2, row 3, row 4 and row 7 have the same row sum;
- In both A_2 and A_2, row 6 and row 8 have the same row sum.

Generate possible partitions

System 1	Systems 1,2	Systems 1,2,3	Systems 1,2,3,4	Systems 1,2,3,4,5	Systems 1,2,4,5,6	Systems 1,2,3,4,5,6,7	Systems 1,2,3,4,5,6,7,8
$d_1 \in \{0\}$	$d_2 \in \{1\}$	$d_3 \in \{2\}$	$d_4 \in \{2,3\}$	$d_5 \in \{3\}$	$d_6 \in \{4\}$	$d_7 \in \{2,5\}$	$d_8 \in \{4,5\}$
0	01	012	0122	01223	012234	0122342	01223424
							01223425
							$d_8 \in \{4,6\}$
						0122345	01223454
							01223456
				$d_5 \in \{4\}$	$d_6 \in \{5\}$	$d_7 \in \{2,3,6\}$	$d_8 \in \{5,6\}$
			0123	01234	012345	0123452	01234525
							01234526
							$d_8 \in \{5,6\}$
						0123453	01234535
							01234536
							$d_8 \in \{5,7\}$
						0123456	01234565
							01234567

Table 2: Generating the set of all partitions of 8 systems with row sum restrictions.

Table 2 shows the process of partition generation using the improved algorithm. As can be seen from it, there are only 10 partitions generated, while the original algorithm generates $B_8 = 4140$ (the 8th Bell number) partitions.
Recall that the number of possible partitions can be calculated from the structure of A_R. In this example, we have

$$A_R = \begin{bmatrix} 1 & 2 & 3 & 3 & 4 & 5 & 3 & 5 \end{bmatrix}^\top.$$

Re-arrange the entries by their values, we have

$$\tilde{A}_R = \begin{bmatrix} 1 & 2 & 3 & 3 & 3 & 4 & 5 & 5 \end{bmatrix}^\top,$$

therefore, the total number of possible partitions equals to $B_1 \times B_1 \times B_3 \times B_1 \times B_2 = 10$, where B_i represents the ith Bell number. In fact, the partitions generated by the improved algorithm is a subset of those generated by the original algorithm. Thereby, there are fewer partitions available for performing the row sum test, thus saving computational effort.

Concluding Remarks

In this paper, we have presented an improved algorithm to calculate the partial synchronization manifolds $\mathcal{M}(\Pi)$ in networks of systems (1), in particular, with invasive coupling time-delay coupling (2). Using this algorithm, the computational cost can be largely reduced as the number of partitions generated to test is greatly smaller compared to the original

algorithm proposed in [12]. In fact, the number of the generated partitions is related to the distribution of row sums of the adjacency matrix. Clearly, no partial synchronization manifolds exist when all rows of the adjacency matrix have distinct row sums for the system (1) with invasive coupling (2). In this case, the algorithm stops and returns the corresponding result. Finally, we remark that the algorithm can also be applied to networks with multiple time delays.

Acknowledgments

This research was supported by the project UCoCoS, funded by the European Unions Horizon 2020 research and innovation program under the Marie Skodowska-Curie Grant Agreement No 675080.

Appendix A: Multiple time delays

In this appendix, we discuss the case where networks have multiple time delays. For the case of multiple time-delays, the network can be decomposed analogously to (8). Here, we also adopt the settings from [12]. Suppose the coupling has p distinct delays. The corresponding coupling functions are

$$u_i(t) = \sum_{j \in \mathcal{N}_i} a_{ij}[y_j(t - \tau_{ij}) - y_i(t)] \tag{14}$$

or

$$u_i(t) = \sum_{j \in \mathcal{N}_i} a_{ij}[y_j(t - \tau_{ij}) - y_i(t - \tau)], \tag{15}$$

where $\tau_{ij} \in \{\tau_1, \tau_2, \ldots, \tau_p\}$ with $0 < \tau_1 < \tau_2 < \ldots < \tau_p = \tau$. Define the matrices $A_\ell, \ell = 1, \ldots, r$ as follows

$$\tilde{A}_\ell = (\tilde{a}_{\ell,ij}), \text{ with } \tilde{a}_{\ell,ij} = \begin{cases} a_{ij} & \text{if } \tau_{ij} = \tau_\ell, \\ 0 & \text{otherwise.} \end{cases} \tag{16}$$

