Algorithm Implementation:

For a matrix $A \in \mathbb{R}^{N\times N}$, its $(\infty \to 1)$-norm is defined as:

$$|A|_{\infty\to 1} = \max_{X\in M_{N,p}} \langle A, X\rangle_F,$$

where $M_{N,p} = \{X \in \mathbb{R}^{N\times p} : X^T X = I\}$ is called Stiefel manifold, and the following is the Frobenius inner product:

$$\langle A, X\rangle_F = \sum_{i,j} A_{ij} X_{ij}, \quad X \in \mathbb{R}^{N\times p}.$$

Note that when $p = 1$ one recovers the usual $\ell_\infty \to \ell_1$ operator norm:

$$\|A\|_{\infty\to 1} = \max_{\|x\|_\infty \le 1} \|Ax\|_1 = \max_{\|x\|_\infty = 1,\ \|y\|_\infty = 1} x^T A y,$$

where $x, y \in \mathbb{R}^N$ are the original symbolic $\{\pm 1\}$ vectors. To obtain the orthogonal relaxation, we replace each scalar sign variable $x_i, y_j \in \{\pm 1\}$ by a unit vector $u_i, v_j \in \mathbb{R}^d$ with $\|u_i\|_2 = \|v_j\|_2 = 1$, and then stack $\{u_i\}$ into $U \in \mathbb{R}^{d\times N}$, and stack $\{v_i\}$ into $V \in \mathbb{R}^{d\times N}$.

This transforms the discrete optimization over $\{\pm 1\}$ into a continuous problem on the Stiefel manifold $M_{N,p}$.

The Grothendieck inequality offers a upper bound of the cut norm of a matrix, converting the calculation into a so-called $(\infty \to 1)$- norm. In detail, for a matrix $A \in \mathbb{R}^{N\times N}$:

$$|A|_\square \le |A|_{\infty\to 1} \le K_G |A|_\square,$$

where K_G is the Grothendieck constant.

The operation for finding a trace can be written as the form of a matrix multiplication by the inner product of two vectors, as follow:

$$\langle A, X\rangle_F = \sum_{i=1}^{N}\sum_{j=1}^{p} A_{ij}\, X_{ij}.$$

Of course, when $p = 1$ (so $X = xy^T$, $\|x\|_\infty = \|y\|_\infty = 1$), we have:

$$\langle A,\ X\rangle_F = \sum_{i,j} A_{ij}x_i y_j = x^T A y.$$

In fact, the Grothendieck constant K_G is a function of the d, that is, $K_G(d)$, where d is the dimension of the real Hilbert space spanned by the vectors $\mathbf{x}_i$ and $\mathbf{y}_j$. Grothendieck proved that $K_G(d)$ is increasing to d and bounded, so that the limit of $K_G(d)$ exists, as well as gave an effective estimate for $K_G(d)$.

For the estimation of K_G, we have:

$$1.57 \approx \frac{\pi}{2} \leq K_G = \sup_d K_G(d) \leq \sinh\frac{\pi}{2} \approx 2.3.$$

In practical calculations, we take $K_G = 1.782$.

Therefore, calculating the cut norm of a matrix is essentially calculating its $\infty \to 1$- norm. N. Alon and A. Naor proposed a method that convert this optimization into a problem of semidefinite programming (SDP). For the SDP solving, Z. Wen and W. Yin designed a solver for linear searching on Stiefel manifold by using the Cayley transformation to do this kind of $|A|_{\infty\to 1}$ SDP relaxation.

Therefore, the relaxation of the cut norm of a matrix can be formulated as the following optimization:

1. The objective function is:

$$\max_X \langle A,\ X\rangle_F.$$

2. The feasible domain is:

$$M_{N,p} = \{X \in \mathbb{R}^{N\times p} \mid X^T X = I\}.$$

After the optimization is completed, we have:

$$|A|_\square \approx \frac{|A|_{\infty\to 1}}{K_G}.$$

However, in the realistic implementation, in order to express the objective function in the form of "taking the trace of a square matrix", while simultaneously obtaining an exact constant in the analysis, we need to extend A and X as follows:

1. **Extending A to $\mathfrak{A}$:**

$$\mathfrak{A} = \begin{pmatrix} A & -A\mathbf{1} \\ -\mathbf{1}^T A & \mathbf{1}^T A \mathbf{1} \end{pmatrix} \in \mathbb{R}^{(N+1)\times(N+1)},$$

where $\mathbf{1}$ denotes the column vector of all ones. It has been proven by N. Alon and A. Naor that $\mathfrak{A}$ has the same cut norm as A, and the sum of the entries in each row and each column is zero.

