Appendix

A More Details on Training VEPM

A.1 A pretrain-finetune scheme

Since modeling the likelihood of the labels involves stacking multiple modules embodied by deep neural networks, approximating the posterior $p_{\boldsymbol{\theta}}(\mathbf{Z} \mid \mathbf{A}, \mathbf{y}_o)$ is difficult from an optimization perspective. To effectively train VEPM, we break down the overall training pipeline into two phases: an *unsupervised pretrain*, followed by a *supervised finetune*. Specifically, we first pretrain the *inference network* by optimizing $\mathcal{L}_{\text{egen}} + \mathcal{L}_{\text{KL}}$ to link latent communities with edge generation. We then join the pretrained *inference network* up with the *generative network*, and train the entire architecture with the objective function given in Equation (5). In this stage, the latent communities are finetuned by $\mathcal{L}_{\text{task}}$ to align them to the task. The efficacy of pretraining the model with an unsupervised loss is also expressed in another work [43] that involves optimizing graph structure for GNNs with a graph autoencoder. The pretrain-finetune scheme is given in Algorithm 1.

Algorithm 1 The overall training algorithm of VEPM

Data: node features $\mathbf{X}$, observed edges $\mathbf{A}$, and observed labels $\mathbf{y}_o$.
Modules: Community ENCoder (CENC), Edge PARTitioner (EPART), Community-GNN BANK (CGNN_BANK) and REPresentation COMPoser (REPCOMP).
Parameters: $\boldsymbol{\phi}$ in the *inference network*, and $\boldsymbol{\theta}$ in the *generative network*.
Initialize $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$;
repeat
  $\mathbf{Z}, \mathbf{K}, \mathbf{\Lambda} \leftarrow \text{CENC}(\mathbf{A}, \mathbf{X})$;
  compute $\mathcal{L}_{\text{egen}}$ and $\mathcal{L}_{\text{KL}}$, given in Equation (5);
  $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} + \eta_{\text{unsup}} \cdot \nabla_{\boldsymbol{\phi}}(\mathcal{L}_{\text{egen}} + \mathcal{L}_{\text{KL}})$;
until CENC converges
repeat
  $\mathbf{Z}, \mathbf{K}, \mathbf{\Lambda} \leftarrow \text{CENC}(\mathbf{A}, \mathbf{X})$;
  $\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \cdots, \mathbf{A}^{(K)} \leftarrow \text{EPART}(\mathbf{Z}, \mathbf{A})$;
  for $step \leftarrow 1$ **to** M **do**
    $\hat{p}_{\mathbf{y}_o} \leftarrow \text{softmax}\big(\text{REPCOMP} \circ \text{CGNN\_BANK}(\mathbf{Z}, \mathbf{X}, \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \cdots, \mathbf{A}^{(K)})\big)$;
    compute $\mathcal{L}_{\text{task}}$, given in Equation (5);
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \eta_{\text{sup},\boldsymbol{\theta}} \cdot \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\text{task}}$;
  end for
  compute $\mathcal{L}$;
  $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} + \eta_{\text{sup},\boldsymbol{\phi}} \cdot \nabla_{\boldsymbol{\phi}} \mathcal{L}$;
until the whole model converges

A.2 Subsampling-Based Acceleration Algorithm

Like all neural graph generative models that model the connection statuses of all $\binom{N}{2}$ node pairs, *e.g.*, VGAE [7] and its various extensions, the time complexity of (pre)training VEPM is also $\mathcal{O}(N^2)$. Thus a direct implementation that considers all node pairs in each iteration will face the scalability issue as N increases. Fortunately, such computation burden could be reduced to $\mathcal{O}(N_{\mathcal{S}}^2)$ through subgraph-sampling-based acceleration, where $N_{\mathcal{S}}$ denotes the number of nodes in the subgraph after sampling, as analogous to FastGAE [30]. Specifically, we develop an explainable and flexible strategy to sample nodes from the original graph *with replacement*, where node i is sampled at probability p_i:

$$p_i = kq_i + (1-k)\frac{1-q_i}{N-1},\ q_i = \frac{f(\mathbf{x}_i)^\alpha}{\sum_{j=1}^{N}(f(\mathbf{x}_j)^\alpha)}. \tag{6}$$

With $f(\cdot)$ measuring the "importance" of node i, q_i denotes the probability of sampling node i for its relative importance in graph $\mathcal{G}$. The sharpness of the probability distribution $\{q_i\}_{i=1,N}$ is adjusted by the exponent α, where greater sharpness can be obtained with a higher value of α, and $\{q_i\}_{i=1,N}$ would be equivalent to a uniform distribution if α is set to 0. Likewise, $\frac{1-q_i}{N-1}$ can be interpreted as the

probability that node i is sampled for its relative unimportance. For better coverage, both "important nodes" and "unimportant nodes" are to be included in the sampled subgraph, with their percentages balanced by $k \in [0, 1]$, *i.e.*, the higher the value of k, the more "important nodes" are expected to be included in the subgraph. We use $k = 0.9$, $\alpha = 1$, and node degrees as $f(\cdot)$.

We record both node classification accuracy and the elapsed time per epoch for VEPM trained with and without subgraph sampling. The model is evaluated on Cora, Citeseer and Pubmed where $N_{\mathcal{S}}$ is set to 200. As shown in Table 7, the acceleration algorithm successfully reduces the time cost by a huge margin, it also results in a slight decrease in model performance and mild inflation in the error bar. However, the negative impacts are not significant enough to offset the achievement in scalability, especially considering the fact that with the implication in model performance, VEPM with scalable training still outperforms other baselines on two out of three benchmarks.

Table 7: Comparison of VEPMs with or without scalable training.

