Weighted Graph Cuts without Eigenvectors: A Multilevel Approach (PAMI 2007)

User-Guided Large Attributed Graph Clustering with Multiple Sparse Annotations (PAKDD 2016)
Weighted Graph Cuts without Eigenvectors: A Multilevel Approach

Inderjit S. Dhillon, Member, IEEE, Yuqiang Guan, and Brian Kulis

- I.S. Dhillon and B. Kulis are with the Department of Computer Sciences, University of Texas at Austin, 1 University Station C0500, Austin, TX 78712-0500. E-mail: {inderjit, kulis}@cs.utexas.edu.
- Y. Guan is with Microsoft Corporation, 1 Microsoft Way, Redmond, WA 98052. E-mail: yuqiang-guan@microsoft.com.

Manuscript received 27 Mar. 2006; revised 1 Nov. 2006; accepted 3 Jan. 2007; published online 14 Feb. 2007.

Recommended for acceptance by J. Buhmann.

For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEECS Log Number TPAMI-0244-0306. Digital Object Identifier no. 10.1109/TPAMI.2007.1115.
Problem Definition

Clustering nonlinearly separable data:

1. Kernel k-means
2. Spectral clustering

**Goal:** Design a fast graph clustering method
Computing eigenvectors is expensive in large graphs.
Given a set of vectors \( a_1, a_2, \ldots, a_n \) the k-means algorithm seeks to find clusters \( \pi_1, \pi_2, \ldots, \pi_k \) that minimize the objective function

\[
D(\{\pi_c\}_{c=1}^k) = \sum_{c=1}^k \sum_{a_i \in \pi_c} \| a_i - m_c \|^2, \text{ where } m_c = \frac{\sum_{a_i \in \pi_c} a_i}{|\pi_c|}.
\]

\( m_c \) is centroid or the mean of cluster \( \pi_c \).
To allow nonlinear separators we use kernel (mapping to higher dimension).

\[
\mathcal{D}(\{\pi_c\}_{c=1}^{k}) = \sum_{c=1}^{k} \sum_{a_i \in \pi_c} \| \phi(a_i) - m_c \|^2,
\]

where \( m_c = \frac{\sum_{a_i \in \pi_c} \phi(a_i)}{|\pi_c|}. \)

The squared distance \( \| \phi(a_i) - m_c \|^2 \) may be rewritten as

\[
\phi(a_i) \cdot \phi(a_i) - \frac{2 \sum_{a_j \in \pi_c} \phi(a_i) \cdot \phi(a_j)}{|\pi_c|} + \frac{\sum_{a_j, a_l \in \pi_c} \phi(a_j) \cdot \phi(a_l)}{|\pi_c|^2}.
\]

We just need kernel matrix \( K \), where \( K_{i,j} = \phi(a_i) \cdot \phi(a_j) \)
## Examples of Popular Kernel Functions

<table>
<thead>
<tr>
<th>Kernel Type</th>
<th>Kernel Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Polynomial Kernel</td>
<td>$\kappa(a_i, a_j) = (a_i \cdot a_j + c)^d$</td>
</tr>
<tr>
<td>Gaussian Kernel</td>
<td>$\kappa(a_i, a_j) = \exp(-|a_i - a_j|^2/2\alpha^2)$</td>
</tr>
<tr>
<td>Sigmoid Kernel</td>
<td>$\kappa(a_i, a_j) = \tanh(c(a_i \cdot a_j) + \theta)$</td>
</tr>
</tbody>
</table>
Weighted KERNEL k-MEANS

\[ D(\{\pi_c\}_{c=1}^k) = \sum_{c=1}^{k} \sum_{a_i \in \pi_c} w_i \| \phi(a_i) - m_c \|^2 , \]

where \( m_c = \frac{\sum_{a_i \in \pi_c} w_i \phi(a_i)}{\sum_{a_i \in \pi_c} w_i} \)

The weights \( w_i \) are non negative.

\( \| \phi(a_i) - m_c \|^2 \) can be written as

\[
\phi(a_i) \cdot \phi(a_i) - \frac{2 \sum_{a_j \in \pi_c} w_j \phi(a_i) \cdot \phi(a_j)}{\sum_{a_j \in \pi_c} w_j} \]
\[
+ \frac{\sum_{a_j, a_l \in \pi_c} w_j w_l \phi(a_j) \cdot \phi(a_l)}{(\sum_{a_j \in \pi_c} w_j)^2} .
\]

(1)

Using the kernel matrix \( K \), the above may be rewritten as

\[
K_{ii} - \frac{2 \sum_{a_j \in \pi_c} w_j K_{ij}}{\sum_{a_j \in \pi_c} w_j} + \frac{\sum_{a_j, a_l \in \pi_c} w_j w_l K_{jl}}{(\sum_{a_j \in \pi_c} w_j)^2} .
\]