2. **Extending X to $\mathfrak{X}$:**
a. Let $\mathscr{X} \in \mathbb{R}^{p\times \mathfrak{n}}$ be a matrix to be initialized, where $\mathfrak{n} = 2(N+1)$ and $\mathscr{X}^T\mathscr{X} = I_{\mathfrak{n}}$. The initialization method for $\mathscr{X}$ will be introduced later. Then, $\mathscr{X}$ is partitioned into horizontal blocks as follows:

$$\mathscr{X} = \left[U \mid V\right], \quad \text{where } U, V \in \mathbb{R}^{p\times(N+1)}.$$

b. In the subsequent optimization process, the Cayley update and tangent space projection are first applied to $\mathscr{X}$. When computing the objective function and gradient, we take:

$$\mathfrak{X} = U^T V \in \mathbb{R}^{(N+1)\times(N+1)}.$$

After extending the matrix, the optimization process is transformed into:

1. The objective function is:

$$\max_{\mathscr{X}^T\mathscr{X}=I_{\mathfrak{n}}} \operatorname{Tr}(\mathfrak{A}^T\mathfrak{X}).$$

2. The feasible domain is:

$$M_{p,\mathfrak{n}} = \left\{\mathscr{X} \in \mathbb{R}^{p\times\mathfrak{n}} \mid \mathscr{X}^T\mathscr{X} = I_{\mathfrak{n}}\right\}.$$

The optimization algorithms contains two steps:

1. Initialization:

The initialization is intended to provide a starting point for subsequent iterative updates. To start with, a random matrix $Z \in \mathbb{R}^{\mathfrak{n}\times p}$ is generated. Then, the QR decomposition is performed on Z, resulting in $Z = QR$, where Q is an orthogonal matrix. The initial value of $\mathscr{X}$ is set from Q, i.e., $\mathscr{X}_0 = Q$. Note that this procedure corresponds precisely to relaxing

each original scalar variable $x_i \in \{\pm 1\}$ (and similarly y_j) into a continuous vector $u_i \in \mathbb{R}^d$ of unit norm, so that the combinatorial problem on $\{\pm 1\}^N$ becomes optimization over the Stiefel manifold $M_{p,\mathrm{n}}$ via the matrix $\mathscr{X}$.

For convenience, we denote $\mathrm{Tr}(\mathfrak{A}^T \mathscr{X})$ as $f(\mathscr{X})$, considering $\nabla f(\mathscr{X}) = \mathfrak{A}$ and $\mathfrak{A} = \mathfrak{A}^T$, then the gradient at initial state can be calculated as follow (tangent space projection):

$$G = \mathfrak{A} - \mathscr{X} \frac{\mathscr{X}^T \mathfrak{A} + \mathfrak{A}^T \mathscr{X}}{2}.$$

This equation offers the direction for gradient update at the start of the algorithm, where the term after the minus sign ensures that the optimization process adheres to the constraints of the Stiefel manifold $M_{p,\mathrm{n}}$.

In fact, since the matrix $\mathfrak{A}$ satisfies $\mathfrak{A} = \mathfrak{A}^T$, this equation can also be written as $G = \mathfrak{A} + \mathfrak{A}^T - \mathscr{X}(\mathscr{X}^T \mathfrak{A} + \mathfrak{A}^T \mathscr{X}) = 2\left[\mathfrak{A} - \mathscr{X}(\mathscr{X}^T \mathfrak{A} + \mathfrak{A}^T \mathscr{X})/2\right]$. It is both correct to write the gradient G either with or without the factor of 2. The difference simply reflects whether the factor of 2 is absorbed into the gradient itself or into the following Cayley transformation, and this does not affect the correctness or convergence of the optimization process.

In order to make sure the matrix $\mathscr{X}$ always satisfies $\mathscr{X}^T \mathscr{X} = I$, the Cayley transformation is used to implement the process of assignment. Let $G = \mathfrak{A} - \mathscr{X}(\mathscr{X}^T \mathfrak{A} + \mathfrak{A}^T \mathscr{X})/2$ as with above, and $W = G\mathscr{X}^T - \mathscr{X} G^T$, so that $W^T = -W$ and W is skew-symmetric. Then the update is as follows:

$$C(\tau) := \left(I - \frac{\tau}{2} W\right)^{-1} \left(I + \frac{\tau}{2} W\right), \quad Y(\tau) = C(\tau)\mathscr{X}.$$

