The Multilayer Artificial Benchmark for Community Detection (mABCD)

Piotr Bródka[1], Michał Czuba[1], Bogumił Kamiński[2], Łukasz Kraiński[2], Paweł Prałat[3], and François Théberge[4]

[1] Dept. of Artificial Intelligence, Wrocław University of Science and Technology, Wrocław, Poland; {piotr.brodka, michal.czuba}@pwr.edu.pl
[2] Decision Analysis and Support Unit, SGH Warsaw School of Economics, Warsaw, Poland; {bkamins, lkrain}@sgh.waw.pl
[3] Dept. of Mathematics, Toronto Metropolitan University, Toronto, Canada; pralat@torontomu.ca
[4] Tutte Institute for Mathematics and Computing, Ottawa, Canada; theberge@ieee.org

Abstract. The **A**rtificial **B**enchmark for **C**ommunity **D**etection (**ABCD**) is a random graph model that incorporates community structure and follows a power-law distribution for both node degrees and community sizes. It produces graphs similar to the well-known LFR model but is faster, more interpretable, and analytically tractable. In this paper, we build on the core principles of **ABCD** to introduce **mABCD**, a variant designed for multilayer networks.

Keywords— ABCD, community detection, multilayer networks, benchmark models

1 Introduction

Community structure is a key feature of real-world networks, revealing their internal organization [14,21]. In social networks, it reflects shared interests; in citation networks, it groups related papers; and in the Web graph, it connects pages on similar topics. Identifying communities helps in understanding network structure.

However, datasets with ground-truth communities are scarce, necessitating synthetic graph models for benchmarking clustering algorithms. The **LFR** model [28,27] is widely used to generate networks with community structures while allowing heterogeneity in node degrees and community sizes.

A more recent alternative, the **ABCD** (**A**rtificial **B**enchmark for **C**ommunity **D**etection) model [20], along with its fast multi-threaded version **ABCDe** [24], provides comparable properties to **LFR** but is faster and more flexible. It enables smooth transitions between disjoint communities and random graphs and is theoretically easier to analyze [19]. Notably, it exhibits self-similar degree distributions [2] and has been extended to handle outliers (**ABCD+o**) [22] and hypergraphs (**h–ABCD**) [23].

The **ABCD** model is gaining traction among both practitioners and researchers. For instance, [1] employs **Adjusted Mutual Information** (**AMI**) to compare 30 community detection algorithms using **LFR** and **ABCD**, highlighting **ABCD**'s scalability and improved parameter control. Given its flexibility, this paper extends **ABCD** to generate multilayer networks.

Multilayer networks [26,13] is a relatively new aspect of complex networks that has gathered a lot of attention over the last decade. Such networks allow us to capture multiple different relations between nodes, thus better reflecting the complexity of real-world interactions. Shortly after the emergence of multilayer networks, the research on community detection in those networks begun [5,36,9] (a detailed description can be found in the survey paper [29]). This, and the fact that real-world datasets with known community structures for multilayer networks are even less common, generated the need for synthetic multilayer networks with known community structures.

The first such models were extensions of the existing models for simple single-layer networks such as **mLFR** [7], which is an extension of the **LFR** model mentioned above. Another model was proposed in [3]. It is designed to generate networks with various kinds of meso-structures, with the community being an example of such a structure. Finally, the authors of the survey paper on community detection in multilayer networks [29] proposed yet another model that allows the generation of a simple community structure and a network with edges based on that structure. This model allowed them to compare several existing community detection algorithms and was partially included in the *multinet* library [30] for analyzing and mining multilayer networks.

While these models provide valuable tools for benchmarking community detection algorithms, each has its trade-offs. Magnani et al.'s [29] approach prioritizes simplicity and algorithm comparison but sacrifices scalability and complexity. Bazzi et al. [3] offer flexibility and mesoscale modelling but face challenges with parameter complexity and computational efficiency. Finally, the **mLFR** model [7] focuses on structural realism but lacks support for heterogeneous and strongly correlated inter-layer properties. These limitations underscore the ongoing need for more advanced models to replicate the multifaceted nature of real-world multilayer networks.

In this paper, we use the underlying ingredients of the **ABCD** model and introduce its variant for multilayer networks, **mABCD**. The paper is structured as follows. First, we introduce the notation and terminology for multilayer networks—see Section 2. In particular, we introduce various correlation measures between the layers. In Section 3, we formally define the **mABCD** model. Conclusions and future directions are presented in Section 4.

2 Multilayer Networks

In this section, we introduce the standard notation used and properties of interest for multilayer networks [13,26]. For a given $n \in \mathbb{N} := \{1, 2, \ldots\}$, we use $[n]$ to denote the set consisting of the first n natural numbers, that is, $[n] := \{1, 2, \ldots, n\}$. We define the multilayer network as a quadruple $M = ([n], [\ell], V = [n] \times [\ell], E)$, where

- $[n]$ is a set of n actors (for example, users of various social networking sites),
- $[\ell]$ is a set of layers (for example, different social networking platforms, such as LinkedIn, Facebook and Instagram, on which actors interact with each other),
- $V = [n] \times [\ell]$ a set of nodes (vertices); node $v = (a, \ell_i) \in V$ represents an actor a in layer ℓ_i,
- E is a set of (undirected) edges between nodes; if $e = v_1 v_2 \in E$ with $v_1 = (a_1, \ell_1) \in V$ and $v_2 = (a_2, \ell_2) \in V$, then $\ell_1 = \ell_2$, that is, edges occur only within layers.

Note that not every actor needs to be present on all layers. For simplicity, in our model, we assume that each layer has exactly n nodes associated with all actors. Actors not engaging with a given layer (we will call them inactive) will be associated with isolated nodes (nodes of degree zero).