(2)
KERNEL_KMEANS_BATCH($K$, $k$, $w$, $t_{max}$, $\{\pi_c^{(0)}\}_{c=1}^k$, $\{\pi_c\}_{c=1}^k$)

**Input:** $K$: kernel matrix, $k$: number of clusters, $w$: weights for each point, $t_{max}$: optional maximum number of iterations, $\{\pi_c^{(0)}\}_{c=1}^k$: optional initial clusters

**Output:** $\{\pi_c\}_{c=1}^k$: final clustering of the points

1. If no initial clustering is given, initialize the $k$ clusters $\pi_1^{(0)}, ..., \pi_k^{(0)}$ (e.g., randomly). Set $t = 0$.
2. For each point $a_i$ and every cluster $c$, compute

$$d(a_i, m_c) = K_{ii} - \frac{2 \sum_{a_j \in \pi_c^{(t)}} w_j K_{ij}}{\sum_{a_j \in \pi_c^{(t)}} w_j} + \frac{\sum_{a_j, a_l \in \pi_c^{(t)}} w_j w_l K_{jl}}{(\sum_{a_j \in \pi_c^{(t)}} w_j)^2}.$$  

3. Find $c^*(a_i) = \arg\min_c d(a_i, m_c)$, resolving ties arbitrarily. Compute the updated clusters as

$$\pi_c^{(t+1)} = \{a : c^*(a_i) = c\}.$$ 

4. Set $t = t + 1$. If not converged or $t < t_{max}$, go to Step 2; otherwise, stop and output final clusters $\{\pi_c^{(t)}\}_{c=1}^k$. 
The algorithm monotonically converges as long as $K$ is positive semi-definite.

Bottleneck is in step 2. Computing distance $d(a_i, m_c)$. $O(n)$ for every data point $\implies O(n^2)$ per iteration.

With sparse matrix $K \implies O(nz)$. Therefore time complexity is: $O(n^2(\tau + m))$. $m$: is original data dimension. $\tau$: number of iterations.
**Given** a graph $G = (V, E, A)$

**Partition** the graph into $k$ disjoint clusters $V_1, \ldots, V_k$ such that their union is $V$.

$\text{links}(A, B)$ is the sum of edge weights between nodes in $A$ and $B$.

$$\text{links}(A, B) = \sum_{i \in A, j \in B} A_{ij}.$$  

$\text{degree}(A) = \text{links}(A, V)$
Maximize within-cluster association relative to the size of the cluster.

\[ RAssoc(G) = \max_{\mathcal{V}_1, \ldots, \mathcal{V}_k} \sum_{c=1}^{k} \frac{\text{links}(\mathcal{V}_c, \mathcal{V}_c)}{|\mathcal{V}_c|} \]
Minimize the cut between clusters and the remaining vertices.

\[
RCut(G) = \min_{\mathcal{V}_1, \ldots, \mathcal{V}_k} \sum_{c=1}^{k} \frac{\text{links}(\mathcal{V}_c, \mathcal{V} \setminus \mathcal{V}_c)}{|\mathcal{V}_c|}
\]

Equal size partitions

\[
KLObj(G) = \min_{\mathcal{V}_1, \ldots, \mathcal{V}_k} \sum_{c=1}^{k} \frac{\text{links}(\mathcal{V}_c, \mathcal{V} \setminus \mathcal{V}_c)}{|\mathcal{V}_c|},
\]

subject to \(|\mathcal{V}_c| = |\mathcal{V}|/k \ \forall c = 1, \ldots, k.

Different objectives (Normalized cut)

\[ NCut(G) = \min_{\mathcal{V}_1, \ldots, \mathcal{V}_k} \sum_{c=1}^{k} \frac{\text{links}(\mathcal{V}_c, \mathcal{V} \setminus \mathcal{V}_c)}{\text{degree}(\mathcal{V}_c)}. \]

minimizing the normalized cut is equivalent to maximizing the normalized association, since
\[ \text{links}(\mathcal{V}_c, \mathcal{V} \setminus \mathcal{V}_c) = \text{degree}(\mathcal{V}_c) - \text{links}(\mathcal{V}_c, \mathcal{V}_c). \]
Different objectives (General weighted graph cuts/association)

We introduce a weight $w_i$ for each node of the graph, and for each cluster $V_c$, we define $w(V_c) = \sum_{i \in V_c} w_i$

$$W\text{ Assoc}(G) = \max_{\mathcal{V}_1, \ldots, \mathcal{V}_k} \sum_{c=1}^{k} \frac{\text{links}(\mathcal{V}_c, \mathcal{V}_c)}{w(V_c)}.$$  

