The Artificial Benchmark for Community Detection with Outliers and Overlapping Communities (ABCD+o^2)

Jordan Barrett[1], Ryan DeWolfe[1], Bogumił Kamiński[2], Paweł Prałat[1], Aaron Smith[3], and François Théberge[4]

[1] Dept of Mathematics, Toronto Metropolitan University, Toronto, Canada; {jordan.barrett, ryan.dewolfe, pralat}@torontomu.ca
[2] Decision Analysis and Support Unit, SGH Warsaw School of Economics, Warsaw, Poland; bkamins@sgh.waw.pl
[3] Dept of Mathematics and Statistics, University of Ottawa, Ottawa, Canada; asmi28@uOttawa.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**) graph is a random graph model with community structure and power-law distribution for both degrees and community sizes. The model generates graphs similar to the well-known **LFR** model but is faster and more interpretable. In this paper, we use the underlying ingredients of the **ABCD** model, and its generalization to include outliers (**ABCD+o**), and introduce another variant for overlapping communities, **ABCD+o^2**.

Keywords— ABCD, community detection, overlapping communities, benchmark models

1 Introduction

One of the most important features of real-world networks is their community structure, as it reveals the internal organization of nodes. In social networks, communities may represent groups by interest; in citation networks, they correspond to related papers; in the Web graph, communities are formed by pages on related topics, etc. Identifying communities in a network is therefore valuable as it helps us understand the structure of the network.

Detecting communities is quite a challenging task. In fact, there is no definition of community that researchers and practitioners agree on. Still, it is widely accepted that a community should induce a graph that is denser than the global density of the network [10]. Numerous community detection algorithms have been developed over the years, using various techniques such as optimizing modularity, removing high-betweenness edges, detecting dense subgraphs, and statistical inference. We direct the interested reader to the survey [8] or one of the numerous books on network science such as [15].

Most community detection algorithms aim to find a partition of the set of nodes, that is, a collection of pairwise disjoint communities with the property that each node

belongs to exactly one of them. This is a natural assumption for many scenarios. For example, most of the employees on LinkedIn work for a single employer. On the other hand, users of Instagram can belong to many social groups associated with their workplace, friends, sports, etc. Researchers might be part of many research groups. A large fraction of proteins belong to several protein complexes simultaneously. As a result, many real-world networks are better modelled as a collection of overlapping communities [22].

In the context of overlapping communities, one can distinguish two forms of overlap. In *crisp* overlap, nodes belong to communities with equal strength, whereas in *fuzzy* overlap, each node may belong to more than one community, but the strength of its membership to each community may vary. Most existing algorithms for detecting overlapping communities are crisp [11]. However, one may start with one of the crisp algorithms and then modify their outcomes to produce fuzzy overlap. Association scores, like the ones we recently proposed in [2], may be used to measure how strongly a node belongs to a community.

Unfortunately, there are very few datasets with ground-truth communities properly identified and labelled. As a result, there is a need for synthetic random graph models with community structure that resemble real-world networks to benchmark and tune clustering algorithms that are unsupervised by nature. The highly popular **LFR** (**L**ancichinetti, **F**ortunato, **R**adicchi) model [20,19] generates networks with communities and, at the same time, allows for heterogeneity in the distributions of both node degrees and of community sizes. It became a standard and extensively used method for generating artificial networks.

A similar synthetic network to **LFR**, the **A**rtificial **B**enchmark for **C**ommunity **D**etection (**ABCD**) [14] was recently introduced and implemented[5], along with a faster and multithreaded implementation[6] (**ABCDe**) [12]. Undirected variants of **LFR** and **ABCD** produce graphs with comparable properties, but **ABCD** (and especially **ABCDe**) is faster than **LFR** and can be easily tuned to allow the user to make a smooth transition between the two extremes: pure (disjoint) communities and random graphs with no community structure. Moreover, **ABCD** is easier to analyze theoretically—for example, in [13] various theoretical asymptotic properties of the are investigated, including the modularity function that, despite some known issues such as the "resolution limit" reported in [9], is an important graph property of networks in the context of community detection. In [3], some interesting and desired self-similar behaviour of the **ABCD** model is analyzed; namely, that the degree distribution of ground-truth communities is asymptotically the same as the degree distribution of the whole graph (appropriately normalized based on their sizes). Finally, the building blocks in the model are flexible and may be adjusted to satisfy different needs. Indeed, the original **ABCD** model was recently adjusted to include potential outliers (**ABCD+o**) [16] and extended to hypergraphs (**h–ABCD**) [17][7]. For these reasons **ABCD** is gaining recognition as a benchmark for community detection algorithms. For example, [1] used the **Adjusted Mutual Information** (**AMI**) between the partitions returned by various algorithms and the ground-truth partitions of **ABCD** and **LFR** graphs to compare 30 community detection algorithms, and mention that *while being directly comparable to **LFR**, **ABCD** offers additional benefits, including higher scalability and better control for adjusting an analogous mixing parameter.*

[5] `https://github.com/bkamins/ABCDGraphGenerator.jl/`
[6] `https://github.com/tolcz/ABCDeGraphGenerator.jl/`
[7] `https://github.com/bkamins/ABCDHypergraphGenerator.jl`

In this paper we extend the **ABCD+o** model further to allow for overlapping communities (**ABCD+o**^{2}). The **LFR** model has been extended in a similar way [19], and in this model the nodes are assigned to communities based on the construction of a random bipartite graph between nodes and communities that results in (a) a small amount of overlap between almost every pair of communities, and (b) rarely any pair of communities with a large overlap. In **ABCD+o**^{2}, we instead generate overlapping communities based on a hidden, low-dimensional geometric layer which tends to yield fewer and larger overlaps. Furthermore, the ancillary benefits of the ABCD model (an intuitive noise parameter, a fast implementation, and theoretical analysis) are still present, so this extension makes **ABCD+o**^{2}an attractive option for benchmarking community detection algorithms.