2.1 Correlations Between Layers

There are no edges between nodes in different layers, but in most real-world multilayered networks, layers are clearly *not* independently generated. Each actor is associated with ℓ nodes, one in each layer, and there are some highly non-trivial correlations between edges across layers. For example, active users on one social media platform are often also active on another one [16]. This creates correlations between degree distributions across layers. Communities that are naturally formed in various layers often depend on the properties of the associated actors. For example, users interested in soccer might group together on Instagram and on Facebook. As a result, partitions of nodes into communities (associated with different layers) are often correlated. Finally, interactions between actors in one layer might increase their chances of interacting in another layer, yielding correlations at the level of edges.

Below, we briefly summarize how we measure these three types of correlations mentioned above. The first two measures are standard, and their detailed description can be found, for example, in [8,21].

Correlations Between Nodes Degrees in Various Layers We will use **Kendall rank correlation coefficient** τ [25] to measure correlations between sequences of node degrees in two different layers. It is a nonparametric measure of the ordinal association between two measured quantities: the similarity of the orderings of the data when ranked by each of the quantities (in our application, the degree sequences). The Kendall correlation between two variables ranges from -1 to 1. It is large when observations have a similar rank between the two variables and is small when observations have a dissimilar rank between the two variables.

Specifically, we will use the "tau-b" statistic, which is adjusted to handle ties. If an actor is inactive in one of the two layers we compare against each other, then we simply ignore the two nodes corresponding to this actor. As a result, the degree sequences are always of the same length.

Correlations Between Partitions in Various Layers The **adjusted mutual information** (**AMI**), a variation of **mutual information** (**MI**), is a common way to compare partitions of the same set [21,37]. Usually, one may want to compare the partitions returned by some clustering algorithms. In our present context, we may want to compare partitions into ground-truth communities from two different layers. The **AMI** takes a value of 1 when the two partitions are identical and 0 when the **MI** between two partitions equals the value expected due to chance alone. Actors, that are inactive in at least one of the two layers we compare against each other, are ignored so that a comparison of partitions is made on the same set of actors.

Correlations Between Edges in Various Layers To measure correlations between edges in different layers, we define $\mathbf{R}$, a $\ell \times \ell$ matrix in which elements $r_{i,j} \in [0,1]$ $(i,j \in [\ell])$ capture correlation between edges present in layers i and j. For any $i,j \in [\ell]$ with $i < j$, let

$$E_i^j = \{a_1a_2 : (a_1,i)(a_2,i) \in E \wedge a_1,a_2 \in [n] \text{ are active in layers } i \text{ and } j\}, \qquad (1)$$

be the set of edges that are present in layer i, involving actors that are also active in layer j. Note that in the definition of E_i^j, edges are defined over actors that are active

in both layers, not nodes in layer i, so that we can perform set operations on edges between layers. Entries $r_{i,j}$ in $\mathbf{R}$ are computed using the following formula:

$$r_{i,j} = \frac{|E_i^j \cap E_j^i|}{\min\{|E_i^j|, |E_j^i|\}}.$$

If $\min\{|E_i^j|, |E_j^i|\} = 0$, then we leave $r_{i,j}$ undefined; in the implementation, NaN value is produced.

Note that the definition of $\mathbf{R}$ implies that $r_{i,i} = 1$ for any $i \in [\ell]$ and $r_{i,j} = r_{j,i}$ for $1 \leq i < j \leq \ell$. The maximum value of 1 is attained when edges in one of the layers form a subset of edges in the other layer. The minimum value of 0 is attained when the two sets of edges in the corresponding layers are completely disjoint. As a result, r_{ij} aims to capture correlations between individual edges, but it is not normalized as, for example, the Kendall rank correlation coefficient τ. The coefficient τ ranges from -1 to 1, corresponding to the two extremes, and 0 corresponds to a neutral case. Graphs associated with the layers are sparse, but one layer might have substantially more edges than the other. Hence, r_{ij} is convenient, but it does not have a natural interpretation as τ. Finally, let us mention that one can easily update the value of r_{ij} when some small operations are applied to either layer i or j. It will become handy when such operations must be performed on our synthetic model to converge to the desired correlation matrix $\mathbf{R}$.

2.2 Examples of Multilayer Networks

Before constructing the model, we first examined whether and how degrees, edges, and partitions correlate across layers in real-world networks. Existing research [29,8] has yet to provide a definitive answer. To fill this gap, we analyzed eight real-world networks from diverse domains, each differing in actors, nodes, edges, and layers. Table 1 summarizes their structural characteristics, while Figure 1 presents correlation matrices for node degrees, partitions, and edges (as defined in Section 2.1) for one of the analyzed networks.

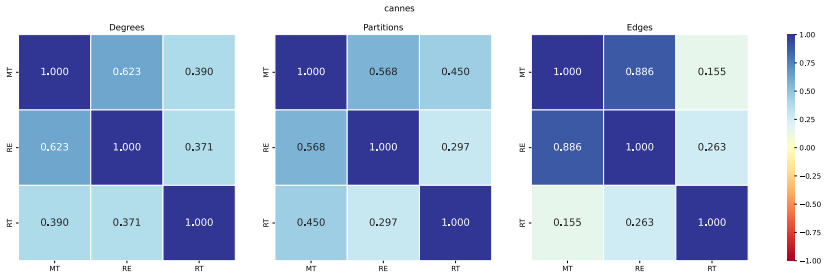

Fig. 1: Node degrees, partitions, and edge correlations between layers of cannes network.

Our analysis revealed no consistent patterns across networks—degrees, edges, and partitions were sometimes correlated and sometimes independent. In one network, two

layers might be correlated, while another layer remained unaligned. To accommodate this variability, we designed **mABCD** for flexibility: advanced users can customize distributions and correlations between layers, while default parameters, such as a power-law degree distribution, provide guidance for less experienced users.