Similarly, for cuts

$$W\text{ Cut}(G) = \min_{\mathcal{V}_1, \ldots, \mathcal{V}_k} \sum_{c=1}^{k} \frac{\text{links}(\mathcal{V}_c, \mathcal{V} \setminus \mathcal{V}_c)}{w(V_c)}.$$  

- Ration association: weights equal to one  
- normalized association: weight equal to degree
At first glance, the two approaches to clustering presented in the previous two sections appear to be unrelated.

kernel k-means objective as a trace maximization problem and weighted graph association problem are equivalent.
EQUIVALENCE OF THE OBJECTIVES

- **Weighted Kernel k-Means as Trace Maximization**

  \[
  \max_{\tilde{Y}} \text{trace} \left( \tilde{Y}^T W^{1/2} K W^{1/2} \tilde{Y} \right)
  \]

  where $\tilde{Y}$ is the orthonormal $n \times k$ matrix that is proportional to the square root of the weight matrix $W$.

- **Graph Clustering as Trace Maximization**

  \[
  WAssoc(G) = \max_{Y} \text{trace} \left( Y^T W^{-1/2} A W^{-1/2} Y \right)
  \]

  \[
  WCut(G) = \min_{Y} \text{trace} \left( Y^T W^{-1/2} L W^{-1/2} Y \right)
  \]
For weighted graph association, we define a matrix $K = W^{-1}AW^{-1}$ to map to weighted kernel k-means.

A is an arbitrary adjacency matrix, so $K$ is not necessarily positive definite.

Given $A$, define $K' = \sigma W^{-1} + W^{-1}AW^{-1}$

$$\text{trace}(\tilde{Y}^TW^{1/2}K'W^{1/2}\tilde{Y}) = \text{trace}(\tilde{Y}^TW^{1/2}\sigma W^{-1}W^{1/2}\tilde{Y})$$

$$+ \text{trace}(\tilde{Y}^TW^{-1/2}AW^{-1/2}\tilde{Y})$$

$$= \sigma k + \text{trace}(\tilde{Y}^TW^{-1/2}AW^{-1/2}\tilde{Y}).$$
Fig. 1. Overview of the multilevel algorithm (for \( k = 2 \)).
Coarsening Phase

Starting with the initial graph $G_0$, the coarsening phase repeatedly transforms the graph into smaller and smaller graphs $G_1; G_2; \ldots; G_m$ such that $|V_0| > |V_i| > \ldots > |V_m|$. One popular approach

- start with all nodes unmarked
- Visit each vertex in a random order.
- For each vertex $x$, if $x$ is not marked, merge $x$ with the unmarked vertex $y$ that corresponds to the highest edge weight among all edges between $x$ and unmarked vertices.
- Then, mark $x$ and $y$.
- If all neighbors of $x$ have been marked, mark $x$ and do not merge it with any vertex.
- Once all vertices are marked, the coarsening for this level is complete.
max-cut coarsening

Given a vertex $x$, instead of merging using the criterion of heavy edges, we instead look for the unmarked vertex $y$ that maximizes

$$\frac{e(x, y)}{w(x)} + \frac{e(x, y)}{w(y)},$$

where $e(x, y)$ corresponds to the edge weight between vertices $x$ and $y$, and $w(x)$ and $w(y)$ are the weights of vertices $x$ and $y$, respectively.
A parameter indicating how small we want the coarsest graph to be. For example, than 5k nodes, where k is the number of desired clusters.

- region-growing (no eigenvector computation)
- spectral clustering
- bisection method (no eigenvector computation)
The final phase of the algorithm is the refinement phase. **Given** a graph $G_i$, we form the graph $G_{i-1}$

- **initialization** If a supernode in $G_i$ is in cluster $c$, then all nodes in $G_{i-1}$ formed from that supernode are in cluster $c$.
- **improve** it using a refinement algorithm
- **(Optimized version)** Use only boundary nodes
A common problem when running standard batch kernel k-means is that the algorithm has a tendency to be trapped into qualitatively poor local minima. An effective technique to counter this issue is to do a local search by incorporating an incremental strategy. A step of incremental kernel k-means attempts to move a single point from one cluster to another in order to improve the objective function.
Gene Network Analysis