Hyperparameters	Cora		Citeseer		Pubmed	
	Accuracy (%)	Epoch duration	Accuracy (%)	Epoch duration	Accuracy (%)	Epoch duration
Full-batch Training	**84.3** $\pm$ 0.1	0.19s	72.5 $\pm$ 0.1	0.25s	**82.4** $\pm$ 0.2	7.28s
Scalable Training	83.4 $\pm$ 0.1	0.02s	71.8 $\pm$ 0.3	0.03s	81.6 $\pm$ 0.3	0.05s

B Statistical Properties of the Probabilistic Distributions

The Bernoulli-Poisson distribution, which augments binary edges into counts of edges, not only adds interpretability in terms of edge partition, but also is well suited for sparse graphs. The gamma prior not only ensures positivity and promotes sparsity on $\mathbf{Z}$, but also allows an analytic KL term by combining with the Weibull variational distribution, which is reparameterizable. Here are some properties of the Weibull distribution [44] that we would like to highlight:

• Similar density characteristics with gamma distribution

The Weibull distribution has similar characteristics with the gamma distribution, *i.e.*, the density functions of the two distributions are similar, and both enjoy the flexibility in modeling sparse and nonnegative latent representations.

$$\begin{aligned} &\text{Weibull}(k, \lambda) \text{ PDF: } p(x|k, \lambda) = \frac{k}{\lambda^k} x^{k-1} e^{(x/\lambda)^k}, \\ &\text{Gamma}(\alpha, \beta) \text{ PDF: } p(x|\alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x}. \end{aligned} \tag{7}$$

• Easily Reparameterization

The latent variable $x \sim \text{Weibull}(k, \lambda)$ can be easily reparameterized as

$$x = \lambda(-\ln(1 - \varepsilon))^{1/k}, \;\; \varepsilon \sim \text{Uniform}(0, 1), \tag{8}$$

leading to an expedient and numerically stable gradient calculation.

• Analytic KL-divergence

Moverover, the KL-divergence between the Weibull and gamma distributions has an analytic expression formulated as

$$\begin{aligned} \text{KL}(\text{Weibull}(k, \lambda)||\text{Gamma}(\alpha, \beta)) &= -\alpha \ln \lambda + \frac{\gamma\alpha}{k} \\ + \ln k + \beta\lambda\Gamma(1 + \frac{1}{k}) - \gamma - 1 - \alpha \ln \beta + \ln \Gamma(\alpha). \end{aligned} \tag{9}$$

C More Details on the Experiments

The source code of VEPM is publicly available at `https://github.com/YH-UtMSB/VEPM`, which is developed upon Pytorch 1.9.0 / 1.11.0 and Python 3.8. Experiments are carried out on NVIDIA GeForce GTX 1080-Ti, NVIDIA Quadro RTX 5000, and NVIDIA RTX A4000.

C.1 Data preparation

Table 8: Statistics of the datasets used for node- and graph-level classification tasks.

Task	Node Classification				Graph Classification							
Dataset	**Cora**	**Citeseer**	**Pubmed**	**WikiCS**	**IMDB-B**	**IMDB-M**	**MUTAG**	**PTC**	**NCI1**	**PROTEINS**	**RDT-B**	**RDT-M**
Graphs	1	1	1	1	1000	1500	188	344	4110	1113	2000	5000
Edges	5,429	4,732	44,338	216,123	96.5	65.9	19.8	26.0	32.3	72.8	497.7	594.8
Features	1,433	3,703	500	300	65	59	7	19	37	3	566	734
Nodes	2,708	3,327	19,717	11,701	19.8	13.0	17.9	25.5	29.8	39.1	429.6	508.5
Classes	7	6	3	2	3	10	2	2	2	2	2	5

For the node classification task, we evaluate VEPM on four benchmarks, including three citation networks and one Wikipedia-based online article network. The citation networks are Cora, Citeseer and PubMed, in which the academic literature is treated as nodes and citations are treated as edges. The node features are bag-of-words document representations. The online article network is WikiCS, which treat each article as a node, and the hyperlinks between these articles as the edges. All four networks used for node classification are undirected, node-attributed graphs. The node features for the citation networks are bag-of-words document representations, for the online article network, the features for each article are the average GloVe word embeddings [45].

For graph classification, we consider four bioinformatics datasets (MUTAG, PTC, NCI1, PROTEINS) and four social network datasets (IMDB-BINARY, IMDB-MULTI, REDDIT-BINARY and REDDIT-MULTI). The input node features are crafted in the same way as Xu et al. [6]. We summarize some key statistics of the benchmarks we used for evaluating VEPM in Table 8.

C.2 Experimental protocols

To make a fair comparison with the baselines, for semi-supervised node classification on the citation networks (Cora, Citeseer, PubMed), we use the train/test/validation split standardized by the GCNs [5], and adopt a cross-validation strategy for hyperparameter determination and early stop. For node classification on WikiCS, we follow the protocol of Kim and Oh [46]. For graph classification, we evaluate VEPM under two experimental protocols. The results recorded in Table 2 are obtained following the 10-fold cross-validation-based evaluation protocol proposed by Xu et al. [6]. Since this protocol is criticized for being prone to overestimating model performance [47], we also evaluate our model following Zhang and Chen [33], who conduct a more rigorous train-validation-test protocol. The results obtained by the second protocol are shown in Table 3.

D Additional Ablation Studies

D.1 Alternatives of input features

We evaluate the model performance with four types of inputs: random noise, hand-crafted features by Xu et al. [6], node-community affiliation scores (*i.e.*, $\mathbf{Z}$), and the combination of the last two. The quality of these inputs is evaluated by the graph classification accuracy as summarized in Table 9. Two conclusions could be drawn: (i) $\mathbf{Z}$ is comparably informative to the end task as hand-crafted node features, both are significantly better than random noise (see rows 1 – 3 of Table 9); (ii) the information $\mathbf{Z}$ and $\mathbf{X}$ hold for the task enhances the discriminative level of learned representations in a complementary manner (see rows 2, 4 of Table 9). These results indicate that inferring $\mathbf{Z}$ and treating it as additional node features would benefit task learning.

Table 9: Comparison of VEPMs with various input node features.

Random	Hand-crafted	Community-based	**IMDB-B**	**IMDB-M**	**MUTAG**	**PTC**	**NCI1**	**PROTEINS**
✔			64.7 ± 1.6	42.3 ± 1.5	84.6 ± 4.3	63.6 ± 2.0	67.5 ± 2.1	76.5 ± 3.5
	✔		$\mathbf{80.3} \pm 2.0$	53.5 ± 2.6	93.1 ± 5.0	74.7 ± 4.1	68.0 ± 2.1	77.9 ± 2.7
		✔	74.7 ± 5.1	51.5 ± 2.0	84.6 ± 5.5	68.9 ± 3.9	80.8 ± 1.4	75.7 ± 3.8
	✔	✔	76.7 ± 3.1	$\mathbf{54.1} \pm 2.1$	$\mathbf{93.6} \pm 3.5$	$\mathbf{75.6} \pm 5.9$	$\mathbf{83.9} \pm 1.8$	$\mathbf{80.5} \pm 2.8$

D.2 Alternatives of training schemes

In this part, we compare the performance of VEPM that undergoes both pretraining and finetuning against that of a VEPM trained with Equation (5) from scratch. Results in Table 10 shows that comparing with training from scratch, the state of the *community encoder* after pretraining is better for the classification tasks, suggesting the benefits brought by pretraining.