Table 1: Networks used in experiments with their basic parameters shortlisted.

Name	Layers	Actors	Nodes	Edges	Note
arxiv	13	14,065	26,796	59,026	Coauthorship network from articles published on the "arXiv" [12].
aucs	5	61	224	620	A graph of interactions between employees of **A**arhus **U**niversity, Department of **C**omputer **S**cience [34].
cannes	3	438,537	659,951	974,743	A network of interactions between Twitter users [31].
ckmp	3	241	674	1,370	A network depicting diffusion of innovations among physicians [11].
eutr-A	37	417	2,034	3,588	The European air transportation network [10].
l2-course	2	41	82	297	A network of interactions between U.S. students learning Arabic [33].
lazega	3	71	212	1,659	A network of interactions between staff of a law corporation [35].
timik	3	61,702	102,247	881,676	A graph of interactions between users of the virtual world platform [18].

3 The mABCD Model

In this section, we introduce the variant of the **ABCD** model that produces a synthetic collection of graphs that form a multilayer structure, **mABCD**.

3.1 Power-law Distribution

Power-law distributions will be used to generate both the degree sequence and community sizes so let us formally define it. For given parameters $\gamma \in (0, \infty)$, $\delta, \Delta \in \mathbb{N}$ with $\delta \leq \Delta$, we define a truncated power-law distribution $\mathcal{P}(\gamma, \delta, \Delta)$ as follows. For $X \sim \mathcal{P}(\gamma, \delta, \Delta)$ and for $k \in \mathbb{N}$ with $\delta \leq k \leq \Delta$,

$$\mathbb{P}(X = k) = \frac{\int_k^{k+1} x^{-\gamma}\, dx}{\int_\delta^{\Delta+1} x^{-\gamma}\, dx}.$$

3.2 The Configuration Model

The well-known configuration model is an important ingredient of the generation process, so let us formally define it here. Suppose then that our goal is to create a graph

on n nodes with a given degree distribution $\mathbf{d} := (d_i, i \in [n])$, where $\mathbf{d}$ is a sequence of non-negative integers such that $m := \sum_{i\in[n]} d_i$ is even. We define a random multi-graph CM($\mathbf{d}$) with a given degree sequence known as the **configuration model** (sometimes called the **pairing model**), which was first introduced by Bollobás [6]. (See [4,38,39] for related models and results.)

We start by labelling nodes as $[n]$ and, for each $i \in [n]$, endowing node i with d_i half-edges. We then iteratively choose two unpaired half-edges uniformly at random (from the set of pairs of remaining half-edges) and pair them together to form an edge. We iterate until all half-edges have been paired. This process yields $G_n \sim \mathrm{CM}(\mathbf{d})$, where G_n is allowed self-loops and multi-edges and thus G_n is a multi-graph.

3.3 Parameters of the mABCD Model

The **mABCD** model is governed by the following parameters. The first family of parameters is responsible for a few global properties of the model.

Parameter	Range	Description
n	$\mathbb{N}$	Number of actors
ℓ	$\mathbb{N}$	Number of layers
$\mathbf{R}$	$[0,1]^{\ell\times\ell}$	Correlation between edges

Actors will be associated with *labels* from the set $[n]$. These labels will affect the degrees of actors. Each actor $a \in [n]$ will be associated with ℓ nodes, $v_i = (a, i)$ with $i \in [\ell]$, one for each of the ℓ layers. Moreover, each actor will be associated with a vector in $\mathbb{R}^d$ representing their features. We will refer to these vectors as vectors in the *reference layer*. This reference layer will affect the process of generating partitions into communities in various layers.

The second family of parameters is responsible for various properties that are specific for each of the ℓ layers; subscripts $i \in [\ell]$ indicate that the corresponding parameters shape the i^{th} layer. In particular, the set of parameters ξ_i, $i \in [\ell]$, will control the level of noise, that is, the fraction of edges in layer i that are between nodes from two different communities.

Parameter	Range	Description
q_i	$(0,1]$	Fraction of active actors
τ_i	$[-1,1]$	Correlation coefficient between degrees and labels
r_i	$[0,1]$	Correlation between communities and reference layer
γ_i	$(2,3)$	Exponent of power-law degree distribution
δ_i	$\mathbb{N}$	Min degree at least δ_i
Δ_i	$\mathbb{N}$ $(1 \le \delta_i \le \Delta_i < n)$	Max degree at most Δ_i
β_i	$(1,2)$	Exponent of power-law community size distribution
s_i	$\mathbb{N}$	Min community size at least s_i
S_i	$\mathbb{N}$ $(\delta < s_i \le S_i \le n)$	Max community size at most S_i
ξ_i	$(0,1)$	Level of noise

3.4 The mABCD Construction

We will use $\mathcal{A}$ for the distribution of graphs (layers) generated by the following 6-phase construction process. The model generates ℓ graphs; graph $G_n^i = ([n] \times \{i\}, E^i)$,

$i \in [\ell]$, is the graph representing the ith layer. Once they are generated, we simply take $V = \bigcup_{i\in[\ell]}([n] \times \{i\})$ and $E = \bigcup_{i\in[\ell]} E^i$.

Phase 1: Selecting Active Nodes As mentioned above, in a multilayer network, not all of the actors are active in all the layers. Actor $a \in [n]$ is *active* in layer i with probability q_i, independently for each $i \in [\ell]$ and all other actors. If an actor is not active in a given layer, then it will be represented by an "artificial" *inactive* node. We use N^i to denote the number of active nodes (and actors) in layer i. (Clearly, N^i is a random variable with expectation $q_i n$.) For convenience, we will keep inactive nodes as part of the corresponding graphs, but one may think of them as being removed from a given layer. Each actor is active in all layers by default, thus $q_i = 1$ $(i \in [\ell])$.

