1 Model setup

The time and space complexity of spectral clustering depends on the type of generating model, as well as the method used to construct the affinity matrix A. For instance, in Gaussian mixture models, using the σ-neighborhood method typically results in a sparser affinity matrix compared to using kernels. Moreover, a random graph model such as the stochastic blockmodel usually leads to a sparse affinity matrix that one can take advantage of in terms of computation speed and memory.

In this report, we focus on the four parameter stochastic blockmodel [4] parametrized by k, s, p and q, where there are k blocks each containing s nodes. Hence, there are $n = ks$ nodes in total, where the probability of a connection between nodes in the same block is $p + q$, and the probability of a connection between nodes in separate blocks is q. Although highly stylized, the benefit of this structure is that under certain conditions, the cluster information is completely encoded in the eigenvectors of $\mathbb{E}A$, where A is an adjacency matrix obtained from SBM(k, s, p, q). Under the four parameter SBM, the clusters are balanced. Further assume for simplicity that there exist self loops, and that the nodes are planted in the following way (when $k = 2$):

$$\mathbb{E}A = \begin{bmatrix} (p+q)\mathbf{1}_s\mathbf{1}_s^\top & q\mathbf{1}_s\mathbf{1}_s^\top \\ q\mathbf{1}_s\mathbf{1}_s^\top & (p+q)\mathbf{1}_s\mathbf{1}_s^\top \end{bmatrix}.$$

It can be seen that the matrix $\mathbb{E}A$ has rank $k = 2$, and has nonzero eigenvalues $(p + 2q)n/2$ and $pn/2$ with corresponding eigenvectors $\frac{1}{\sqrt{n}}\mathbf{1}_n$ and $\frac{1}{\sqrt{n}}[\mathbf{1}_s^\top, -\mathbf{1}_s^\top]^\top$. In other words, the second nonzero eigenvector holds all the cluster information. For $k \geq 3$, it can be seen that $\mathbb{E}A$ has an eigenvalue $(p + kq)s$ of multiplicity one with corresponding eigenvector $\frac{1}{\sqrt{n}}\mathbf{1}_n$, and an eigenvalue ps of multiplicity $k-1$. It is clear that no matter which basis we choose for the eigenspace of ps, we will get k distinct rows[1] in the matrix U whose columns are the eigenvectors corresponding to the k nonzero eigenvalues of $\mathbb{E}A$ [8]. In this setting, provided that $q > 0$ and $\frac{\|A-\mathbb{E}A\|_2}{ps} \ll p\sqrt{2s}$ (left hand side is the eigenvector perturbation bound, and the right hand side is the population cluster center separation), we can invoke the Davis-Kahan theorem [7] to guarantee the rows of the sample eigenvector matrix $\hat{U}$ can be perfectly clustered [3] with high probability. Figure 1 shows an example where we are able to cluster the nodes perfectly along with another example where many of the nodes will be misclustered due to high eigenvector fluctuations.

2 Complexity analysis

Algorithm 1 Adjacency spectral clustering of four parameter SBM

Require: k, s, p, q.

1: Generate adjacency matrix A based on SBM(k, s, p, q).
2: Compute the degrees of each node, and check for high degree nodes.
3: Compute regularized $\tilde{A}$ by zeroing out the rows and columns of A corresponding to high degree nodes.
4: Compute truncated top-k eigenvalue decomposition $\tilde{A} \approx \hat{U}\hat{D}\hat{U}^\top$.
5: Perform k-means on the rows of $\hat{U}$.

2.1 Time complexity

It typically takes $O(n^2)$ operations to generate a $n \times n$ random matrix, and $O(nd)$ to compute the degrees of a sparse adjacency matrix whose average degree is d. Both of these tasks are simple routines which are easily parallelizable. A naive (hence unrecommended) way to compute the truncated eigenvalue decomposition in step 4 of algorithm 1 would be to compute the full eigenvalue decomposition of A, which takes $O(n^3)$ flops. In order to avoid computing the full eigenvalue decomposition, one can use `eigs` in Matlab, which is also provided in the R package `RSpectra`. Alternatively, `partial_eigen` in the R package `irlba` is an option, although `partial_eigen` is limited to symmetric matrices. These functions only compute the top-k eigenvalues and eigenvectors, and moreover is able to take A in sparse format to further reduce operation counts.