Through this process, a new trial matrix $Y(\tau)$ can be generated, which is then used to decide whether to replace $\mathscr{X}$ with $Y(\tau)$ according to the following Armijo-Wolfe condition:

$$\begin{cases} f(\mathscr{X} + \tau \Delta \mathscr{X}) \leq f(\mathscr{X}) + c_1 \tau \nabla f(\mathscr{X})^T \Delta \mathscr{X}, \\ \nabla f(\mathscr{X} + \tau \Delta \mathscr{X})^T \Delta \mathscr{X} \geq c_2 \nabla f(\mathscr{X})^T \Delta \mathscr{X}, \end{cases}$$

where $c_1 \in [0, 0.5]$ and $c_2 \in [c_1, 1]$.

Specifically, the first inequality, known as the Armijo condition, ensures that each update achieves a sufficient decrease in the gradient. The second inequality, called the Wolfe condition, keeps the step length aligned with a reasonable gradient descent direction.

If this condition is satisfied, $Y(\tau)$ will be used to update $\mathscr{X}$. Otherwise, $\mathscr{X}$ will remain unchanged, and the value of τ will be adjusted as follows:

$$\tau = \frac{s^T y}{y^T y} \quad \text{or} \quad \tau = \frac{s^T s}{s^T y},$$

where $s = \mathscr{X}_k - \mathscr{X}_{k-1}$ and $y = \nabla f(\mathscr{X}_k) - \nabla f(\mathscr{X}_{k-1})$. Note that, the former of these two formulas tends to keep the step length stable, while the latter tends to speed up the update process.

Then, $Y(\tau)$ needs to be recalculated according to above assignment, and a suitable τ should be found to meet the Armijo-Wolfe condition. When the first assignment from $Y(\tau)$ to $\mathscr{X}$ occurs, the initialization is complete. At this point, the algorithm will enter an iterative loop.

2. Iteration:
Each step of this iteration generates a new trial matrix $Y(\tau)$ based on the current $\mathscr{X}$ and uses the Armijo-Wolfe condition to evaluate whether to use $Y(\tau)$ to update $\mathscr{X}$:

1. Calculating the new direction for gradient update:

$$W = \nabla f(\mathscr{X}) - \mathscr{X} \nabla f(\mathscr{X})^T;$$

2. Generating the new trial matrix:

$$Y(\tau) = (I - \frac{\tau}{2} W)^{-1} (I + \frac{\tau}{2} W) \mathscr{X};$$

3. Assigning $Y(\tau)$ to $\mathscr{X}$ if the Armijo-Wolfe condition meets; otherwise, adjusting τ and recalculating $Y(\tau)$ until the Armijo-Wolfe condition is satisfied.

The iteration ends when the redefined maximum number of steps is reached, or the gradient update $|\nabla f(\mathscr{X})|$ is less than the predefined threshold (tolerance ε).

The time complexity of the constraint-preserving update scheme is $8Np^2 + O(p^3)$. By selecting p as the column number of unextended X, by:

$$p = \max\left(\min\left(\text{round}\left(\sqrt{2N}/2\right), 100\right), 1\right),$$

the computational complexity can be reduced to $4N^2 + O(N^{3/2})$.

Here, p refers to the dimension of the identity matrix $I = X^T X$ used to maintain the

orthogonality constraint during the original $|A|_{\infty\to 1}$ SDP relaxation before the extension, where the unextended $X \in \mathbb{R}^{N\times p}$ [Wen et al., 2013]. Because the method utilizes gradient methods on Stiefel manifolds combined with line search, achieving linear convergence to the optimum, the algorithm will converge in $O(\varepsilon^{-1})$ iterations, where ε is the tolerance [Wen et al., 2013]. Therefore, the overall computational cost for cut distance is $O(N/\varepsilon)$, which simplifies to $O(N^2)$ with a constant tolerance ε [Z. Wen and W. Yin, 2013].

In the Grothendieck inequality $|A|_\square \le |A|_{\infty\to 1} \le K_G|A|_\square$, the Grothendieck constant K_G can be amplified to a safe upper bound of 4, resulting in a new inequation:

$$|\mathfrak{A}|_\square \le |\mathfrak{A}|_{\infty\to 1} \le 4|\mathfrak{A}|_\square.$$

As mentioned earlier, the row and column sums of $\mathfrak{A}$ are zero. N. Alon and A. Naor proved that, in this case, the equality on the right-hand side can be attained. Therefore, one can consider $|A|_\square \sim |A|_{\infty\to 1}/K_G \sim |\mathfrak{A}|_{\infty\to 1}/4$ and take $|\mathfrak{A}|_\square = |\mathfrak{A}|_{\infty\to 1}/4$ directly.