The rest of the paper is organized as follows. In Section 2 we present the **ABCD+o**^{2} model, with a full description of generating a graph in Section 2.5. Next, in Section 3 we show the properties of the model and test theoretical expectations versus simulated results. Then, in Section 4 we use the model to benchmark various community detection algorithms and compare their quality under different levels of noise and overlap. Finally, some concluding remarks are given in Section 5.

2 ABCD+o^2 — ABCD with Overlapping Communities and Outliers

As mentioned in the introduction, the original **ABCD** model was extended to include outliers resulting in the **ABCD+o** model. For our current needs, we extend **ABCD+o** further to allow for non-outlier nodes to belong to multiple communities, resulting in the **ABCD+o**^{2} model, **ABCD** with **o**verlapping communities and **o**utliers.

2.1 Notation

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, $[n] := \{1, 2, \ldots, n\}$.

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 $\delta \le \Delta$, 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 $\delta \le k \le \Delta$,

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

2.2 The Configuration Model

The well-known configuration model is an important ingredient of all variants of the **ABCD** models, so let us formally define it here. Suppose 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 [5]. (See [4,24,25] 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 a graph $G_n \sim \mathrm{CM}(\mathbf{d})$ on n nodes, where G_n is allowed self-loops and multi-edges and thus G_n is a multi-graph.

2.3 Parameters of the ABCD+o^2 Model

The following ten parameters govern the **ABCD+o^2** model.

Parameter	Range	Description
n	$\mathbb{N}$	Number of nodes
s_0	$\mathbb{N}$	Number of outliers
η	$[1, \infty)$	Average number of communities a non-outlier node is part of
γ	$(2, 3)$	Power-law degree distribution with exponent γ
δ	$\mathbb{N}$	Min degree as least δ
Δ	$\mathbb{N} \setminus [\delta - 1]$	Max degree at most Δ
β	$(1, 2)$	Power-law community size distribution with exponent β
s	$\mathbb{N} \setminus [\delta]$	Min community size at least s
S	$\mathbb{N} \setminus [s - 1]$	Max community size at most S
ξ	$[0, 1]$	Level of noise

Note that the ranges for γ and β can be relaxed and, more generally, any valid sequences for degrees and community sizes can be given as input to the model. However, the ranges $\gamma \in (2, 3)$ and $\beta \in (1, 2)$ were used in all previous theoretical work on the **ABCD** and **ABCD+o** models, and are suggested parameters based on the behaviour of real-world networks (see [14] for more details).

2.4 Big Picture

The **ABCD+o^2** model generates a random graph on n nodes with degree sequence $(d_i, i \in [n])$ and community size sequence $(s_i, i \in [L])$ following power laws with exponents γ and, respectively, β.

There are s_0 outliers and $\hat{n} = n - s_0$ non-outliers. Outliers will form their own auxiliary "community" C_0. Non-outliers will span a family of L communities $(C_j, j \in [L])$ with each non-outlier belonging to at least one of the communities. These communities will overlap (unless $\eta = 1$) so that non-outliers will belong to η communities, on average. The non-outliers, with their respective degrees, populate $(C_j, j \in [L])$ randomly with the caveat that high degree nodes cannot enter small communities.

Parameter $\xi \in [0, 1]$ dictates the amount of noise in the network. Each non-outlier node i has its degree d_i split into two parts: *community degree* y_i and *background degree* z_i ($d_i = y_i + z_i$). The goal is to get $y_i \approx (1 - \xi)d_i$ and $z_i \approx \xi d_i$. However, y_i and z_i must be non-negative integers, and y_i must be split into η_i non-negative integers, one for each community. Moreover, the sum of degrees assigned to each community must be even. We achieve the first requirement by using an appropriate random rounding of $(1 - \xi)d_i/\eta_i$, and achieve the second requirement by making a few ± 1 adjustments at the end. Note that the neighbours of outliers are sampled from the entire graph, ignoring the underlying community structure, meaning $y_i = 0$ and $z_i = d_i$ if i is an outlier.

Once nodes are assigned to communities and their degrees are split, the edges of each community are then independently generated by the configuration model on the corresponding community degree sequences. After that, the background graph is generated by the configuration model on the degree sequence $(z_i, i \in [n])$. The final **ABCD+o²** model, after an additional clean-up phase handling possible self-loops and duplicate edges, is the union of the community graphs and the background graph.

2.5 The ABCD+o² Construction

The following 6-phase construction process generates the **ABCD+o²** synthetic networks.

Phase 1: creating the degree distribution This phase is the same as in the original **ABCD** model and its generalization, **ABCD+o**. The degree distribution of **ABCD+o²** can be injected into the model as an input. However, by default, it is a distribution that satisfies (a) a power-law with parameter γ, (b) a minimum value of at least δ, and (c) a maximum value of at most Δ.

