# Performance test for the approximate algorithm for calculating cut distance

## 1 Test model (Erdős–Rényi model)

We use the Erdős–Rényi random graph model, which is a classic model of random graphs introduced by Pál Erdős and Alfréd Rényi in the late 1950s as a benchmark to test the performance of the approximate algorithm for calculating cut distance.

The Erdős–Rényi random graph, often denoted as $G(n, \mathcal{P})$, is defined by two parameters: $n$, the number of nodes, and $\mathcal{P} \in [0, 1]$, the probability that any given pair of nodes is connected by an edge. The probability $\mathfrak{P}(G)$ of generating a specific graph $G$ with $m$ edges from $n$ nodes in the $G(n, \mathcal{P})$ model is given by:

$$\mathfrak{P}(G) = \mathcal{P}^m (1 - \mathcal{P})^{\binom{n}{2} - m}$$

where $\binom{n}{2}$ is the binomial coefficient representing the total number of possible edges between $n$ nodes, and $m$ is the actual number of edges in the graph $G$. This is because the density of the network is often measured by the average degree (the average number of connections per node). In the $G(n, \mathcal{P})$ model, the average degree $\langle k \rangle$ is given by $\langle k \rangle = \mathcal{P}(n - 1)$. Thus, the larger $\mathcal{P}$ is, the higher the average degree, and the denser the graph.

The reason why we use the Erdős–Rényi model as the benchmark is that by varying $\mathcal{P}$ within the interval $[0, 1]$, we can examine completely random graphs with different densities controlled by $\mathcal{P}$. This approach allows the generation of random graphs ranging from very sparse to very dense, making it a robust and general test bed for various algorithms, especially the approximate algorithm for calculating cut distance between two graphs.

## 2 Cut distance on Erdős–Rényi graphs

In fact, the Erdős–Rényi random graph $G(n, \mathcal{P})$ we generate each time is different due to its probabilistic nature, when we calculate $\mathrm{d}_\square(G(n, \mathcal{P}_1), G(n, \mathcal{P}_2))$, we are actually computing its expect $\mathbb{E}(\mathrm{d}_\square(G(n, \mathcal{P}_1), G(n, \mathcal{P}_2)))$, which provides an estimate of the expected distance between the two graph distributions $G(n, \mathcal{P}_1)$ and $G(n, \mathcal{P}_2)$.

The cut distance between two graphs $G_1$ and $G_2$ with the same set of vertices can be written as:

$$\mathrm{d}_\square(G, H) = \min_\sigma \frac{2}{n(n-1)} \sum_{i<j} |A_G(i, j) - A_H(\sigma(i), \sigma(j))|$$

Here, $\sigma$ ranges over all permutations of vertices, and $A_G$ and $A_H$ are the adjacency matrices of graphs $G$ and $H$, respectively.

Consider two Erdős–Rényi random graphs $G_1 = G(n, \mathcal{P}_1)$ and $G_2 = G(n, \mathcal{P}_2)$, with adjacency matrices $A_{G_1}$ and $A_{G_2}$, respectively. Then, for each pair of vertices $(i, j)$ where $i < j$, we have:

$$\mathbb{E}[|A_{G_1}(i,j) - A_{G_2}(i,j)|]$$

Since $A_{G_1}(i,j)$ and $A_{G_2}(i,j)$ are independent Bernoulli random variables with $\mathcal{P}_1$ and $\mathcal{P}_2$, respectively:

$$A_{G_1}(i,j) \sim \text{Bernoulli}(\mathcal{P}_1), \quad A_{G_2}(i,j) \sim \text{Bernoulli}(\mathcal{P}_2)$$

To calculate the expected absolute difference of three situations:

- When $A_{G_1}(i,j) = 1$ and $A_{G_2}(i,j) = 0$, the probability is $\mathcal{P}_1(1 - \mathcal{P}_2)$, and the absolute value is 1.
- When $A_{G_1}(i,j) = 0$ and $A_{G_2}(i,j) = 1$, the probability is $(1 - \mathcal{P}_1)\mathcal{P}_2$, and the absolute value is 1.
- When $A_{G_1}(i,j) = A_{G_2}(i,j)$, the probability is $\mathcal{P}_1\mathcal{P}_2 + (1-\mathcal{P}_1)(1-\mathcal{P}_2)$, and the absolute value is 0.