Table 10: Comparison of VEPMs with different training schemes.

	IMDB-B	**IMDB-M**	**MUTAG**	**PTC**	**NCI1**	**PROTEINS**
trained from scratch	61.0 ± 3.6	46.8 ± 0.8	86.6 ± 6.6	68.6 ± 2.0	67.2 ± 8.4	60.6 ± 4.1
pretrain + finetune	**76.7** ± 3.1	**54.1** ± 2.1	**93.6** ± 3.5	**75.6** ± 5.9	**83.9** ± 1.8	**80.5** ± 2.8

Why training from scratch is not good. As shown in Section 4.2, VEPM's performance is dependent on the quality of edge partition. A relatively meaningful edge partition as the initial state for finetuning would facilitate information exchange between the model and the task, and thus is beneficial for optimization. Contrariwise, a less meaningful edge partition as the initial state would make it more difficult for the model to converge to an optimal state. We notice that the performance of VEPM, when trained from scratch, is between baselines and a random guess, and we assume that the cause of such behavior is a less meaningful edge partition obtained by that training scheme. Intuitively speaking, considering an extreme case where edge partition is not relevant to the task, which would render community-specific node representations meaningless for the task, and by passing them to the representation composer, which is a layer-reduced GIN, it is plausible that a worse result would be obtained. In the meantime, since the original graph is used in the representation composer, which still carries some information about the task, the model performance is protected from a total failure.

D.3 Sensitivity analyses

The number of (meta)communities. We hold all other hyperparameters as constants and run 10-fold cross-validations with $K \in \{4, 5, 6, 7, 8\}$, with K denoting the number of (meta)communities. From the results in Table 11, we can find that (i) the variation in graph classification performance that VEPM achieves with different values of K is not significant, most of the results recorded in the same column are within one standard deviation with each other; (ii) there is not a collective pattern between model performance and K shared across all benchmarks, in other words, not a single setting of K could consistently outperform other settings on all datasets. Given the above observations, we conclude that the performance of VEPM performance on graph classification is not significantly influenced by the number of metacommunities within a reasonable range.

Table 11: Comparisons of VEPMs with different numbers of metacommunities. * denotes the value we use in 10-fold cross-validations.

# metacommunities	**IMDB-B**	**IMDB-M**	**MUTAG**	**PTC**	**NCI1**	**PROTEINS**
4*	76.7 ± 3.1	54.1 ± 2.1	93.6 ± 3.4	**75.6** ± 5.9	**83.9** ± 1.8	**80.5** ± 2.8
5	77.0 ± 1.8	52.9 ± 2.2	94.1 ± 6.3	72.4 ± 6.5	80.8 ± 2.2	78.1 ± 2.3
6	77.3 ± 3.2	53.9 ± 2.7	91.5 ± 6.4	73.5 ± 5.1	81.0 ± 3.0	75.8 ± 2.6
7	77.0 ± 2.5	**54.2** ± 1.7	**94.7** ± 4.2	71.8 ± 5.1	82.0 ± 1.3	79.0 ± 2.9
8	**78.3** ± 2.1	54.2 ± 2.1	92.5 ± 4.3	71.2 ± 3.7	78.3 ± 2.9	78.2 ± 2.9

Network depth allocation of the *generative network*. In this part, we focus on studying the potential influence of different network structures on graph classification tasks. As we decide to make the *generative network* in VEPM to have a comparable number of parameters with GIN [6], the variation space left for the network structure is at how to distribute the 4 GIN layers to the *community-GNN bank* and *representation composer*. We use the name convention VEPM-$\{L_1\}$-$\{L_2\}$ to denote the variant with L_1 GIN layers in the *community-GNN bank* and L_2 GIN layers in the *representation composer*, where $L_1, L_2 \in \mathbb{Z}_+$ and $L_1 + L_2 = 4$. We hold all other hyperparameters as constants and evaluate VEPM under 3 structures: VEPM-$\{1\}$-$\{3\}$, VEPM-$\{2\}$-$\{2\}$ and VEPM-$\{3\}$-$\{1\}$. From the results shown in Table 12, the graph classification performance that VEPM achieves at different combinations of L_1 and L_2 are quite close to each other, the differences among the results in each column are at a low statistical significance level. We hence conclude that the predictive power of VEPM is relatively stable at the variation of the network structure of the *generative network*.

Table 12: Comparisons of VEPMs with various network structures. * denotes the structure we adopt in 10-fold cross-validations.

Network Structures	**IMDB-B**	**IMDB-M**	**MUTAG**	**PTC**	**NCI1**	**PROTEINS**
VEPM-{1}-{3}	**77.8** $\pm$ 2.0	**56.3** $\pm$ 2.1	91.0 $\pm$ 4.9	73.3 $\pm$ 3.6	81.2 $\pm$ 2.5	77.4 $\pm$ 3.6
VEPM-{2}-{2}*	76.7 $\pm$ 3.1	54.1 $\pm$ 2.1	**93.6** $\pm$ 3.4	**75.6** $\pm$ 5.9	83.9 $\pm$ 1.8	**80.5** $\pm$ 2.8
VEPM-{3}-{1}	74.5 $\pm$ 2.2	54.3 $\pm$ 3.2	87.0 $\pm$ 6.8	66.3 $\pm$ 5.0	**84.0** $\pm$ 2.7	78.9 $\pm$ 3.2

E The Characteristics of the Detected Communities

We summarize the patterns of communities detected by some well-established community detection / graph clustering / graph partition algorithms and compare them with VEPM.

Table 13: Characters of communities detected by various algorithms.