To achieve the desired degree sequence, degrees are sampled i.i.d. from the distribution $\mathcal{P}(\gamma, \delta, \Delta)$. Let $\mathbf{d}_n = (d_i, i \in [n])$ be the generated degree sequence of G_n with $d_1 \geq \cdots \geq d_n$. Finally, to ensure that $\sum_{i\in[n]} d_i$ is even, we decrease d_1 by 1 if necessary; we relabel as needed to ensure that $d_1 \geq d_2 \geq \cdots \geq d_n$.

Phase 2: assigning nodes as outliers This phase is also the same as in the **ABCD+o** model. As mentioned in the big picture summary, the neighbours of outliers will be sampled from the entire graph, ignoring the underlying community structure. It feels that this part is straightforward, but one potential problem might occur when ξ is close to zero. In the extreme case when $\xi = 0$, only outliers have a non-zero degree in the background graph. In order to make sure that there exists a simple graph that satisfies the required degree distribution, in such extreme situations all outliers must have degrees smaller than s_0.

To prepare for potential problems, the following procedure is proposed in the **ABCD+o** model, which we also keep here. We have that $\ell = \sum_{i\in[n]} \min(1, \xi d_i)$ is a lower bound for the expected number of nodes that will have a non-zero degree in the background graph. Moreover, since outliers have all neighbours in the background graph, there must be at least s_0 nodes of positive degree in the background graph. Assuming that outliers are selected uniformly at random, we expect $\ell + (n - \ell)(s_0/n)$ nodes of positive degree in the background graph. (In fact, since there is a slight bias toward selecting small degree nodes for outliers, we expect slightly more nodes of positive degree in the background graph, which is good.) We introduce the following constraint: a node i of degree d_i can become an outlier if

$$d_i \leq \ell + s_0 - \ell s_0/n - 1. \tag{2}$$

Finally, s_0 nodes satisfying (2) are selected uniformly at random to become outliers.

Phase 3: creating overlapping communities By the end of Phase 2, we have a degree sequence $(d_i, i \in [n])$ and an assignment of outliers and non-outliers. We next assign communities to the **ABCD+o²** model. It is important to keep in mind

that overlapping communities are created in this phase but we do not assign specific nodes to these communities just yet. This assignment process will be handled in the next phase, Phase 4. To make sure there is no confusion, in this phase we will be referring to overlapping sets of *elements (not nodes!)* and in the next phase we will match non-outlier nodes with elements of these sets.

The communities we create here will overlap, provided that $\eta > 1$. There are $\hat{n} = n - s_0$ elements that will eventually be matched with non-outliers and, at the end of this phase, we would like them to belong to $\eta \geq 1$ communities, on average. To achieve this goal and to be compatible with the original **ABCD** model, each non-outlier will belong to a single **primary** community and possibly some **secondary** communities. In particular, this ensures that primary memberships form a partition of non-outlier nodes. We will first generate this partition and then grow each part by a factor of η so that the collective size of all communities is equal to $\eta\hat{n} = \eta(n - s_0)$ in expectation.

Similar to the degree distribution, the distribution of community sizes $(s_j, j \in [L])$ will satisfy (a) a power-law with parameter β, (b) a minimum value of s, and (c) a maximum value of S. Hence, the distribution of primary communities $(\hat{s}_j, j \in [L])$ needs to satisfy power-law with the same parameter β but with a minimum value of $\hat{s} = \lceil s/\eta \rceil$ and a maximum value of $\hat{S} = \lfloor S/\eta \rfloor$. In addition, we require $\sum_{j\in[L]} \hat{s}_j = \hat{n}$. To satisfy both requirements, communities are generated with sizes determined independently by the distribution $\mathcal{P}\left(\beta, \hat{s}, \hat{S}\right)$ until their collective size is at least $\hat{n}$. If, at this point, the sum is $\hat{n} + a$ with $a > 0$ then we perform one of two actions: if the last added community has size at least $a + s$, then we reduce its size by a. Otherwise (that is, if its size is $c < a + s$), then we delete this community, select $c - a$ old communities and increase each of their sizes by 1.

Let L be the random variable counting the number of communities (ignoring the auxiliary "community" C_0 consisting of outliers). Each primary community of size $\hat{s}_j$ will grow to size $s_j = \lfloor \eta \hat{s}_j \rceil$. 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 $\mathbb{E}\left[\lfloor a + b \rceil\right] = a(1 - b) + (a + 1)b = a + b$.) As a result,

$$\mathbb{E}\left[\sum_{j\in[L]} s_j\right] = \sum_{j\in[L]} \mathbb{E}\left[s_j\right] = \eta \sum_{j\in[L]} \hat{s}_j = \eta\hat{n} = \eta(n - s_0),$$

as desired.

For communities to overlap in a natural way, we first create a hidden **reference** layer that will guide the process of assigning elements to specific, overlapping communities. One may think of this auxiliary layer as various latent properties of objects associated with nodes (such as people's age, education, geographic location, beliefs, etc.) shaping communities (such as communities in social media). In this reference layer, each of the $\hat{n}$ elements is assigned a random vector in $\mathbb{R}^2$ that is taken independently and uniformly at random from the ball of radius 1 centred at $\mathbf{0} = (0, 0)$.