[1] These k distinct rows are often referred to as the population cluster centers.

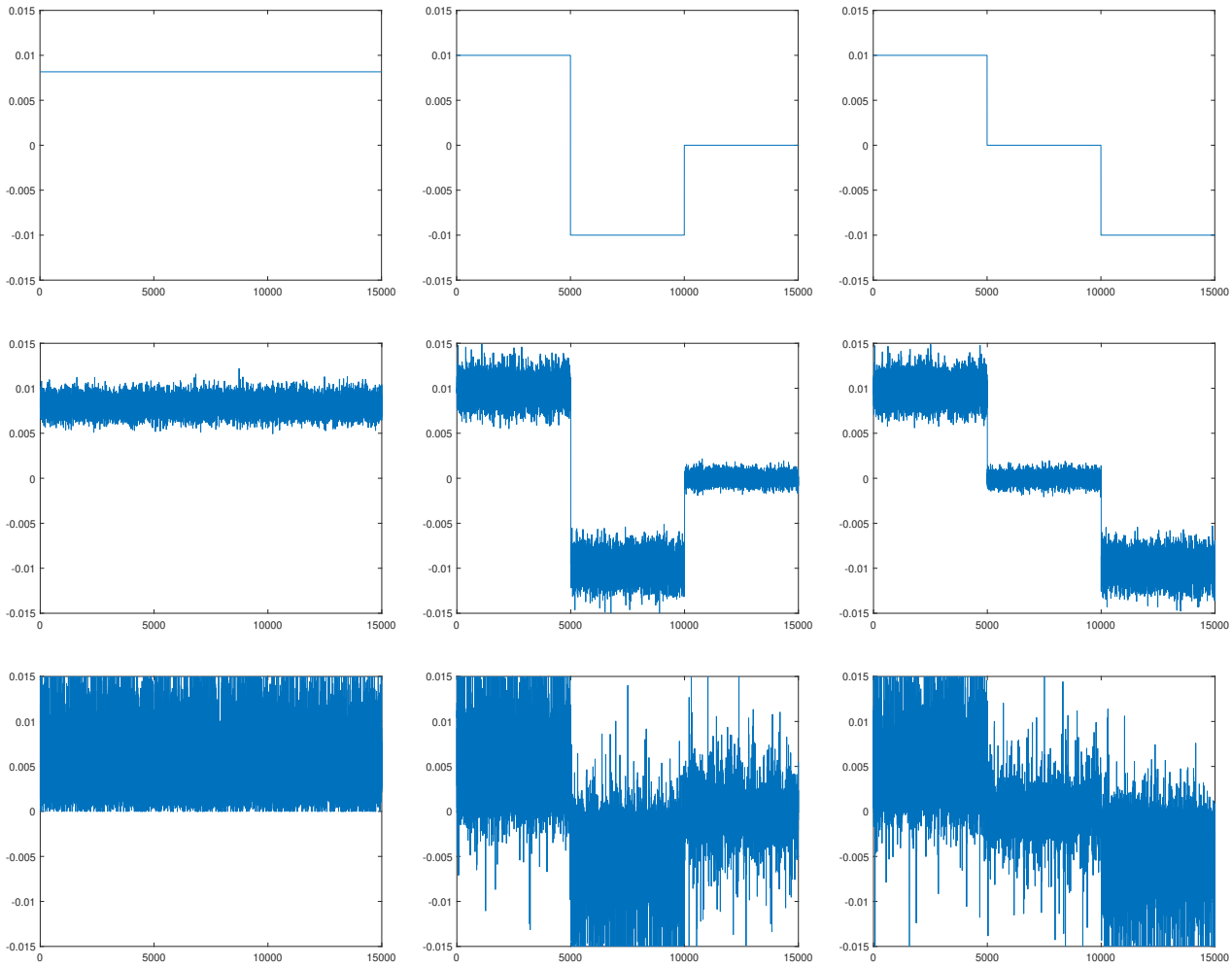

Figure 1: The profile of top three eigenvectors of $\mathbb{E}A$ (top row) and A (bottom two rows), where $k = 3$, $s = 5e+3$. $p = 1e-2$, $q = 1e-3$ in the middle row, and $p = 6e-4$, $q = 6e-5$ in the last row.