Phase 2: Creating Degree Sequences The degree sequences for all of the ℓ layers are generated independently so we may concentrate on a given layer $i \in [\ell]$. We ensure that the degree sequence satisfies (a) a power-law with parameter γ_i, (b) a minimum value of at least δ_i, and (c) a maximum value of at most Δ_i.

Inactive nodes (representing actors not present in particular layer) are easy to deal with, they simply have degree zero. The remaining N^i degrees are i.i.d. samples from the distribution $\mathcal{P}(\gamma_i, \delta_i, \Delta_i)$. We use $\mathbf{d}^i_n = (d^i_v, v \in [n])$ for the generated degree sequence of G^i_n with $d^i_1 \geq d^i_2 \geq \cdots \geq d^i_n$; $\mathbf{d}^i_{N^i}$ is a degree subsequence of active nodes. Finally, to ensure that $\sum_{v\in[n]} d^i_v$ is even, we decrease d^i_1 by 1 if necessary; we relabel as needed to ensure that $d^i_1 \geq d^i_2 \geq \cdots \geq d^i_n$.

Parameter $\tau_i \in [-1, 1]$ controls how degrees of the nodes are correlated with labels of the associated actors (recall that node (a, i) in layer i is associated with an actor with label a). In one important case, namely, when $\tau_i = 0$, there is no correlation at all and the degree sequence $\mathbf{d}^i_{N^i}$ is assigned randomly to the N_i active nodes. When $\tau_i = 1$, the order of active nodes with respect to their labels is the same as the order with respect to their degrees; the largest degree node is first. In other words, if nodes $(a_1, i), (a_2, i)$ with $1 \leq a_1 < a_2 \leq n$ are active, then $\deg^i(a_1) \geq \deg^i(a_2)$, where $\deg^i(a_1)$ is the degree of node (a_1, i). Similarly, if $\tau_i = -1$, then the order of active nodes with respect to their labels is also consistent with the order with respect to their degrees but this time the last node is of the largest degree. Since τ_i's could be different for different layers, one node could have large degrees in some layers but small ones in some other ones.

To achieve the desired property, each active node (a, i) independently generates a normally distributed random variable $X_a = N(a/n, \sigma_i)$, where the variance σ_i is a specific function of τ_i. (Recall that we concentrate on a given layer $i \in [\ell]$. For convenience, we simplify the notation and stop referencing to layer i in notation such as X_a. Still, there are many independent random variables for each active node (a, i) associated with actor a.) We sort active nodes in increasing order of their values of X_a and assign the degree sequence accordingly; that is, node (a, i) gets degree d^i_r, where $r \in [N_i]$ is the rank of X_a. In particular, the node with the smallest value of X_a gets assigned the largest degree, namely, d^i_1. Note that if $\sigma_i = 0$, then $X_a = a/n$ (deterministically), and so we recover the perfect correlation between the degrees and the labels ($\tau_i = 1$). On the other hand, if $\sigma_i \to \infty$, then the order of nodes is perfectly random (with uniform distribution), so we recover the other desired extreme ($\tau_i = 0$).

Function $\sigma_i : [0, 1] \to [0, \infty)$ is empirically approximated so that the variance $\sigma_i = \sigma_i(\rho_i)$ yields the Kendall rank correlation close to $\tau_i \in [0, 1]$ between the ordering

generated by the ranks of X_a and the labels a associated with corresponding actors that are active in layer i. Twenty degree distributions are independently generated (with different random seeds), and the one with the correlation coefficient that is the closest to the desired value of τ_i is kept. To deal with negative correlations $\tau_i \in [-1, 0)$, we simply "flip" the order generated for $|\tau_i|$.

Phase 3: Creating Communities Our next goal is to create community structure in each layer of the **mABCD** model. When we construct a community, we assign a number of nodes to said community equal to its size. Initially, the communities form empty graphs. Then, in later phases we handle the construction of edges using the degree sequence established in Phase 2.

Similarly to the process of generating the degree sequences, the distributions of community sizes are generated independently, ensuring that the distribution for a given layer $i \in [\ell]$, satisfy (a) a power-law with parameter β_i, (b) a minimum value of s_i, and (c) a maximum value of S_i. In addition, we also require that the sum of community sizes is exactly n. Specifically, inactive nodes (if there are any) form their own community, namely, C_0^i. Other (regular) communities are generated with sizes determined independently by the distribution $\mathcal{P}(\beta_i, s_i, S_i)$. We generate communities until their collective size is at least n. If the sum of community sizes at this moment is $n + x$ with $x > 0$, then we perform one of two actions: if the last added community has a size at least $x + s_i$, then we reduce its size by x. Otherwise (that is, if its size is $c < x + s_i$), then we delete this community, select $c - x$ old communities and increase their sizes by 1.

Now, given that the sequences of community sizes are already determined (for all layers), it is time to assign nodes to communities. To allow communities to be correlated with each other, we first create a latent *reference* layer that will guide the process of assigning nodes to specific communities across all layers. One may think of this auxiliary layer as properties of actors (such as people's age, education, geographic location, beliefs, etc.) shaping different layers (for example, various social media platforms). This single reference layer will be used for all ℓ layers. In this reference layer, each actor $a \in [n]$ gets assigned a random vector in $\mathbb{R}^d$ (by default, $d = 2$) that is taken independently and uniformly at random from the ball of radius one centred at $\mathbf{0} = (0, 0, \ldots, 0)$.