Recall that the sequence $(\hat{s}_j, j \in L)$ of primary community sizes is already generated. Let R be the set of $\hat{n}$ elements. We assign these elements to communities, dealing with one primary community at a time, in a random order. When a primary community $\hat{C}_j$ is about to be formed, we first select an element from R that is at the furthest distance from the center $\mathbf{0}$ (in the reference layer). This element, together with its

$\hat{s}_j - 1$ nearest neighbours in R, are put to $\hat{C}_j$. We remove $\hat{C}_j$ from R and move on to generating the next primary community. Once all elements are assigned to primary communities, we get a partition; each element belongs to a single community which we call its **primary** community.

Now it is time to grow each primary community of size $\hat{s}_j$ so that its final size is s_j. We can grow communities in any order, as the order will not matter. As before, let R be the set of all $\hat{n}$ elements (in the reference layer). For a given primary community $\hat{C}_j$ of size $\hat{s}_j$, we first compute the center of mass of elements assigned to this primary community, $\mathbf{x}_j \in \mathbb{R}^2$. Then, we investigate elements in R in the order of increasing distances from $\mathbf{x}_j$. If some element $v \in R$ is not already a primary member of this community, we assign this community to v as its **secondary** community. We stop the procedure once the number of members of this community (both primary and secondary) is exactly s_j. We will use $C_j \supseteq \hat{C}_j$ to denote this community, $|C_j| = s_j \geq \hat{s}_j = |\hat{C}_j|$.

Note that each element belongs to exactly one primary community but can be part of many (or none) secondary communities. In Figure 1 we show an example of the reference layer on $\hat{n} = 150$ elements and three communities. Each of the three primary communities in this example consists of 50 elements before growing by a factor of $\eta = 1.5$, attracting an additional 25 elements as its secondary members.

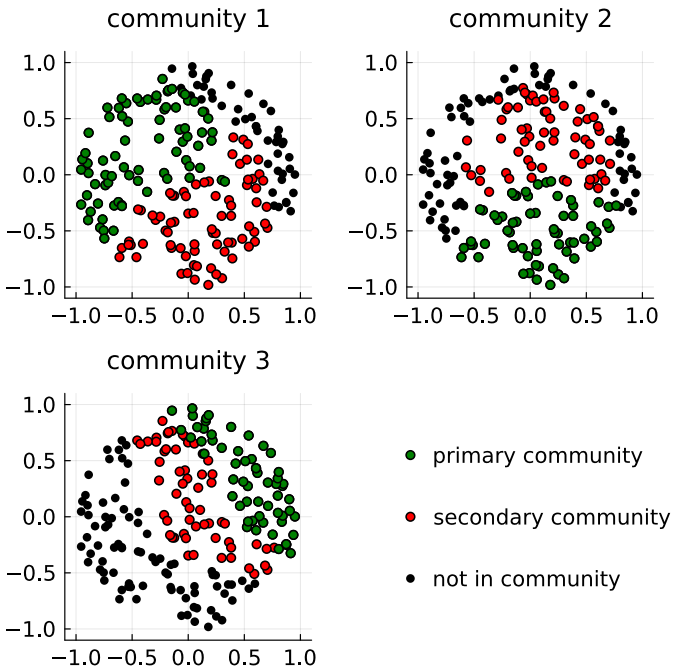

Fig. 1. Example of the reference layer on $\hat{n} = 150$ elements consisting of 3 overlapping communities with equal sizes and $\eta = 1.5$.

Phase 4: assigning degrees to nodes At this point in the construction of **ABCD+o**^{2} we have a degree sequence $(d_i, i \in [n])$, an assignment of outliers to degrees in the sequence, and a collection of overlapping communities containing "elements", each element having a primary community and some number of secondary communities. Let $\hat{\mathbf{d}}_{\hat{n}}$ be the subsequence (of length $\hat{n}$) of $(d_i, i \in [n])$ corresponding to the non-outliers. We are now ready to assign degrees in $\hat{\mathbf{d}}_{\hat{n}}$ to the community elements. Each element $j \in [\hat{n}]$, that will eventually get node i of degree d_i assigned, is expected to have ξd_i neighbours in the background graph and the remaining $(1-\xi)d_i$ neighbours split evenly between $\eta_j \geq 1$ communities. Note that, although each element has a distinct primary community, its degree will be split with no preference given to said primary community.

Similarly to the potential problem with outliers, we need to make sure that non-outliers of a large degree do not join small communities. Although, for a node $j \in [\hat{n}]$, we know the expected fraction of j's neighbours belonging to community C_k, $k \in [L]$, this is in fact a lower bound as some neighbours of j from the background graph might be in C_k by chance. To make enough room in the community graph, a small correction (typically negligible in practice) is introduced in both **ABCD** and **ABCD+o** that is guided by the parameter ϕ (typically ϕ is very close to 1). For consistency, we keep it in the **ABCD+o**^{2} model as well.

We iteratively assign degrees to elements as follows. Recall that the degree sequence $\hat{\mathbf{d}}_{\hat{n}}$ is sorted with $\hat{d}_1$ being the maximum degree. Starting with $i = 1$, let U_i be the collection of unassigned elements at step i. At step i, choose an element j uniformly at random from the set of elements in U_i that satisfy

$$\hat{d}_i \leq \frac{\eta_j}{1-\xi\phi} \cdot \min\left\{|C_k| - 1 : j \in C_k\right\}, \tag{3}$$

where η_j is the number of communities element j belongs to and

$$\phi = 1 - \sum_{k\in[L]} \left(\frac{\hat{s}_k}{\hat{n}}\right)^2 \frac{\hat{n}\xi}{\hat{n}\xi + s_0},$$

and assign this element j to the ith node in the subsequence $\hat{\mathbf{d}}_{\hat{n}}$ that is of degree $\hat{d}_i$; we have that $U_{i+1} = U_i \setminus \{j\}$.