The function `eigs` is based on ARPACK [2], which includes the implicitly restarted Arnoldi method (IRAM) and implicitly restarted Lanczos method (IRLM) [5]: two methods that gained popularity due to their performance on computing (approximate) extreme eigenpairs of large sparse matrices[2]. Let m denote the dimension of the Krylov subpsace that is constructed before each restart, and let k denote the number of eigenpairs to be computed. The implicitly restarted Arnoldi method combines an implicitly shifted QR iteration with r Arnoldi iterations, where $m = k+r$. In our setting, we assume $k, m \ll n$. The implicitly shifted QR step involves r iterations of $m \times m$ QR factorizations resulting in $O(rm^3)$ flops, which is negligible in terms of time complexity. The r-step Arnoldi iteration amounts to computing r times a matrix-vector product of the form Av, which requires $O(rn^2)$ flops, and can be reduced to $O(rnd)$ flops, where d is the average degree of the nodes. Repeating until convergence thus results in $O(trnd)$ complexity for IRAM, where t is the number of iterations until convergence. There is no exact guarantee for t, since t depends on the distribution of eigenvalues of A as well as the starting point. In our setting, the top k eigenvalues of A are well separated from the remaining ones (justified by Weyl's inequality), hence we expect t to be small. In the numerical experiments in section 3, `eigs` usually converges in $t = 2$ iterations when $m = 20$ (default value of subspace size in `eigs`).

The $O(n^2)$ complexity of generating A cannot be significantly reduced. Hence, generating the adjacency matrix is the bottleneck in clock time without the help of parallel computing resources, or an approximation scheme.

[2] `partial_eigen` uses the implicitly restarted Lanczos bidiagonalization algorithm: a variant of implicitly restarted Lanczos.

2.2 Space complexity

The memory requirement for IRAM is $2nk + O(k^2)$ [5], which is because only the vectors of the form Av are stored. In general, storing the whole adjacency matrix requires $O(n^2)$ space. However, when A is an undirected graph, the space complexity can be reduced if we only generate and store the edge set of the graph. For node i, let d_i^+ denote the number of nodes whose indices are greater than i that are connected to node i. Then, the memory footprint of storing A can be reduced to $O\left(n\sum_{i=1}^n d_i^+\right)$. It is clear that $\sum_{i=1}^n d_i^+$ is proportional to np, which means that we can only store very sparse graphs for very large n. Alternatively, one might avoid short-term memory limitations by writing a large dense matrix on a hard disk drive row by row, then streaming the matrix several rows at a time since the matrix A only needs to be accessed via the matrix product Av. This reduces the short-term memory requirement to $O(n)$, although writing and reading a matrix to and from the hard disk drive takes longer to execute, thus requiring powerful computing resources to run in a reasonable amount of time.

3 Numerical experiments

The following experiments were conducted on a personal laptop equipped with an Intel i7-1065G7 (1.30GHz base frequency) processor with four cores and a 16GB RAM. Since a double precision number contains 64 bits (8 bytes), my RAM runs out of memory at a full square matrix of size $n \approx 4e+4$, or a sparse matrix whose number of nonzero entries is on the order of $O(n^8)$.

In table 1 and figure 2, "Step 1" refers to the elapsed time for generating A, "Step 2" refers to the elapsed time for computing the truncated eigenvalue decomposition using `eigs`, and "Step 3" refers to the elapsed time for performing k-means on the rows of U. "Iter." is the number of IRAM iterations (t above) performed inside `eigs`. Step 1 utilizes multithreading on the four cores of my laptop, wheras steps 2 and 3 are inherently serial.

k	s	n	p	q	Step 1	Step 2	Step 3	Total	Iter.	Miscluster rate
4	5e+3	2e+4	1e-4	1e-5	3.22e+0	1.79e-1	1.14e-2	3.41e+0	11	0.2498
4	5e+3	2e+4	1e-3	1e-4	3.22e+0	8.75e-2	6.02e-2	3.37e+0	6	0.0146
4	5e+3	2e+4	1e-2	1e-3	3.27e+0	9.83e-2	1.31e-2	3.38e+0	2	0
4	5e+3	2e+4	1e-1	1e-2	5.06e+0	7.01e-1	1.21e-2	5.26e+0	2	0
4	5e+3	2e+4	9e-1	1e-1	2.07e+1	4.84e+0	1.18e-2	2.56e+1	1	0
4	2.5e+2	1e+3	1e-3	1e-4	3.89e-1	8.99e-3	4.82e-3	4.03e-1	1	0.249
4	1e+3	4e+3	1e-3	1e-4	6.82e-1	9.49e-2	9.73e-3	7.87e-1	15	0.247
4	2.5e+3	1e+4	1e-3	1e-4	1.32e+0	2.28e-1	4.18e-2	1.59e+0	16	0.004
4	1e+4	4e+4	1e-3	1e-4	1.11e+1	1.59e-1	4.85e-2	1.13e+1	4	0.001
4	2.5e+4	1e+5	1e-3	1e-4	6.55e+1	9.09e-1	3.84e-1	6.68e+1	3	0