Let us now concentrate on a given layer $i \in [\ell]$. Recall that the community sizes have already been generated. We write L^i for the (random) number of regular communities in layer i partitioning the set of active nodes in this layer and use $\mathbf{c}^i = (c_j^i, j \in \{0\} \cup [L^i])$ for the corresponding sequence of community sizes. Recall that inactive nodes (representing actors not present in a particular layer) form their own community (namely, C_0^i) so $c_0^i = n - N^i$ is the number of inactive nodes in layer i. Let R be the set of active nodes. We assign nodes to communities, dealing with one community at a time, in a random order. When community C_j^i is formed (for some $j \in [L^i]$), we first select a node from R that is at the largest distance from the center $\mathbf{0}$ (in the reference layer). This node, together with its $c_j^i - 1$ nearest neighbours in R, are put to C_j^i. We remove C_j^i from R and move on to the next community.

The above strategy creates a partition of nodes that is highly correlated with the geometric locations of nodes in the reference layer; nodes that are close to each other in the reference layer are often in the same community—see Figure 2 for an example. To reduce the correlation strength (modelled by the parameter $r_i \in [0, 1]$), we perform the following procedure. Each active node independently leaves its own community with

probability $1 - r_i$, freeing a spot in this community. All the nodes that left are then put back randomly to any available spot (which typically is in a community that this node was *not* originally in). Note that in the extreme case, when $r_i = 0$, the resulting partition does not depend on the reference layer at all, so there is no correlation. We write $\mathbf{C}_n^i = (C_j^i, j \in \{0\} \cup [L^i])$ for the generated collection of communities in G_n^i. Again, let us stress the fact that $\mathbf{C}_n^i$ is a random partition of $[n]$ of random size $L^i + 1$.

Finally, note that in the above process of assigning nodes to communities, as opposed to the original **ABCD** model, we ignore the degree of nodes. Indeed, the original **ABCD** model tries to make sure that large degree nodes are not assigned to small communities. In **mABCD**, there are many layers and non-trivial correlations between partitions into communities and degree sequences between layers. In a hypothetical extreme situation, it might happen that each node belongs to some small community in some layer. Hence, the **mABCD** model does not try to prevent such unavoidable situations and will resolve potential issues later (see Phase 5).

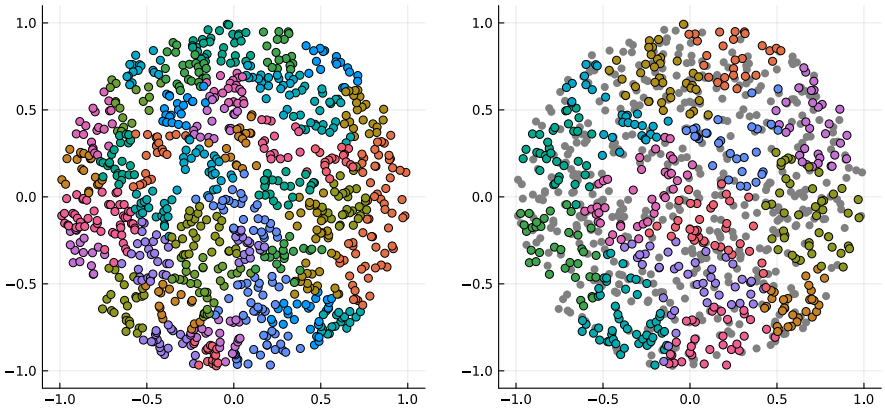

Fig. 2: Two partitions generated based on the same reference layer with $n = 1{,}000$ nodes: (left) $q_1 = 1$ (all nodes active), $S_1 = 32$, $s_1 = 16$, $\beta_1 = 1.5$, (right): $q_2 = 0.5$ (50% nodes active), $S_2 = 50$, $s_2 = 25$, $\beta_2 = 1.5$.

Phase 4: Creating Edges Now, it is time to form edges in **mABCD**. It will be done in the next three phases, Phases 4–6. Phases 4 and 5 will independently generate ℓ graphs G_n^i, $i \in [\ell]$, for each of the ℓ layers whereas Phase 6 will make sure that edges across various layers are correlated, if needed. We may then concentrate on a given layer $i \in [\ell]$.

At this point G_n^i contains n nodes labelled as (a, i), $a \in [n]$, partitioned by the communities $\mathbf{C}_n^i$, with node (a, i) containing $\deg^i(a)$ unpaired half-edges. Firstly, for each $a \in [n]$, we split the $\deg^i(a)$ half-edges of (a, i) into two distinct groups, which we call *community* half-edges and *background* half-edges. For $a \in \mathbb{Z}$ and $b \in [0, 1)$ define

the random variable $\lfloor a+b \rceil$ as

$$\lfloor a+b \rceil = \begin{cases} a & \text{with probability } 1-b, \text{ and} \\ a+1 & \text{with probability } b\,. \end{cases}$$

(Note that $\mathbb{E}\left[\lfloor a+b \rceil\right] = a(1-b)+(a+1)b = a+b$.) Now define $Y_a := \left\lfloor (1-\xi)\deg^i(a) \right\rceil$ and $Z_a := \deg^i(a) - Y_a$ (note that Y_a and Z_a are random variables with $\mathbb{E}\left[Y_a\right] = (1-\xi)\deg^i(a)$ and $\mathbb{E}\left[Z_a\right] = \xi \deg^i(a)$ and since we generate each layer separately they are different for each layer) and, for all $a \in [n]$, split the $\deg^i(a)$ half-edges of (a,i) into Y_a community half-edges and Z_a background half-edges. Next, for all $j \in [L^i]$, construct the *community graph* $G^i_{n,j}$ as per the configuration model on node set C^i_j and degree sequence $(Y_a, a \in C^i_j)$. Note that C_0 consists of inactive nodes which, by design, have degree zero. Hence, there is no need to do anything with them. Finally, construct the *background graph* $G^i_{n,0}$ as per the configuration model on node set $[n]$ and degree sequence $(Z_a, a \in [n])$. In the event that the sum of degrees in a community is odd, we pick a maximum degree node (a,i)) in said community and replace Y_a with Y_a+1 and Z_a with Z_a-1. Note that $G^i_{n,j}$ is a graph, and C^i_j is the set of nodes in this graph; we refer to C^i_j as a *community* and $G^i_{n,j}$ as a *community graph*. Note also that $G^i_n = \bigcup_{0 \le j \le L^i} G^i_{n,j}$.