Recall that, eventually, degree d_i of node i belonging to η_i communities will be split into the background degree (approximately ξd_i) and the community degree that will be further split into η_i parts (approximately $(1-\xi)d_i/\eta_i$ each). This explains the condition (3): $(1-\xi)d_i/\eta_i$ has to be smaller than the smallest community i is part of. Indeed, we bound the degrees assignable to element j in the community C_k to ensure that there are enough elements in $C_k \setminus \{j\}$ for j to pair with, preventing guaranteed self-loops or guaranteed multi-edges during the next phase of the construction. Element j could possibly belong to multiple communities, but the bottleneck is clearly with the smallest one that is of size $\min\{|C_k| : j \in C_k\}$. This strategy guarantees that the assignment is selected uniformly at random from the set of all admissible assignments. The details are quite involved and not overly important for our present discussion. Thus, we point the reader to [16,13,14] for a full explanation of the assignment process.

Phase 5: creating edges At this point there are n nodes with labels from $[n]$; $\hat{n} = n - s_0$ of them are non-outliers and the remaining ones are outliers. There is also a family of overlapping communities with each non-outlier node $i \in [n]$ belonging to $\eta_i \geq 1$ communities. Finally, each node $i \in [n]$ (either outlier or non-outlier) is assigned

a degree d_i which we interpret as a set of d_i unpaired half-edges. The last step is to form the edges.

For each non-outlier $i \in [n]$ we split its d_i half-edges into *community* half-edges and *background* half-edges. To this end, define $Y_i := \lfloor (1-\xi)d_i \rceil$ and $Z_i := d_i - Y_i$ (note that Y_i and Z_i are random variables with $\mathbb{E}[Y_i] = (1-\xi)d_i$ and $\mathbb{E}[Z_i] = \xi d_i$) and, for all non-outliers $i \in [n]$, split the d_i half-edges of i into Y_i community half-edges and Z_i background half-edges. Community half-edges are further split into η_i communities non-outlier node i belongs to, as evenly as possible. Specifically, for the communities containing node i, $Y_i - \eta_i \lfloor Y_i/\eta_i \rfloor$ communities (chosen randomly) each receive $\lfloor Y_i/\eta_i \rfloor + 1$ half-edges and the remaining communities each receive $\lfloor Y_i/\eta_i \rfloor$ half-edges. On the other hand, if $i \in [n]$ is an outlier then we set $Z_i = d_i$.

Once the assignment of degrees is complete, for each $j \in [L]$, we independently construct the *community graph* $G_{n,j}$ as per the configuration model on node set C_j and the corresponding degree sequence. In the event that the sum of degrees in a community is odd, we pick a maximum degree node i in said community and decrease its community degree by one while increasing its background graph degree by one. Finally, construct the *background graph* $G_{n,0}$ as per the configuration model on node set $[n]$ and degree sequence $(Z_i, i \in [n])$. Let $G_n = \bigcup_{0 \le j \le n} G_{n,j}$ be the union of all graphs generated in this phase.

Phase 6: rewiring self-loops and multi-edges Note that, although we are calling $G_{n,0}, G_{n,1}, \ldots, G_{n,L}$ *graphs*, they are in fact *multi-graphs* at the end of phase 5. To ensure that G_n is simple, we perform a series of *rewirings* in G_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 $G_{n,j}$ ($j \in [L]$), and the background graph $G_{n,0}$, independently as follows.

1. For each edge $e \in E(G_{n,j})$ that is a loop, we add e to a *recycle* list that is assigned to $G_{n,j}$. Similarly, if $e \in E(G_{n,j})$ contributes to a multi-edge, we put all but one copies of this edge to the *recycle* list.
2. We shuffle the *recycle* list and, for each edge e in the list, we choose another edge e' uniformly from $E(G_{n,j}) \setminus \{e\}$ (not necessarily in the 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 a collective *global recycle* list.

As a result, after ignoring edges in the *global recycle* list, all community graphs are simple and the background graph is simple. However, as is the case in the original **ABCD** model, an edge in the background graph can form a multi-edge with an edge in a community graph. Another problem that might occur, specific to **ABCD+o**^{2} model, is that an edge from one community can form a multi-edge with an edge from a different but overlapping community. All of these problematic edges are added to the *global recycle* list. We merge all community graphs with the background graph. Finally, the *global recycle* list is transformed into a list of half-edges and new edges are created from it. We follow the same procedure as for the community graphs. However, we do not "give up" recycling and follow the process until all required edges are created. As the background graph is sparse, this final rewiring is very fast in practice.

3 Properties of the ABCD+o^2 Model

In this section, we present the results of some experiments highlighting properties of the **ABCD+o**^{2} model.