Table 1: Elapsed time for spectral clustering and miscluster rate for various values of k, s, p and q.

4 Remarks

There are a few observations worth noticing in figure 2. First, comparing the upper-left plot and the lower-right plot, `eigs` (Step 2) uses less iterations as p grows, but gets slower at the same time. This is because the time complexity of `eigs`, which is $O(trnd)$, is getting close to $O(trn^2)$. Intuitively, since p determines the signal strength, a higher p results in faster convergence of IRAM, but slower implementation due to dense matrix operations. Next, by looking at the upper-right plot, we can see that even with multithreading in Step 1, the time for generating A (Step 1) is more severly affected by s (hence, n) than the `eigs` step, or k-means clustering step. The bottom-left plot is a manifestation of the Davis-Kahan $\sin\Theta$ theorem.

Approximate spectral clustering methods usually fall into one of two categories: approximating the affinity matrix generating step [6], or approximating the eigenvalue decomposition step using randomized algorithms [1]. Although `eigs` works very quickly in our setting, we have seen that decreasing the sparsity generally makes `eigs` slower. If we have to deal with dense affinity matrices, we can no longer rely on sparse matrix operations and storage, hence it may be necessary to use approximate spectral clustering algorithms.

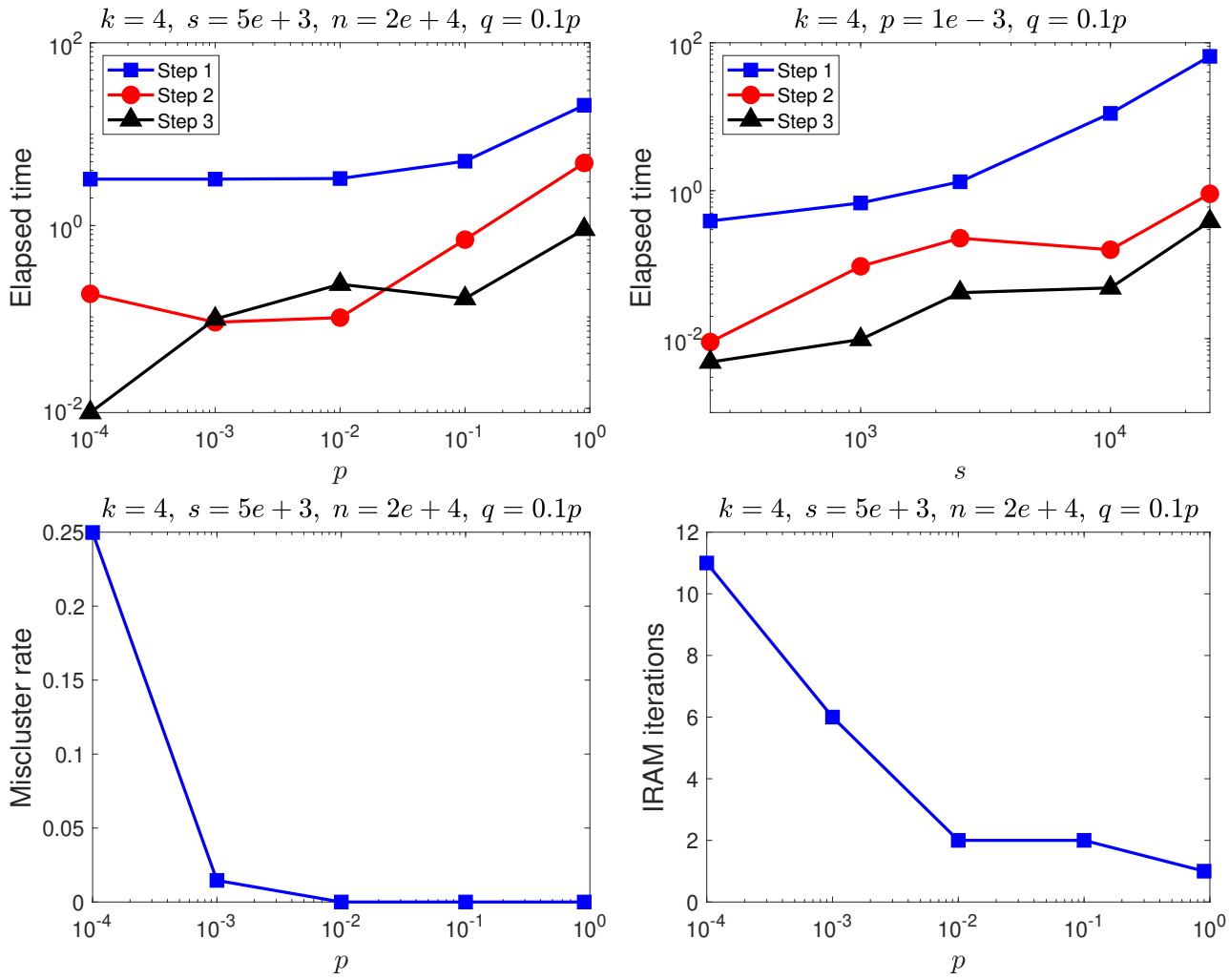

Figure 2: Elapsed time, miscluster rate and IRAM iterations plotted against various values of p and s.

References

[1] H. Jia et al. "An efficient Nyström spectral clustering algorithm using incomplete Cholesky decomposition". *Expert Systems with Applications* 186 (2021), p. 115813. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2021.115813. URL: https://www.sciencedirect.com/science/article/pii/S0957417421011817 (cited on p. 3).