Therefore, the expected absolute difference is:

$$\begin{aligned}\mathbb{E}[|A_{G_1}(i,j) - A_{G_2}(i,j)|] &= |\mathcal{P}_1(1 - \mathcal{P}_2) - (1 - \mathcal{P}_1)\mathcal{P}_2| \\ &= |\mathcal{P}_1 - \mathcal{P}_2|\end{aligned}$$

Considering all pairs of vertices in the upper triangular matrix, which is $\binom{n}{2} = \frac{n(n-1)}{2}$ pairs, the total expected absolute difference is:

$$\begin{aligned}\mathbb{E}\left[\sum_{i<j} |A_{G_1}(i,j) - A_{G_2}(i,j)|\right] &= \sum_{i<j} \mathbb{E}[|A_{G_1}(i,j) - A_{G_2}(i,j)|] \\ &= \binom{n}{2}|\mathcal{P}_1 - \mathcal{P}_2| \\ &= \frac{n(n-1)}{2}|\mathcal{P}_1 - \mathcal{P}_2|\end{aligned}$$

To obtain the cut distance, we need to normalize this total expected difference by $\binom{n}{2}$:

$$\begin{aligned}\mathbb{E}[\mathrm{d}_\square(G_1, G_2)] &= \frac{2}{n(n-1)} \cdot \mathbb{E}\left[\sum_{i<j} |A_{G_1}(i,j) - A_{G_2}(i,j)|\right] \\ &= \frac{2}{n(n-1)} \cdot \frac{n(n-1)}{2} \cdot |\mathcal{P}_1 - \mathcal{P}_2| \\ &= |\mathcal{P}_1 - \mathcal{P}_2|\end{aligned}$$

In the aspect of numerical computation, we need to generate multiple instances of the graphs, compute the distances between each pair of generated graphs, and then take the average of these distances.

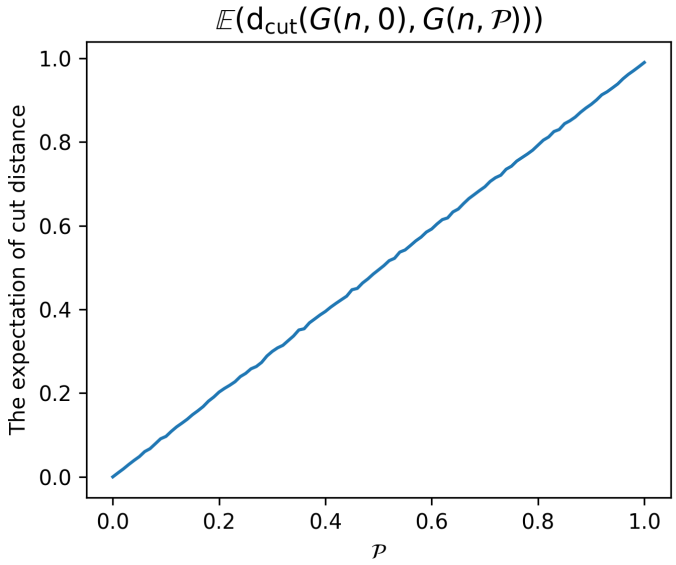


Figure 1: $\mathbb{E}[\mathrm{d}_{\square}(G(n,0),G(n,\mathcal{P}))]$, $\mathcal{P}\in[0,1]$, setting $n=10$ and taking 10-times' average.

Here, we take the average for 10 times, set the number of nodes as $n=100$, and use the method corresonding to our code, i.e, the SDP relaxation for optimization on Stiefel manifold by using the Cayley transformation.

Fig. 1 is taking $G(n,0)$ as the reference and getting the result of $\mathbb{E}[\mathrm{d}_{\square}(G(n,0),G(n,\mathcal{P}))]$, where $\mathcal{P}\in[0,1]$. Fig. 2 is the heatmap for presenting $\mathbb{E}[\mathrm{d}_{\square}(G(n,\mathcal{P}_1),G(n,\mathcal{P}_2))]$ for each pair of $\mathcal{P}_1,\mathcal{P}_2\in[0,1]$. The numerical computation has verified the results of the theoretical derivation, i.e., $\mathbb{E}[\mathrm{d}_{\square}(G_1,G_2)]=|\mathcal{P}_1-\mathcal{P}_2|$, which means the expected distance is proportional to the absolute difference in the probability of existing an edge between any two nodes.

# 3 Performance test

In theory, the time complexity of constraint-preserving update scheme is $8np^2+O(p^3)$, where $n$ is the dimension of the matrix $A\in\mathbb{R}^{n\times n}$. By selecting an appropriate $p=\max\left(\min\left(\mathrm{round}\left(\sqrt{2n}/2\right),100\right),1\right)$, the computational complexity can be reduced to $4n^2+O(n^{3/2})$. Here, $p$ refers the dimension of the identity matrix $I=X^TX$ used to maintain the orthogonality constraint during the $|A|_{\infty\to1}$ SDP relaxation, where $X\in\mathbb{R}^{n\times p}$. The objective is $\max_X \mathrm{Tr}(AX)$, and the feasible domain is the Stiefel manifold $M_{np}=\{X\in\mathbb{R}^{n\times p}:X^TX=I\}$. After completing the optimization, we can get $|A|_{\square}\approx\frac{|A|_{\infty\to1}}{K_G}$, where $K_G$ is the Grothendieck constant. Because the method utilizes gradient methods on Stiefel manifolds combined with line search, achieving linear convergence to the optimum, the algorithm will converge in $O(\epsilon^{-1})$ times of iterations, where $\epsilon$ is the tolerance. Therefore, the overall computational cost for cut distance is $O(n/\epsilon)$, which simplifies to $O(n^2)$ with a constant tolerance $\epsilon$.