Phase 5: Rewiring Self-loops and Multi-edges We continue concentrating on a given layer $i \in [\ell]$. Note that, although we are calling $G^i_{n,j}$ ($j \in \{0\} \cup [L^i]$) *graphs*, they are in fact *multi-graphs* at the end of Phase 4. To ensure that G^i_n is simple, we perform a series of *rewirings* in G^i_n. A rewiring takes two edges as input, splits them into four half-edges, and creates two new edges distinct from the input. We first rewire each community graph $G^i_{n,j}$, $j \in [L^i]$, independently as follows.

1. For each edge $e \in E(G^i_{n,j})$ that is either a loop or contributes to a multi-edge, we add e to a *recycle* list that is assigned to $G^i_{n,j}$.
2. We shuffle the *recycle* list and, for each edge e in the list, we choose another edge e' uniformly from $E(G^i_{n,j}) \setminus \{e\}$ (not necessarily in the *recycle* list) and attempt to rewire these two edges. We save the result only if the rewiring does not lead to any further self-loops or multi-edges, otherwise we give up. In either case, we then move to the next edge in the *recycle* list.
3. After we attempt to rewire every edge in the *recycle* list, we check to see if the new *recycle* list is smaller. If yes, we repeat step 2 with the new list. If no, we give up and move all of the "bad" edges from the community graph to the background graph.

We then rewire the background graph $G^i_{n,0}$ in the same way as the community graphs, with the slight variation that we also add edge e to *recycle* if e forms a multi-edge with an edge in a community graph or, as mentioned previously, if e was moved to the background graph as a result of giving up during the rewiring phase of its community graph. At the end of Phase 5, we have a simple graph G^i_n representing the i-th layer of a multilayer network.

Phase 6: Correlations Between Edges in Various Layers During this last phase, we continue performing a series of rewiring (in batches) with the goal of creating a multilayer network with the correlations between edges in various layers (as defined in Subsection 2.1) to be as close to the desired matrix $\mathbf{R}$ (provided as one of the

parameters of the model) as possible. It is important to highlight the fact that during this phase, not only do the degrees of the involved nodes not change, but the community degrees stay the same (as well as the background ones). Hence, in particular, the level of noise stays the same.

We run t independent *batches* of operations (by default, $t = 100$). Before every batch, we re-compute the (empirical) correlation matrix $\hat{\mathbf{R}}$ for the current multilayer network $(G_n^i : i \in [\ell])$ and compare it with the desired matrix $\mathbf{R}$. We select an entry ij at random with the probability proportional to the discrepancy between the empirical and the desired values. In other words, we select a pair (i, j) $(1 \leq i < j \leq \ell)$ with probability

$$p_{ij} = \frac{|\hat{r}_{ij} - r_{ij}|}{\sum_{1 \leq r < s \leq \ell} |\hat{r}_{rs} - r_{rs}|}.$$

We attempt to rewire $\lceil \epsilon \min\{|E_i^j|, |E_j^i|\} \rceil$ of edges in each batch with the goal to bring $\hat{r}_{ij}$ closer to r_{ij} (by default, $\epsilon = 0.05$). Recall that E_i^j can be viewed as the set of edges in layer i that are between actors that are active in layer j (and, trivially, also active in layer i since inactive actors form isolated nodes), see (1).

Suppose first that $\hat{r}_{ij} < r_{ij}$, that is, the correlation between layer i and layer j is smaller than what we wished for. Each of the attempts does the following. Randomly select one of the two graphs, G_n^i or G_n^j, and call it *primary*. Then, pick a random edge uv from the primary graph between actors that are active in both layers. Our goal is to try to introduce edge uv in the other graph (call it *secondary*) unless it is already there, in which case we simply finish this attempt prematurely. If u and v belong to one of the communities in the secondary graph (say to the community C), then we take u' to be a random neighbour of u in C (if there are any), take v' to be a random neighbour of v in C (again, if there are any). If u, v, u', v' are four distinct nodes and there is no edge $u'v'$ in the secondary graph, then we remove the two edges uu' and vv' and introduce two new edges uv and $u'v'$. If anything goes wrong, then we simply finish prematurely and move on to another attempt. If u and v are from two different communities in the secondary graph, then the procedure is exactly the same, but this time, our goal is to select four nodes, each from a different community. Specifically, we try to pick a random neighbour u' of u outside of the communities u or v belong to. Then, we try to pick a random neighbour v' of v outside of the communities u, v, or u' belong to. If the four selected nodes are different and there is no edge $u'v'$ in the secondary graph, we do the rewiring.

Suppose now that $\hat{r}_{ij} > r_{ij}$, that is, the correlation between layer i and layer j is larger than what we wished for. As before, during each attempt, we randomly make one of the two graphs, G_n^i, G_n^j, to be *primary* and the second one to be *secondary*. Then, pick a random edge uv from the intersection of the two graphs. Our goal is to try to remove edge uv from the secondary graph. If u and v belong to one of the communities in the secondary graph (say to the community C), then we take a random edge $u'v'$ from C. If u, v, u', v' are four distinct nodes and there are no edges uu' nor vv' in the secondary graph, then we remove the two edges uv and $u'v'$ and introduce two new edges uu' and vv'. As before, if anything goes wrong, then we simply finish prematurely and move on to another attempt. If u and v are from two different communities in the secondary graph, then we pick a random edge $u'v'$ from the secondary graph with the property that all four nodes belong to different communities. We try to rewire the two edges, making sure that no multi-edges get created.