	METIS [48]	**Spectral Clustering / Min-Cut** [49]	**SBM** [50]	**MMSB** [10]	**VEPM**
Equisized	YES	N/A	N/A	N/A	N/A
High (low) local edge density	YES	YES	YES	YES	YES
Overlapping membership	NO	NO	NO	YES	YES
Task-dependent	NO	NO	NO	NO	YES

- Equisized: the size of communities are approximately the same.
- High (low) local edge density: the intra-community connection rate is specified to be high or low.
- Overlapping membership: nodes are permitted to be affiliated with multiple communities simultaneously.
- Task-dependent: the community membership is affected by a downstream task.

F Limitations and Broader Impacts

While our model exhibits superiority over traditional convolution-based graph neural networks on both node-level and graph-level representation learning tasks, several caveats are noteworthy. From the perspective of model design, we only focus on intra-community node interaction in our graph generative model, while it is a common practice and usually leads to good results in modeling homophilous graphs, it cannot well explain the edges generated as a result of cross-community node interaction, which potentially limits its modeling performance on heterogeneous graphs.

VEPM offers a new perspective to model the network with latent communities, it separates latent communities from the entire network and models them individually. Used with goodwill, it can understand what optimally benefits each community and help policy-making accordingly. However, along with its modeling power that could be harnessed to benefit communities, it can be dangerous if put into socially harmful applications. After all, no model is self-immune to malicious purposes, the potential misusages of VEPM include but are not limited to a group-targeted broadcast of fake news or conspiracy theories.

G Partitioned Adjacency Matrix

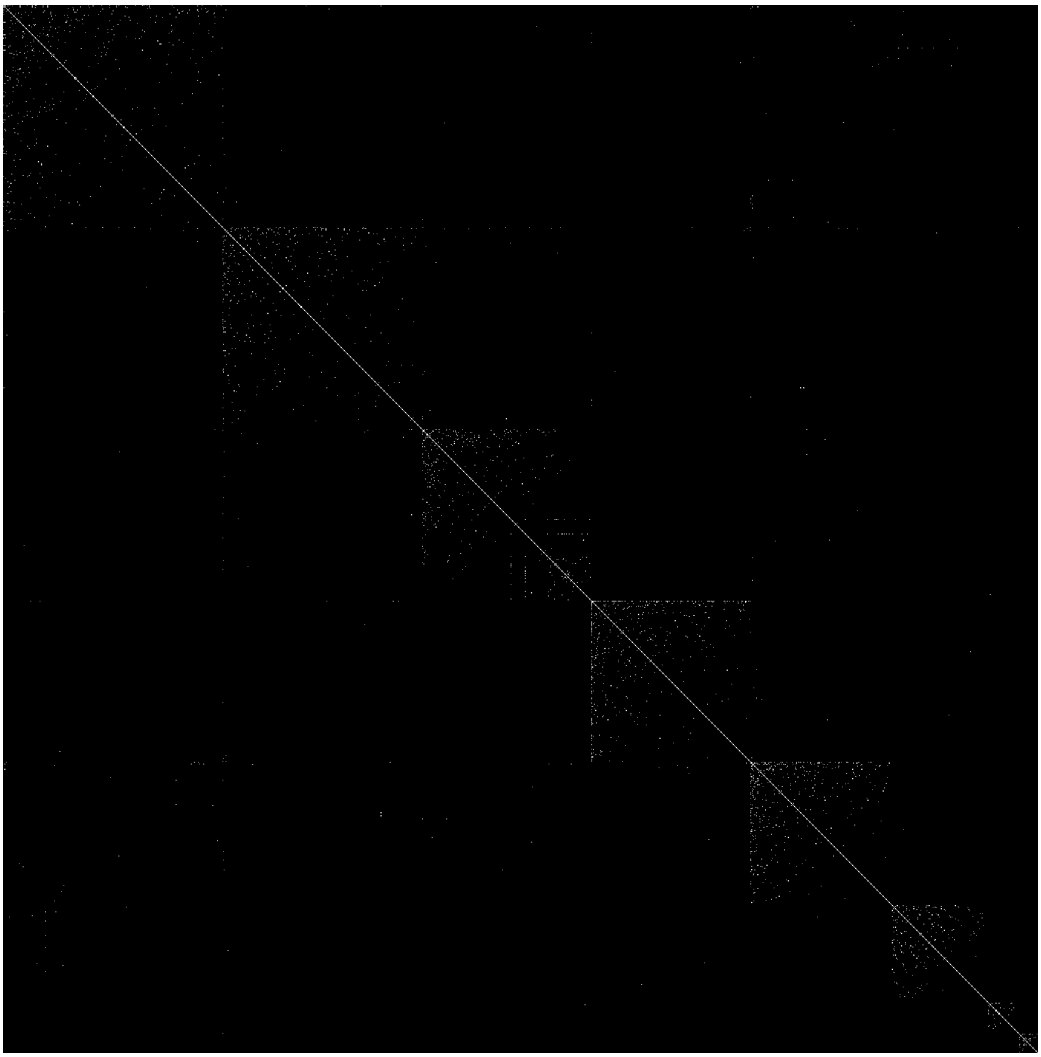

Figure 6: $\mathbf{A}$, the entire adjacency matrix of the Cora dataset.

Figure 7: $\mathbf{A}^{(1)}$, the partitioned graph corresponding to the largest metacommunity.