Degree distribution and community size distribution In the first experiment, we generate **ABCD+o**^{2} graphs with three degree sequences and with $n =$ 100,000. The minimum and the maximum degrees are fixed to be $\delta = 5$, $\Delta = 316 \approx \sqrt{n}$, but the power-law exponents vary: $\gamma \in \{2.2, 2.5, 2.8\}$. For a given integer k, let $f(k)$ be the experimental cumulative degree distribution, that is, $f(k)$ is the fraction of nodes of degree at least k. For a given set of parameters, the theoretical cumulative degree distribution is given by (1). We show that the experimental degree distributions are very close to the desired, theoretical, ones—see Figure 2 (Left) for the cumulative degree distributions of the three sequences.

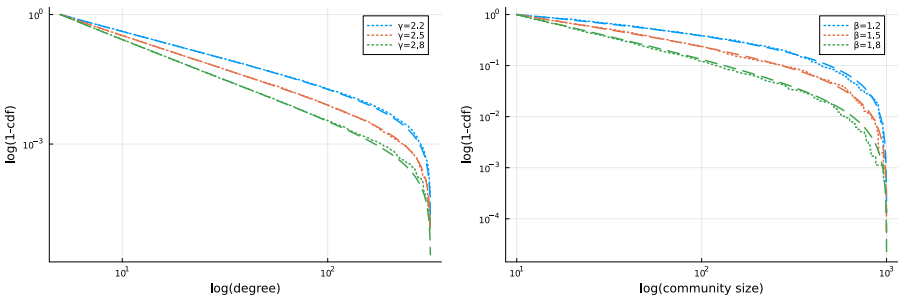

Fig. 2. Left: Empirical (dots) and theoretical (dashes) cumulative degree distributions for three degree sequences: $n = 100\,000, \delta = 5, \Delta = 316$. Right: Empirical (dots) and theoretical (dashes) cumulative community sizes distributions for three sizes sequences: $n = 100\,000, s = 10, S = 1\,000$.

Similarly, we generate three sequences of community sizes with $n =$ 100,000. The minimum and the maximum community size is fixed to be $s = 10$ and $S = 1000$ for the three sequences, but the power-law exponents vary: $\beta \in \{1.2, 1.5, 1.8\}$. We show that the experimental sequences are also very close to the desired ones—see Figure 2 (Right) for the cumulative community sizes distributions.

Overlapping of communities The **ABCD+o**^{2} model generates random graphs in which non-outlier nodes belong to η communities, on average. Let ρ_k be the fraction of non-outliers that belong to exactly k communities. The sequence $(\rho_k)_{k\geq 1}$ depends on the structure of the underlying hidden layer. To show one example, we generated five **ABCD+o**^{2} graphs with varying overlap parameter $\eta \in \{1, 1.5, 2, 2.5, 3\}$. Each of these graphs consists of $n =$ 10,000 nodes, including $s_0 = 250$ outliers, node degrees in range $[10, 100]$ with power law exponent $\gamma = 2.5$, and community sizes in range $[50, 1170]$ with power law exponent $\beta = 1.5$. The corresponding sequences $(\rho_k)_{k\geq 1}$ are presented in Figure 3 (left). In the case when $\eta = 3$, we also give the number and size of the non-empty overlaps between 2, 3 or 4 communities—see Figure 3 (right).

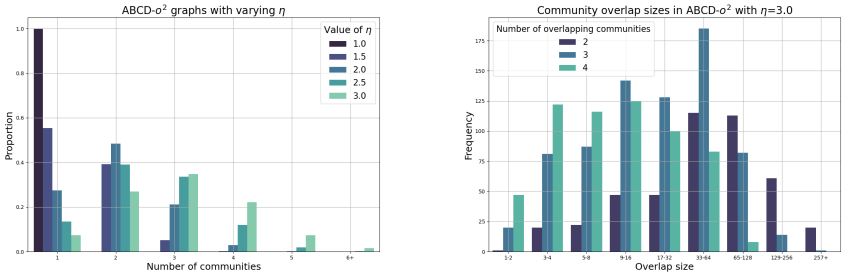

Fig. 3. Distribution of the number of community memberships for non-outlier nodes on **ABCD+o²**graphs with $n = 10{,}000$ and varying η (left), and distribution of overlap sizes in the non-empty intersections between 2, 3 or 4 communities (right).

Community association strength The **ABCD+o²** model aims to generate graphs in which nodes are much more associated with the communities they belong to than with other communities. The next experiment suggests that this goal is achieved.

In the coming experiment, we use three measures of community association strength: Internal Edge Fraction (IEF), Normalized Internal Edge Fraction (NIEF), and P-score (P). For node i and community C, $\text{IEF}(i, C)$ is the fraction of i's edges with the other end-point in C, $\text{NIEF}(i, C) = \text{IEF}(i, C) - \mathbb{E}[\text{IEF}(i, C)]$, where expectation is taken with respect to the Chung-Lu null model, and $P(i, C)$ is based on the classic p-value test, i.e., based on the probability that the $\text{IEF}(i, C)$ score was achieved randomly (again, using Chung-Lu as a null model). We point the interested reader to [2] for a more thorough discussion of these three measures.

For the experiment, we generate two **ABCD+o²** graphs, one with a low level of noise ($\xi = 0.35$) and the other with a high level of noise ($\xi = 0.65$). For both cases, we computed the following. For each of the three community association strength measures, and for each value of $K \in \mathbb{N}$, we investigate all nodes that belong to at least K communities and check what fraction of them have their K'th top ranked community (with respect to a given association strength) align with one of their ground-truth communities. The results are presented in Figure 4. The results suggest that each of the measures can accurately predict 1 or 2 communities a node is a member of, and with a low noise parameter, the prediction accuracy remains high as the number of communities increases.