The goal of the sequence of t batches is to bring the (empirical) correlation matrix $\hat{\mathbf{R}}$ closer to the desired matrix $\mathbf{R}$. Unfortunately, fixing one entry of $\mathbf{R}$ may affect the

other entries. Hence, it is not guaranteed that the best solution is found after exactly t batches. To take this into account, we track the quality of the multilayer networks $(G_n^i : i \in [\ell])$ at the beginning of each bath (via L_2 norm between $\hat{\mathbf{R}}$ and $\mathbf{R}$) and the final network is one of the t networks that performed best.

4 Conclusions and Futue Directions

In this paper, we introduced **mABCD**, an extension of the **ABCD** model designed for multilayer networks. This new benchmark model provides a flexible and scalable framework for generating synthetic multilayer networks with ground-truth communities while maintaining desirable properties such as power-law degree and community size distributions.

A key contribution of **mABCD** is its ability to control correlations between layers, including node degrees, community structures, and edge connections. Unlike some of the existing multilayer benchmarks, which often impose rigid structural assumptions, **mABCD** allows for a smooth transition between independent layers and highly correlated structures. This makes it particularly suitable for evaluating community detection algorithms across different network configurations.

Our empirical analysis of real-world multilayer networks highlighted the high variability in inter-layer correlations, underscoring the need for a flexible synthetic model. **mABCD** directly addresses this need by allowing users to either fully customize correlation structures or rely on default settings that mimic observed patterns.

With its scalability, adaptability, and theoretical tractability, **mABCD** is a powerful tool for researchers and practitioners working on community detection in multilayer networks. We believe that this benchmark will facilitate more robust evaluations and drive further advancements in the field.

The future work which we aim to present in the follow-up journal paper will include a deeper theoretical analysis of its properties. Initial experiments confirm that **mABCD** effectively captures the desired correlations between degrees, community structures, and edges across layers. However, a more rigorous study of these relationships, including their behaviour and sensitivity to parameter variations, would enhance our understanding of the model's capabilities and limitations. Another area that needs deeper evaluation is correlation control between layers. Our current approach provides flexibility in tuning inter-layer dependencies, but we would like to empirically test if correlations between degrees, partitions, and edges always match the desired values and, if not, what are the tradeoffs between exact matching and model efficiency (generation speed). The computational efficiency of **mABCD** is the next property we would like to evaluate to see if the model could be optimized. For example, optimizing the correlation adjustment phase (Phase 6) could significantly reduce execution time for large networks. Finally, we would like to explore **mABCD** in other applications than community detection. For example, using **mABCD** to generate networks with different topologies (e.g., in terms of correlations between layers or weak/strong community structure) to evaluate how network topology can affect various dynamic processes on those networks. For example, assessing the impact of community structure on information diffusion, epidemic spreading, or opinion dynamics could provide valuable insights into real-world multilayer networks.

Acknowledgement. This research was partially supported by the National Science Centre, Poland, under Grant no. 2022/45/B/ST6/04145 (`https://multispread.pwr.`

edu.pl/), the Polish Ministry of Education and Science within the programme "International Projects Co-Funded", and the EU under the Horizon Europe, grant no. 101086321 OMINO. Views and opinions expressed are, however, those of the authors only and do not necessarily reflect those of the National Science Centre, Polish Ministry of Education and Science, EU or the European Research Executive Agency.

The authors have no competing interests to declare that are relevant to the content of this article.

Code and Data The algorithm is implemented in Julia programming language. Source code and installation instructions are available on mABCD GitHub repository[5].

References

1. Samin Aref, Hriday Chheda, and Mahdi Mostajabdaveh. The Bayan algorithm: Detecting communities in networks through exact and approximate optimization of modularity. *arXiv preprint arXiv:2209.04562*, 2022.
2. Jordan Barrett, Bogumił Kamiński, Paweł Prałat, and François Théberge. Self-similarity of communities of the ABCD model. *Theoretical Computer Science*, 1026:115012, 2025.
3. Marya Bazzi, Lucas GS Jeub, Alex Arenas, Sam D Howison, and Mason A Porter. A framework for the construction of generative models for mesoscale structure in multilayer networks. *Physical Review Research*, 2(2):023100, 2020.
4. Edward A Bender and E Rodney Canfield. The asymptotic number of labeled graphs with given degree sequences. *Journal of Combinatorial Theory, Series A*, 24(3):296–307, 1978.
5. Michele Berlingerio, Michele Coscia, and Fosca Giannotti. Finding and characterizing communities in multidimensional networks. In *2011 international conference on advances in social networks analysis and mining*, pages 490–494. IEEE, 2011.
6. Béla Bollobás. A probabilistic proof of an asymptotic formula for the number of labelled regular graphs. *European Journal of Combinatorics*, 1(4):311–316, 1980.
7. Piotr Bródka. A method for group extraction and analysis in multilayer social networks. *arXiv preprint arXiv:1612.02377*, 2016.
8. Piotr Bródka, Anna Chmiel, Matteo Magnani, and Giancarlo Ragozini. Quantifying layer similarity in multiplex networks: a systematic study. *Royal Society open science*, 5(8):171747, 2018.
9. Piotr Bródka, Tomasz Filipowski, and Przemysław Kazienko. An introduction to community detection in multi-layered social network. In *Information Systems, E-learning, and Knowledge Management Research: 4th World Summit on the Knowledge Society, WSKS 2011, Mykonos, Greece, September 21-23, 2011. Revised Selected Papers 4*, pages 185–190. Springer, 2013.
10. Alessio Cardillo, Jesús Gómez-Gardeñes, Massimiliano Zanin, Miguel Romance, David Papo, Francisco del Pozo, and Stefano Boccaletti. Emergence of network features from multiplexity. *Scientific Reports*, 3(1344):1–6, 2 2013. URL: https://www.nature.com/articles/srep01344, doi:10.1038/srep01344.