[2] R. B. Lehoucq, D. C. Sorensen, and C. Yang. *ARPACK Users' Guide*. Society for Industrial and Applied Mathematics, 1998. URL: https://epubs.siam.org/doi/abs/10.1137/1.9780898719628 (cited on p. 2).

[3] V. Lyzinski et al. "Perfect clustering for stochastic blockmodel graphs via adjacency spectral embedding". *Electronic Journal of Statistics* 8.2 (2014). Publisher: Institute of Mathematical Statistics and Bernoulli Society, pp. 2905–2922. DOI: 10.1214/14-EJS978. URL: https://doi.org/10.1214/14-EJS978 (cited on p. 1).

[4] K. Rohe, S. Chatterjee, and B. Yu. "Spectral clustering and the high-dimensional stochastic blockmodel". *The Annals of Statistics* 39.4 (2011). Publisher: Institute of Mathematical Statistics, pp. 1878–1915. DOI: 10.1214/11-AOS887. URL: https://doi.org/10.1214/11-AOS887 (cited on p. 1).

[5] D. C. Sorensen. "Implicitly Restarted Arnoldi/Lanczos Methods for Large Scale Eigenvalue Calculations". *Parallel Numerical Algorithms*. Ed. by D. E. Keyes, A. Sameh, and V. Venkatakrishnan. Dordrecht: Springer Netherlands, 1997, pp. 119–165. ISBN: 978-94-011-5412-3. DOI: 10.1007/978-94-011-5412-3_5. URL: https://doi.org/10.1007/978-94-011-5412-3_5 (cited on pp. 2, 3).

[6] D. Yan, L. Huang, and M. I. Jordan. "Fast Approximate Spectral Clustering". *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '09. event-place: Paris, France. New York, NY, USA: Association for Computing Machinery, 2009, pp. 907–916. ISBN: 978-1-60558-495-9. DOI: 10.1145/1557019.1557118. URL: https://doi.org/10.1145/1557019.1557118 (cited on p. 3).

[7] Y. Yu, T. Wang, and R. J. Samworth. "A useful variant of the Davis–Kahan theorem for statisticians". *Biometrika* 102.2 (Apr. 2014), pp. 315–323. ISSN: 0006-3444. DOI: 10.1093/biomet/asv008. URL: https://doi.org/10.1093/biomet/asv008 (cited on p. 1).

[8] A. Y. Zhang. *Fundamental Limits of Spectral Clustering in Stochastic Block Models*. eprint: 2301.09289. 2023 (cited on p. 1).