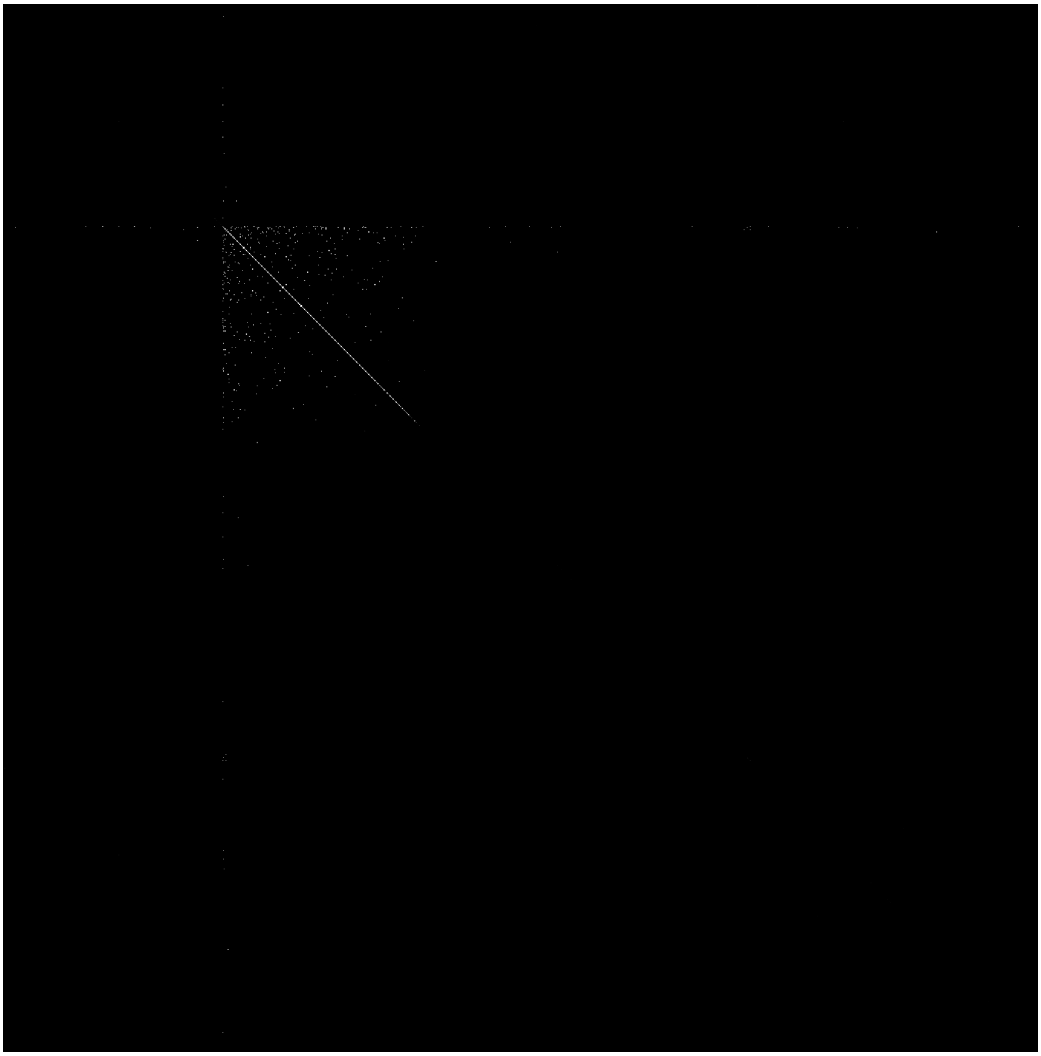

Figure 8: $\mathbf{A}^{(2)}$, the partitioned graph corresponding to the second largest metacommunity.

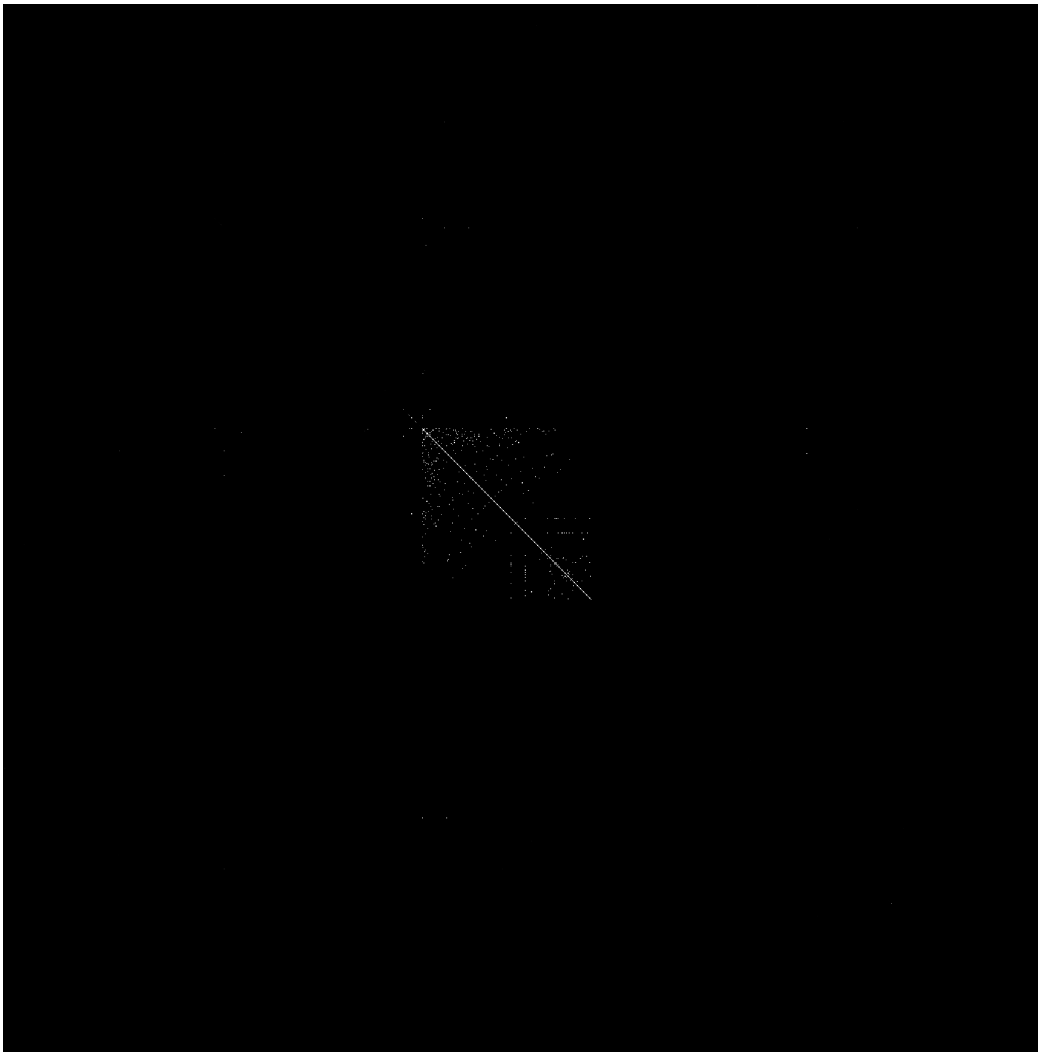

Figure 9: $\mathbf{A}^{(3)}$, the partitioned graph corresponding to the third largest metacommunity.