4 Benchmarking Community Detection Algorithms

The main purpose of having synthetic models with ground-truth community structure is to test, tune, and benchmark community detection algorithms. To showcase **ABCD+o²** in this light, we use the model to evaluate the performance of five community detection algorithms. The algorithms are as follows.

Leiden [23]: a greedy algorithm that attempts to optimize the modularity function. Note that this algorithm returns a partition, and we use it merely as a benchmark to compare with algorithms that attempt to find overlapping communities.

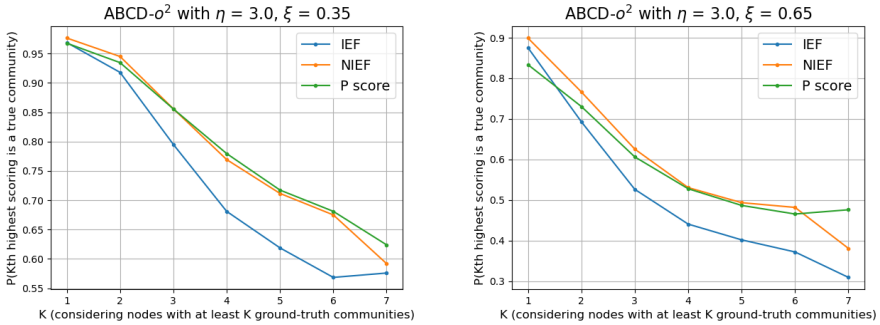

Fig. 4. The top-K CAS scores for nodes with K community memberships or more. For each score and each K, we show the proportion of true communities respectively for low noise **ABCD+o²**graphs (left) and high noise (right).

Clique Percolation [6]: an algorithm, based on a positive integer k, that finds all k-cliques and declares two such cliques adjacent if they share $k-1$ nodes. Then, the connected collections of cliques yield a collection of overlapping communities. In our experiment, we choose $k = 3$.

Edge Clustering [18]: an edge-partitioning algorithm that translates to overlapping clusters of nodes. Here, pairs of edges are measured based on similarity of neighbourhoods, and these similarity measures dictate the order in which edge-communities merge, starting from each edge in its own community. As edge-communities merge, the modularity is tracked on the line-graph, and the maximum modularity attained yields the edge-communities, which in turn yields overlapping node-communities.

Ego-Split [7]: a method which finds overlapping clusters in a graph G by applying a partitioning algorithm such as Leiden to an auxiliary graph G' and then mapping the resulting partition onto G. The auxiliary graph G' is constructed from G by creating multiple copies, or "egos", of each node based on its neighbourhood.

Ego-Split+CAS: the same algorithm as Ego-Split, but with a post-processing step that re-assigns nodes to communities based on the NIEF measure.

This is by no means an exhaustive list of community detection algorithms. We wish only to showcase the usefulness of **ABCD+o²** in comparing the quality of detection algorithms. The measure we use to determine the quality of a collection of communities is the overlapping Normalized Mutual Information (oNMI) measure: a similarity measure for two collections of subsets $\mathcal{X}, \mathcal{Y}$ of a set S [21].

The parameters of **ABCD+o²** with the most influence on the quality of detection algorithms are ξ (the level of noise) and η (the average number of communities a non-outlier is part of). Thus, we perform two versions of this experiment, one which varies $\xi \in \{0.15, 0.25, \ldots, 0.65\}$ and fixes $\eta = 2$, and the other which varies $\eta \in \{1, 1.5, 2, 2.5, 3\}$ and fixes $\xi = 0.15$. Figure 5 presents the results of the experiment. We see that Ego-Split+CAS performs the best overall, except when $\eta = 1$ in which case Leiden performs better. We also see a general trend of all algorithms performing worse as the graph gets noisier, either by increasing ξ or η. From numerous and varying tests, we have found in general that increasing η is far more damming to detection algorithms than increasing ξ.

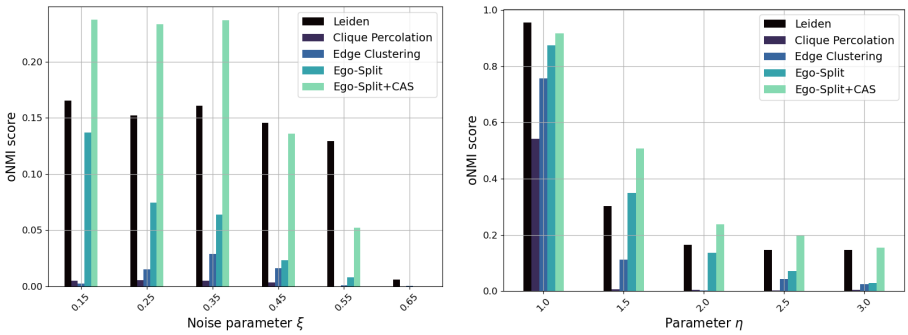

Fig. 5. The quality of five clustering algorithms on **ABCD+o**^{2} graphs with 10,000 nodes including 250 outlier nodes. In the left plot, we fix $\eta = 2.0$ and vary ξ. In the right plot, we fix $\xi = 0.15$ and vary η.