After completing the process of optimization, one can have:

$$|A|_\square \approx \frac{|\mathfrak{A}|_{\infty\to 1}}{4}.$$

Especially, for node-labelled graphs, one can use:

If G and H are node-labelled graphs on the same vertex set, and their adjacency matrices M^G and M^H are arranged so that the order of rows and columns corresponds to the same ordering of the vertices, one can directly take the permutation in the formula of cut distance (See Section 8.2.2 of Large networks and graph limits) as the identity operator and have:

$$\mathrm{d}_\square(G,H) = |M^G - M^H|_\square.$$

It should be noted that its validity is discussed in detail in Section 8.1.2 of Large networks and graph limits, and this is also the method we use in the PRB paper.

We may also need to emphasize that "node-labelled graphs" above refer to the case where all nodes are labeled in the "partially labeled graphs" defined in Section 3.2 of Large networks and graph limits, which are also called "flat" or "fully labeled" graphs in that book. They are referred to by different names in various references, but all mean the same thing, as long as one does not confuse them.

Additional Notes:

In the code implementation, we used the trick of extending A and X to square matrices $\mathfrak{A}$ and $\mathfrak{X}$ in $\mathbb{R}^{(N+1)\times(N+1)}$, respectively, and expressing the objective function as taking the strict trace of a square matrix. We stress that it is purely a technical convenience for programming and computation, and is not central to the theoretical foundation (Alon and Wen). Except for this repo, this detail is also reflected in the original code, which was archived on Zenodo at the time of publication and used for all computations reported in our PRB paper.

Furthermore, we present the Frobenius inner product $\langle A^T, X\rangle_F$ as the trace taking form of $\mathrm{Tr}(AX)$ in our PRB paper, and it is standard terminology of the optimization field. Another observation is that the task of Eq.(3) in our PRB paper is to perform a trial update along the original gradient A prior to the formation of $W_0 = GX^{\mathsf{T}} - XG^{\mathsf{T}}$ expressly. Here A can also be replaced by W_0 without performing anything to the subsequent iterations or the eventual output of $|A|_{\infty\to 1}$. Moreover, both $(I + \frac{\tau}{2}W)^{-1}(I - \frac{\tau}{2}W)$ and $(I - \frac{\tau}{2}W)^{-1}(I + \frac{\tau}{2}W)$ are correct expressions for the Cayley transform in Eq.(3) and Eq.(6), and the only difference between them is the sign of τ. As long as $W^{\mathsf{T}} = -W$ holds, each update will stay on the Stiefel manifold, thereby ensuring both the correctness and convergence of the iteration.

Finally, for the gradient, the notation used in our PRB paper is:

$$G = A + A^T - X(X^TA + A^TX),$$

whereas the notation used in this document (Algorithm.pdf) is:

$$G = \mathfrak{A} - \mathscr{X}\frac{\mathscr{X}^T\mathfrak{A} + \mathfrak{A}^T\mathscr{X}}{2}.$$

In fact, these two notations are entirely equivalent.

The reason is that for the extended square matrix $\mathfrak{A} \in \mathbb{R}^{(N+1)\times(N+1)}$, $-A\mathbf{1}$ in the upper triangular part and $-\mathbf{1}^TA$ in the lower triangular part are transposes of each other. As a result, $\mathfrak{A}$ is a symmetric matrix, that is, $\mathfrak{A} = \mathfrak{A}^T$, so $\mathfrak{A} + \mathfrak{A}^T = 2\mathfrak{A}$. Therefore, after substituting A with $\mathfrak{A}$, the gradient in the PRB paper should naturally be converted to:

$$G = \mathfrak{A} + \mathfrak{A}^T - \mathscr{X}(\mathscr{X}^T\mathfrak{A} + \mathfrak{A}^T\mathscr{X}) = 2\cdot\left[\mathfrak{A} - \mathscr{X}\frac{\mathscr{X}^T\mathfrak{A} + \mathfrak{A}^T\mathscr{X}}{2}\right].$$

What we have previously written in this document (Algorithm.pdf) is a version that omits the multiplication by 2. In fact, whether the gradient expression in G is multiplied by 2 merely determines whether this coefficient appears in the gradient or within the Cayley transform. Obviously, this distinction does not affect the correctness or convergence of the iterative process in any way.