Therefore, the network $\mathcal{G}$ can be decomposed as:

$$\mathcal{G}(\mathcal{V}, \mathcal{E}, A) = \tilde{\mathcal{G}}_1(\mathcal{V}, \tilde{\mathcal{E}}_1, \tilde{A}_1) \oplus \cdots \oplus \tilde{\mathcal{G}}_p(\mathcal{V}, \tilde{\mathcal{E}}_r, \tilde{A}_p), \tag{17}$$

where $\tilde{\mathcal{E}}_\ell \subset \mathcal{V} \times \mathcal{V}$ presents the set of edges corresponding to $\tau_\ell, \ell = 1, \ldots, p$.
Then, we have the following condition for existence of partial synchronization manifolds for this type of networks.

Theorem 3 *[12] Consider a graph $\mathcal{G}$ that decomposes as* (17)*. Given a permutation matrix Π of appropriate dimension and associate with that Π a reordering matrix R such that $R^\top \Pi R$ is of the form* (6)*. Assume system* (1) *is left-invertible, then the following statements are equivalent:*

1) *$\mathcal{M}(\Pi)$ is a partial synchronization manifold for (1) and (2), respectively (1) and (3);*
2) *for each $\ell = 1, \ldots, p$, all blocks, respectively all off-diagonal blocks, of each block-structured matrices $R^\top \tilde{A}_\ell R$, partitioned in blocks of size $k_i \times k_j$ according to* (6)*, have constant row-sums.*

By comparing Theorem 2 with 3, it can be concluded that we can use almost the same algorithm to compute the partial synchronization manifolds of networks with multiple time delays. The difference is that when applying the Algorithm 1 for $\tilde{A}_1, ..., \tilde{A}_p$, one must take into account that the entries of these matrices might be not integer valued, hence $A_{1,R}, \ldots, A_{p,R}$ should be constructed with some prescribed tolerance. A further decomposition as (8) is possible if the rational dependency structure of the weights permits. If all $\tilde{A}_l$ containing non-integer element(s) can be decomposed in form of (9) with rationally independent basis weights, the algorithm can still be carried out in exact arithmetic.

References

[1] E. T. Bell. Exponential numbers. *The American Mathematical Monthly*, 41(7):411–419, 1934.

[2] B. Bollobas. *Modern Graph Theory*. Springer-Verlag GmbH, 1998.

[3] J. Buck and E. Buck. Synchronous fireflies. *Scientific American*, 234:74–9, 82–5, May 1976.

[4] C. M. Gray. Synchronous oscillations in neuronal systems: mechanisms and functions. *Journal of computational neuroscience*, 1:11–38, June 1994.

[5] C. Huygens. *Oeuvres complètes de Christiaan Huygens: L'hhorloge à pendule de 1651 à 1666*, volume 17. Swets & Zeitlinger, 1967.

[6] H. Nijmeijer K. Y. Pettersen, J. T. Gravdahl, editor. *Group Coordination and Cooperative Control*. Springer-Verlag GmbH, 2006.

[7] F. L. Lewis, H. Zhang, K. Hengster-Movric, and A. Das. *Introduction to Synchronization in Nature and Physics and Cooperative Control for Multi-Agent Systems on Graphs*, pages 1–21. Springer London, London, 2014.

[8] H. Nijmeijer and A. Rodriguez-Angeles. *Synchronization of Mechanical Systems*. WORLD SCIENTIFIC PUB CO INC, 2003.

[9] T. Oguchi and H. Nijmeijer. A synchronization condition for coupled nonlinear systems with time-delay: a frequency domain approach. *International Journal of Bifurcation and Chaos*, 21(09):2525–2538, sep 2011.

[10] A. Pogromsky, G. Santoboni, and H. Nijmeijer. Partial synchronization: from symmetry towards stability. *Physica D: Nonlinear Phenomena*, 172(1-4):65–87, nov 2002.

[11] K. Ryono and T. Oguchi. Partial synchronization in networks of nonlinear systems with transmission delay couplings. *IFAC-PapersOnLine*, 48(18):77–82, 2015.

[12] E. Steur, H. U. Ünal, C. van Leeuwen, and W. Michiels. Characterization and computation of partial synchronization manifolds for diffusive delay-coupled systems. *SIAM Journal on Applied Dynamical Systems*, 15(4):1874–1915, jan 2016.

[13] H. S. Wilf. *Generatingfunctionology. 2nd ed.* Boston, MA: Academic Press, 2nd ed. edition, 1994.