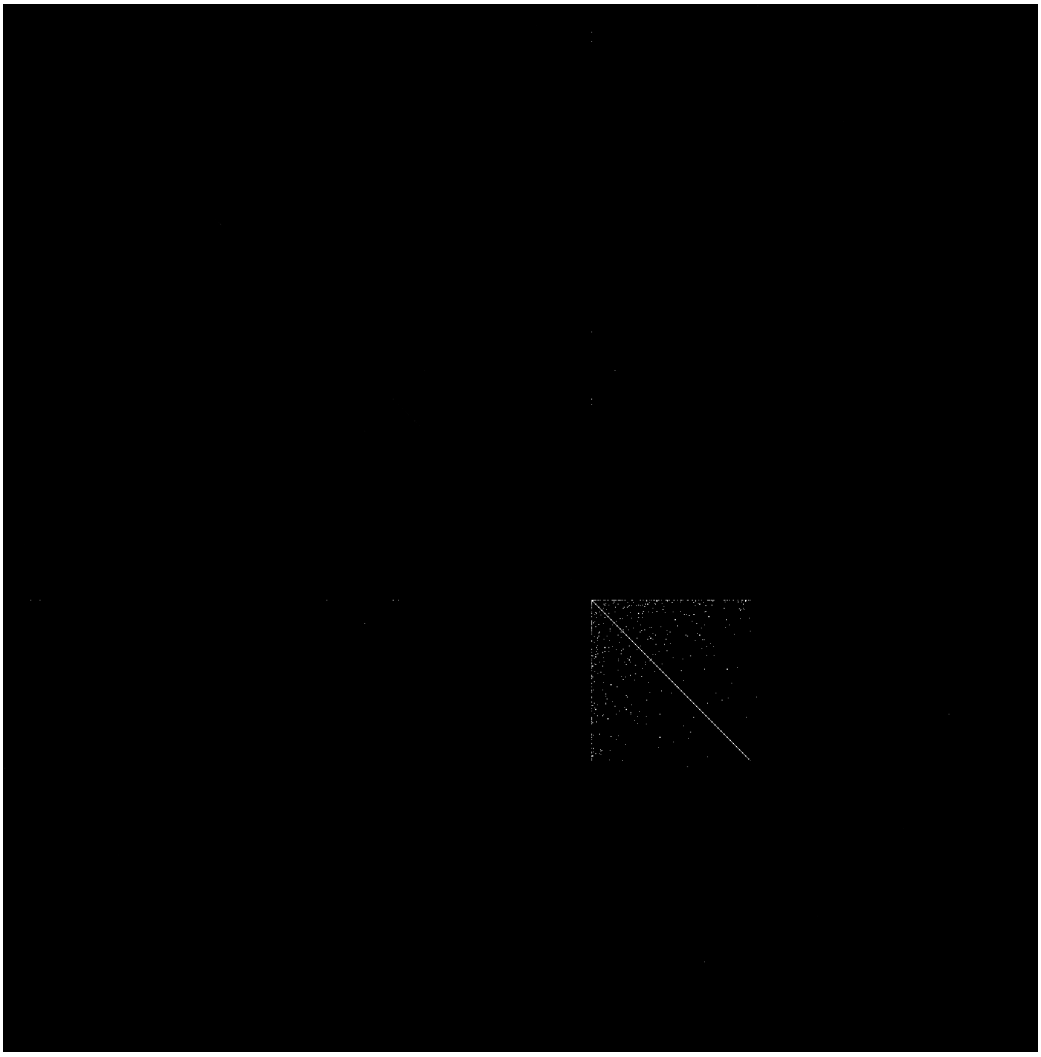

Figure 10: $\mathbf{A}^{(4)}$, the partitioned graph corresponding to the fourth largest metacommunity.