[5] https://github.com/KrainskiL/MLNABCDGraphGenerator.jl

11. James Coleman, Elihu Katz, and Herbert Menzel. The diffusion of an innovation among physicians. *Sociometry*, 20(4):253–270, 1957. URL: http://www.jstor.org/stable/2785979.
12. Manlio De Domenico, Andrea Lancichinetti, Alex Arenas, and Martin Rosvall. Identifying modular flows on multilayer networks reveals highly overlapping organization in interconnected systems. *Phys. Rev. X*, 5:011027, 3 2015. URL: https://link.aps.org/doi/10.1103/PhysRevX.5.011027, doi:10.1103/PhysRevX.5.011027.
13. Mark E Dickison, Matteo Magnani, and Luca Rossi. *Multilayer social networks*. Cambridge University Press, Cambridge, England, UK, 7 2016. doi:10.1017/CBO9781139941907.
14. Santo Fortunato. Community detection in graphs. *Physics reports*, 486(3-5):75–174, 2010.
15. Jeffrey Gottfried. Americans' social media use. *Pew Research Center*, 31, 2024.
16. Jarosław Jankowski, Radosław Michalski, and Piotr Bródka. A multilayer network dataset of interaction and influence spreading in a virtual world. *Scientific Data*, 4(1):170144, 2017. doi:10.1038/sdata.2017.144.
17. Bogumił Kamiński, Bartosz Pankratz, Paweł Prałat, and François Théberge. Modularity of the ABCD random graph model with community structure. *Journal of Complex Networks*, 10(6):cnac050, 2022.
18. Bogumił Kamiński, Paweł Prałat, and François Théberge. Artificial benchmark for community detection (ABCD) - Fast random graph model with community structure. *Network Science*, pages 1–26, 2021.
19. Bogumił Kamiński, Paweł Prałat, and François Théberge. *Mining Complex Networks*. Chapman and Hall/CRC, 2021. doi:10.1201/9781003218869.
20. Bogumił Kamiński, Paweł Prałat, and François Théberge. Artificial benchmark for community detection with outliers (ABCD+o). *Applied Network Science*, 8(1):25, 2023.
21. Bogumił Kamiński, Paweł Prałat, and François Théberge. Hypergraph artificial benchmark for community detection (h–ABCD). *Journal of Complex Networks*, 11(4):cnad028, 2023.
22. Bogumił Kamiński, Tomasz Olczak, Bartosz Pankratz, Paweł Prałat, and François Théberge. Properties and performance of the ABCDe random graph model with community structure. *Big Data Research*, 30:100348, 2022.
23. Maurice G Kendall. A new measure of rank correlation. *Biometrika*, 30(1-2):81–93, 1938.
24. Mikko Kivelä, Alex Arenas, Marc Barthelemy, James P Gleeson, Yamir Moreno, and Mason A Porter. Multilayer networks. *Journal of Complex Networks*, 2(3):203–271, 07 2014. doi:10.1093/comnet/cnu016.
25. Andrea Lancichinetti and Santo Fortunato. Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Physical Review E*, 80(1):016118, 2009.
26. Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. Benchmark graphs for testing community detection algorithms. *Physical review E*, 78(4):046110, 2008.
27. Matteo Magnani, Obaida Hanteer, Roberto Interdonato, Luca Rossi, and Andrea Tagarelli. Community detection in multiplex networks. *ACM Computing Surveys (CSUR)*, 54(3):1–35, 2021.
28. Matteo Magnani, Luca Rossi, and Davide Vega. Analysis of multiplex social networks with r. *Journal of Statistical Software*, 98:1–30, 2021.

29. Elisa Omodei, Manlio De Domenico, and Alex Arenas. Characterizing interactions in online social networks during exceptional events. *Frontiers in Physics*, 3:59, 2015.
30. Michał B Paradowski, Nicole Whitby, Michał Czuba, and Piotr Bródka. Peer interaction dynamics and second language learning trajectories during study abroad: A longitudinal investigation using dynamic computational social network analysis. *Language Learning*, 2024.
31. Luca Rossi and Matteo Magnani. Towards effective visual analytics on multiplex and multilayer networks. *Chaos, Solitons & Fractals*, 72:68–76, 2015. Multiplex Networks: Structure, Dynamics and Applications. URL: https://www.sciencedirect.com/science/article/pii/S0960077914002422, doi:10.1016/j.chaos.2014.12.022.
32. Tom A. B. Snijders, Philippa E. Pattison, Garry L. Robins, and Mark S. Handcock. New specifications for exponential random graph models. *Sociological Methodology*, 36(1):99–153, 2006. doi:10.1111/j.1467-9531.2006.00176.x.
33. Lei Tang, Xufei Wang, and Huan Liu. Community detection via heterogeneous interaction analysis. *Data mining and knowledge discovery*, 25:1–33, 2012.
34. Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: is a correction for chance necessary? In *Proceedings of the 26th annual international conference on machine learning*, pages 1073–1080, 2009.
35. Nicholas C Wormald. Generating random regular graphs. *Journal of algorithms*, 5(2):247–280, 1984.
36. Nicholas C Wormald et al. Models of random regular graphs. *London Mathematical Society Lecture Note Series*, pages 239–298, 1999.