5 Conclusion

We presented **ABCD+o**^{2}: a generalization of the **ABCD+o** model that allows for overlapping communities. We then tested properties of this new model, emphasizing properties based on the new overlap parameter η. Finally, we showcased the model's ability to benchmark community detection algorithms and compare their quality.

This paper acts as a first step in our study of the **ABCD+o**^{2} model. In future work, we will study more properties of the model and compare our findings with (i) real networks containing overlapping ground-truth communities, and (ii) the overlapping **LFR** model. In particular, we believe that the nature of community overlap (based on the hidden, geometric reference layer) is more natural and realistic in **ABCD+o**^{2} than in **LFR** and we will explore this conjecture further.

We are interested in theoretical results of the **ABCD+o**^{2} model that generalize results of the **ABCD** and **ABCD+o** models. In [13] the modularity was studied and it was found that the maximum modularity came from the ground truth communities until a certain level of noise, after which a higher modularity could be attained. A similar behaviour should be seen with **ABCD+o**^{2} and its overlap parameter η. Additionally, in [3] it was shown that **ABCD** graphs exhibit self-similar behaviour, namely, the degree distributions of communities are asymptotically the same as the degree distribution of the whole graph (up to an appropriate normalization). We suspect that this self-similar property persists in **ABCD+o**^{2}.

Finally, we are interested in modifying the **ABCD+o**^{2} model in various ways. On the one hand, the underlying geometry is the key ingredient in forming overlaps between communities, and changing this geometry will surely change the behaviour of the overlap. Moreover, certain metric spaces may yield **ABCD+o**^{2} graphs with more realistic properties. On the other hand, the assignment of degrees to nodes can be tweaked, say, to bias large degrees towards nodes that are members of a large number of communities. In [26] it was shown that real networks tend to have a higher density of edges in the intersections of communities than in the communities themselves. We hope that by tweaking the degree-to-node assignment process in **ABCD+o**^{2} we can find this same density result.

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, Ryan DeWolfe, Bogumił Kamiński, Paweł Prałat, Aaron Smith, and François Théberge. Improving community detection via community association strength scores, 2025. URL: `https://arxiv.org/abs/2501.17817`, `arXiv:2501.17817`.
3. Jordan Barrett, Bogumił Kamiński, Paweł Prałat, and François Théberge. Self-similarity of communities of the ABCD model. *Theoretical Computer Science*, 1026:115012, 2025.
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. 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.
6. Imre Derényi, Gergely Palla, and Tamás Vicsek. Clique percolation in random networks. *Physical review letters*, 94(16):160202, 2005.
7. Alessandro Epasto, Silvio Lattanzi, and Renato Paes Leme. Ego-splitting framework: From non-overlapping to overlapping clusters. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 145–154, 2017.
8. Santo Fortunato. Community detection in graphs. *Physics reports*, 486(3-5):75–174, 2010.
9. Santo Fortunato and Marc Barthelemy. Resolution limit in community detection. *Proceedings of the national academy of sciences*, 104(1):36–41, 2007.
10. Michelle Girvan and Mark EJ Newman. Community structure in social and biological networks. *Proceedings of the national academy of sciences*, 99(12):7821–7826, 2002.
11. Steve Gregory. Fuzzy overlapping communities in networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2011(02):P02017, 2011.
12. 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.
13. 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.
14. 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.
15. Bogumił Kamiński, Paweł Prałat, and François Théberge. *Mining Complex Networks*. Chapman and Hall/CRC, 2021.
16. Bogumił Kamiński, Paweł Prałat, and François Théberge. Artificial benchmark for community detection with outliers (ABCD+o). *Applied Network Science*, 8(1):25, 2023.
17. Bogumił Kamiński, Paweł Prałat, and François Théberge. Hypergraph artificial benchmark for community detection (h–ABCD). *Journal of Complex Networks*, 11(4):cnad028, 2023.

18. Paul Kim and Sangwook Kim. Detecting overlapping and hierarchical communities in complex network using interaction-based edge clustering. *Physica A: Statistical Mechanics and its Applications*, 417:46–56, 2015. URL: https://www.sciencedirect.com/science/article/pii/S0378437114007936, doi:10.1016/j.physa.2014.09.035.
19. 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.
20. Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. Benchmark graphs for testing community detection algorithms. *Physical review E*, 78(4):046110, 2008.
21. Aaron F McDaid, Derek Greene, and Neil Hurley. Normalized mutual information to evaluate overlapping community finding algorithms. *arXiv preprint arXiv:1110.2515*, 2011.
22. Gergely Palla, Imre Derényi, Illés Farkas, and Tamás Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, 2005.
23. V. Traag, L. Waltman, and Nees Jan van Eck. From Louvain to Leiden: guaranteeing well-connected communities. *Scientific Reports*, 9:5233, 03 2019. doi:10.1038/s41598-019-41695-z.
24. Nicholas C Wormald. Generating random regular graphs. *Journal of algorithms*, 5(2):247–280, 1984.
25. Nicholas C Wormald et al. Models of random regular graphs. *London Mathematical Society Lecture Note Series*, pages 239–298, 1999.
26. Jaewon Yang and Jure Leskovec. Overlapping communities explain core–periphery organization of networks. *Proceedings of the IEEE*, 102(12):1892–1902, 2014. doi:10.1109/JPROC.2014.2364018.