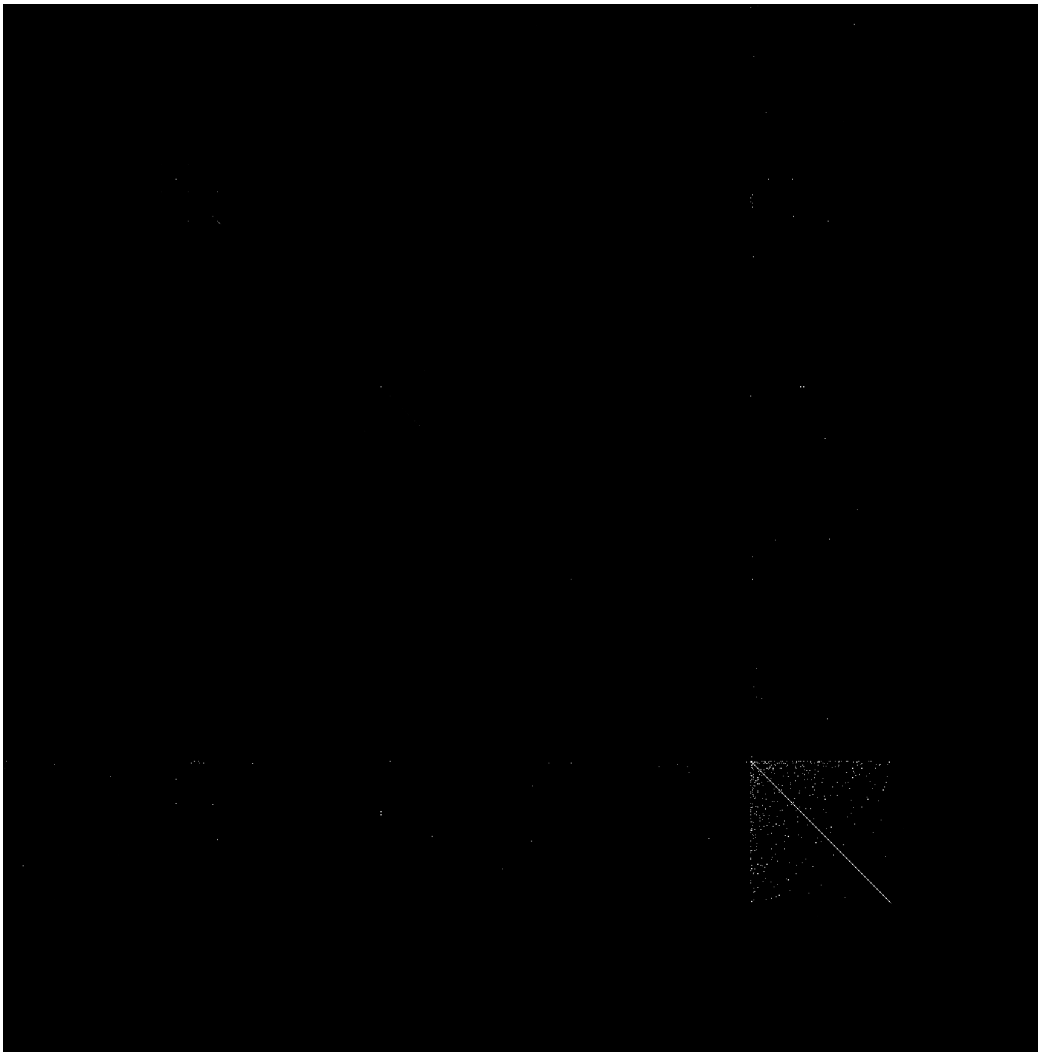

Figure 11: $\mathbf{A}^{(5)}$, the partitioned graph corresponding to the fourth smallest metacommunity.

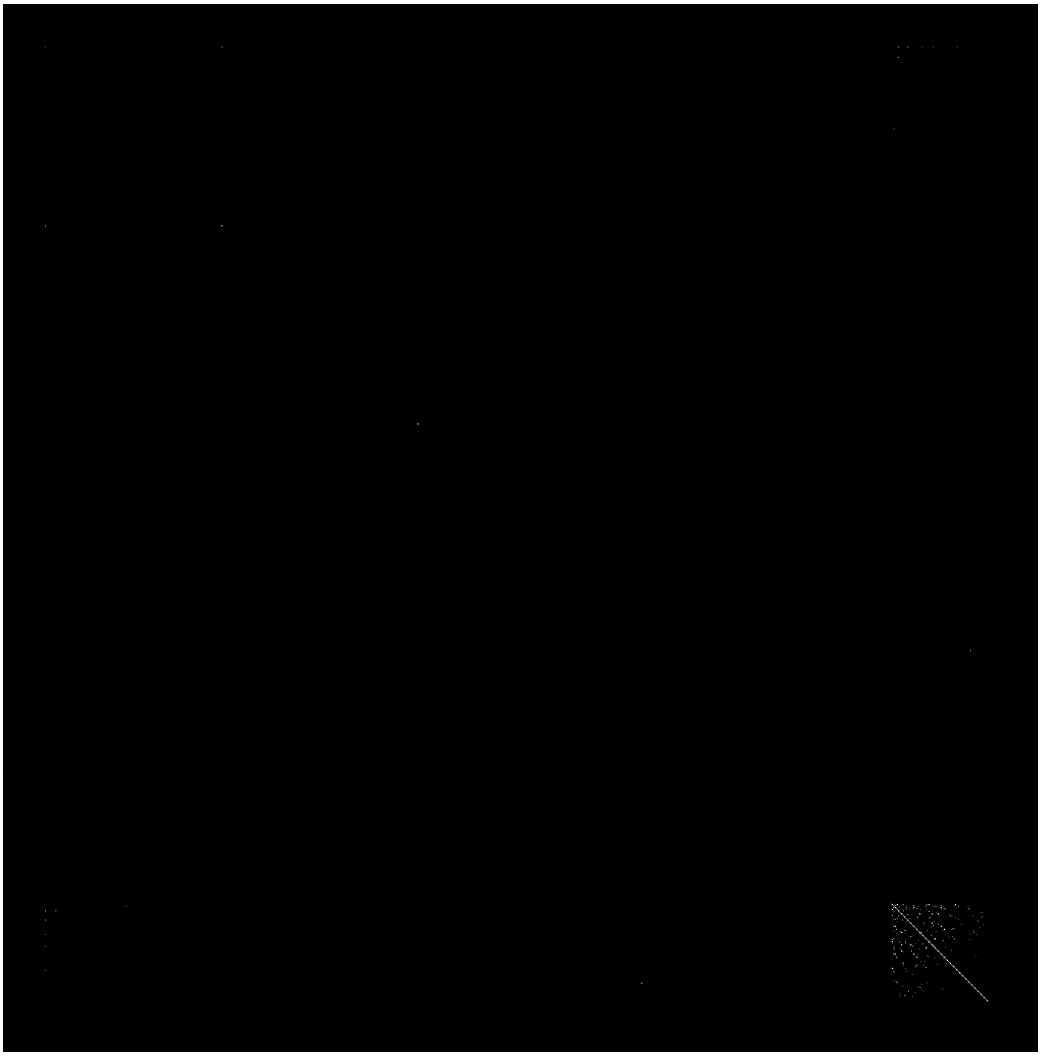

Figure 12: $\mathbf{A}^{(6)}$, the partitioned graph corresponding to the third smallest metacommunity.