In experiment, we traversed $\mathcal{P}\in[0,1]$ and took 10 averages for $\mathbb{E}[\mathrm{d}_{\square}(G(n,0),G(n,\mathcal{P}))]$, and obtained the average time to compute the cut distance once. We obtained the average time spent to compute the cut distance between two graphs with $n$ nodes by setting $n=0,200,\cdots,4000$, as shown in Fig. 3. The actual measured time consumption and the theoretically derived time consumption, aligned with the actual measurement for $n=4000$, exhibit the same growth trend with $n$, namely $O(n^2)$.

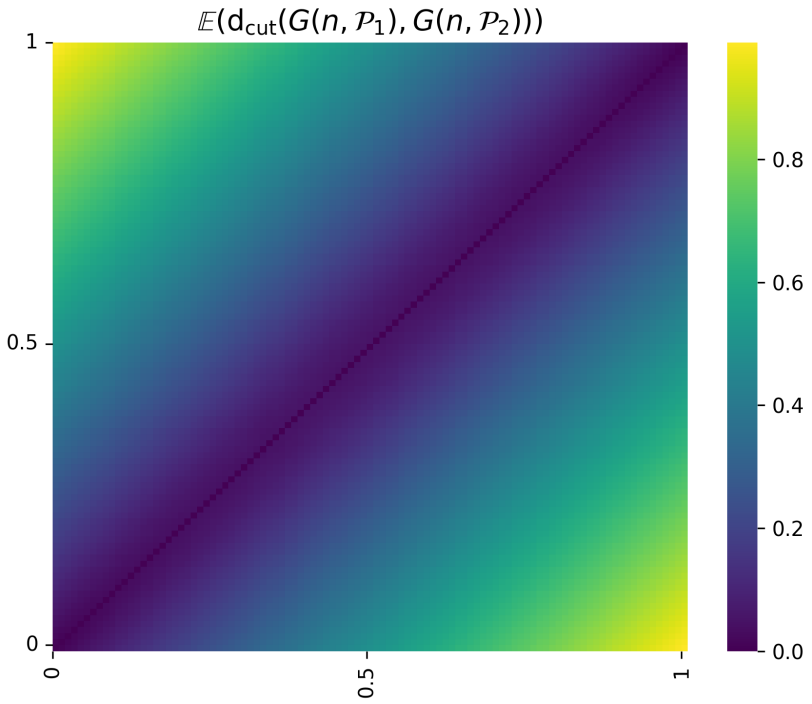


Figure 2: $\mathbb{E}[\mathrm{d}_{\square}(G(n,\mathcal{P}_1),G(n,\mathcal{P}_2))]$, $\mathcal{P}_1,\mathcal{P}_2 \in [0,1]$, setting $n = 10$ and taking 10-times' average. The x-axis represents $\mathcal{P}_1$, the y-axis represents $\mathcal{P}_2$ and the colorbar on the right of the heatmap represents the expected distance.

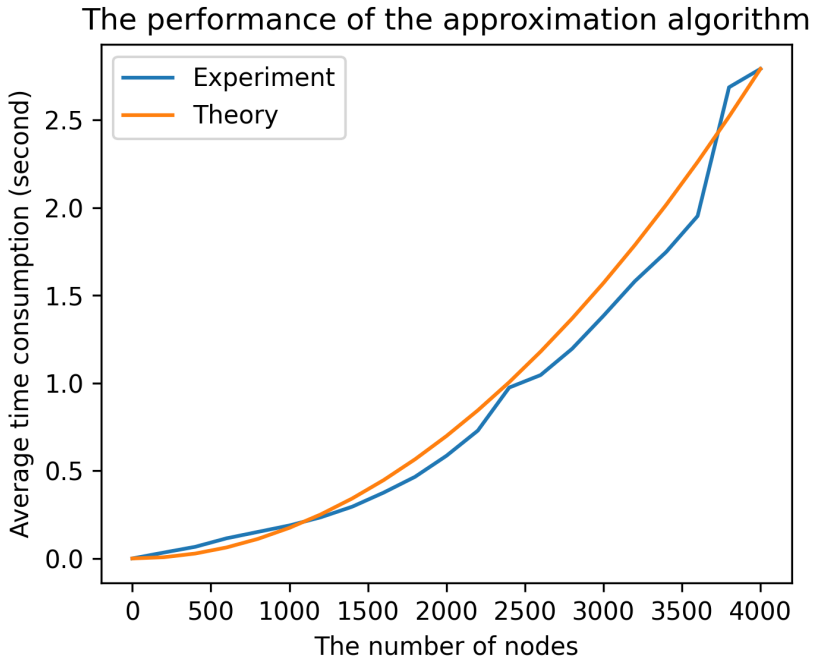


Figure 3: The average time consumption for calculating the cut distance using the approximation algorithm. The blue line is the experimental measurement result on a PC (CPU: Intel® Xeon® Gold 6130 CPU @ 2.10GHz ×32, Memory: 256GB). The orange line is the theoretical time consumption derived from the computational complexity of $O(n^2)$ is aligned with the actual time consumption for $n = 4000$.