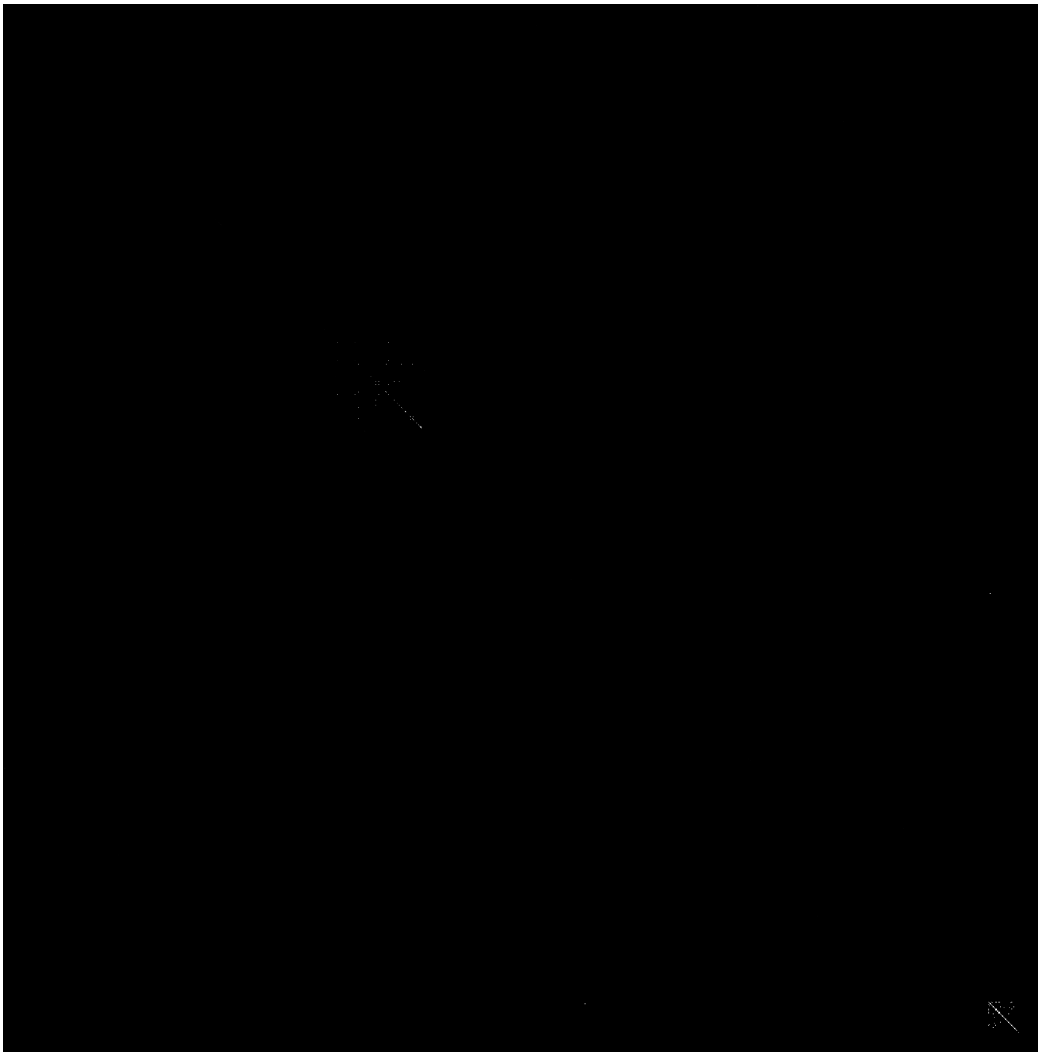

Figure 13: $\mathbf{A}^{(7)}$, the partitioned graph corresponding to the second smallest metacommunity.

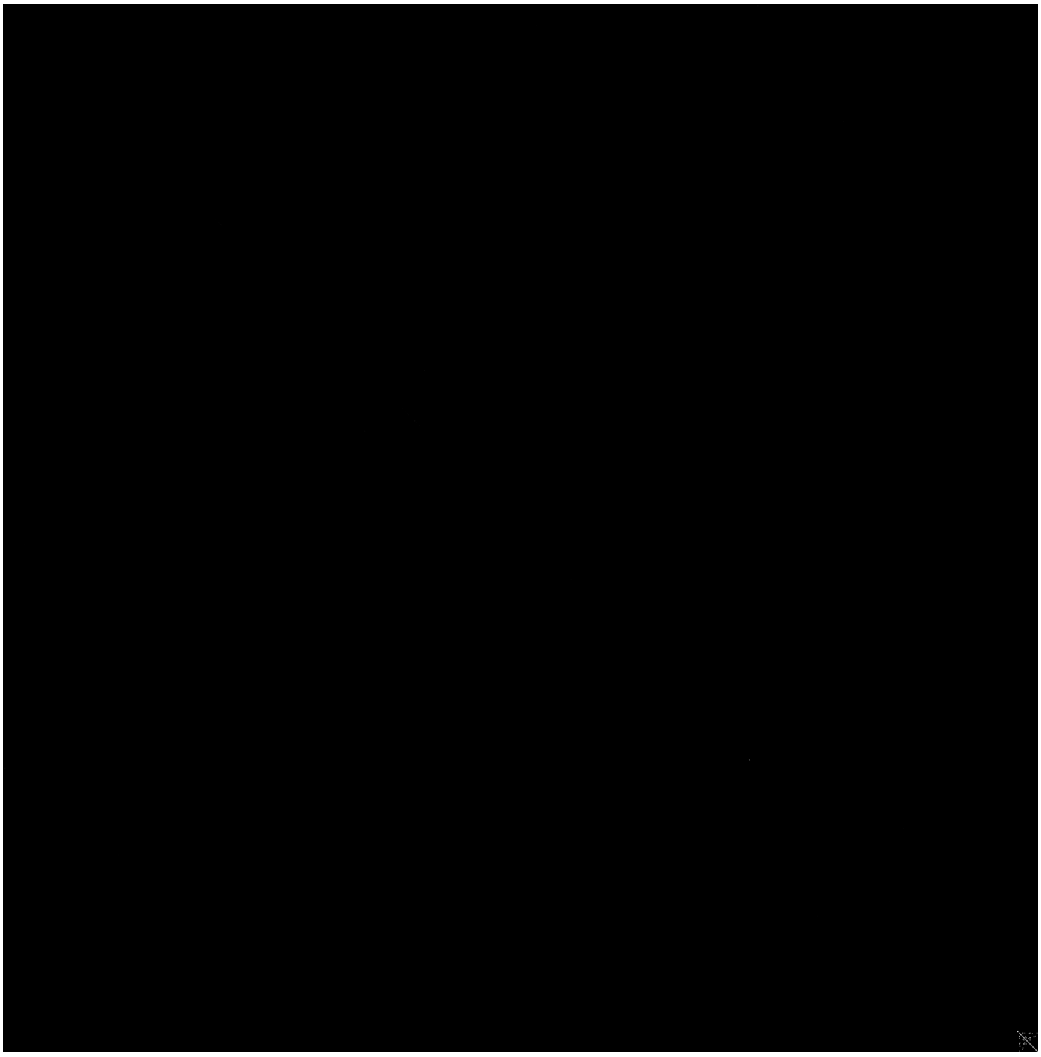

Figure 14: $\mathbf{A}^{(8)}$, the partitioned graph corresponding to the smallest metacommunity.