TABLE 3
Normalized Cut Values Returned by Graclus and the Spectral Method

<table>
<thead>
<tr>
<th># clusters</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>32</th>
<th>64</th>
<th>128</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graclus_All</td>
<td>0</td>
<td>.009</td>
<td>.12</td>
<td>.21</td>
<td>3.89</td>
<td>16.78</td>
</tr>
<tr>
<td>Graclus_Bdry</td>
<td>0</td>
<td>.009</td>
<td>.13</td>
<td>.52</td>
<td>4.44</td>
<td>17.65</td>
</tr>
<tr>
<td>Spectral</td>
<td>0</td>
<td>.037</td>
<td>.13</td>
<td>.92</td>
<td>5.37</td>
<td>25.46</td>
</tr>
<tr>
<td># clusters</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>16</td>
<td>32</td>
<td>64</td>
</tr>
<tr>
<td>------------</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>Graclus_All</td>
<td>.040</td>
<td>.155</td>
<td>.410</td>
<td>1.01</td>
<td>3.52</td>
<td>9.92</td>
</tr>
<tr>
<td>Graclus_Bdry</td>
<td>.050</td>
<td>.186</td>
<td>.467</td>
<td>1.17</td>
<td>3.80</td>
<td>10.09</td>
</tr>
<tr>
<td>Spectral</td>
<td>.048</td>
<td>.213</td>
<td>.832</td>
<td>4.14</td>
<td>16.87</td>
<td>-</td>
</tr>
</tbody>
</table>

Normalized cut values—lower cut values are better

**Computation time (in seconds)**

<table>
<thead>
<tr>
<th># clusters</th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>32</th>
<th>64</th>
<th>128</th>
<th>256</th>
<th>512</th>
<th>1024</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graclus_All</td>
<td>41.49</td>
<td>45.08</td>
<td>47.89</td>
<td>51.19</td>
<td>54.25</td>
<td>67.22</td>
<td>83.11</td>
<td>116.14</td>
<td>177.84</td>
<td>261.88</td>
</tr>
<tr>
<td>Graclus_Bdry</td>
<td>31.87</td>
<td>34.45</td>
<td>35.64</td>
<td>36.70</td>
<td>39.87</td>
<td>47.76</td>
<td>54.37</td>
<td>65.86</td>
<td>75.58</td>
<td>100.56</td>
</tr>
<tr>
<td>Spectral</td>
<td>163.92</td>
<td>301.39</td>
<td>518.60</td>
<td>1566.35</td>
<td>4410.10</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
### TABLE 5
**Ratio Association Values and Computation Time for a Varied Number of Clusters of the IMDB Movie Data Set, Using Two Variants of Our Multilevel Algorithm and the Spectral Method**

<table>
<thead>
<tr>
<th># clusters</th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>32</th>
<th>64</th>
<th>128</th>
<th>256</th>
<th>512</th>
<th>1024</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graclus.All</td>
<td>45.16</td>
<td>105.71</td>
<td>184.47</td>
<td>327.97</td>
<td>623.20</td>
<td>1076.16</td>
<td>1822.21</td>
<td>2971.87</td>
<td>4712.56</td>
<td>7253.46</td>
</tr>
<tr>
<td>Graclus.Bdry</td>
<td>12.18</td>
<td>24.39</td>
<td>50.49</td>
<td>98.31</td>
<td>189.84</td>
<td>352.14</td>
<td>628.04</td>
<td>1064.83</td>
<td>1802.05</td>
<td>3197.86</td>
</tr>
<tr>
<td>Spectral</td>
<td>10.13</td>
<td>19.30</td>
<td>52.77</td>
<td>94.30</td>
<td>172.36</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**Ratio association values—larger association values are better**

<table>
<thead>
<tr>
<th></th>
<th>Graclus.All</th>
<th>Graclus.Bdry</th>
<th>Spectral</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computation time (in seconds)</td>
<td>84.62</td>
<td>82.09</td>
<td>84.11</td>
</tr>
<tr>
<td></td>
<td>105.01</td>
<td>118.04</td>
<td>125.57</td>
</tr>
<tr>
<td></td>
<td>161.16</td>
<td>261.85</td>
<td>410.84</td>
</tr>
<tr>
<td></td>
<td>613.36</td>
<td>413.36</td>
<td>156.20</td>
</tr>
</tbody>
</table>


Fig. 4. Segmentation of a sample image. The leftmost plot is the original image and each of the three plots to the right of it is a component (cluster)—body, tail, and background. The normalized cut value for Graclus is 0.0221, smaller than 0.0239, the normalized cut value obtained using the spectral method.
User-Guided Large Attributed Graph Clustering with Multiple Sparse Annotations

Jianping Cao¹, Senzhang Wang²(✉), Fengcai Qiao¹, Hui Wang¹, Feiyue Wang¹, and Philip S. Yu³

¹ College of Information Systems and Management, National University of Defense Technology, Changsha, Hunan, China
² State Key Laboratory of Software Development Environment, Beihang University, Beijing, China
✉ swang@buaa.edu.cn
³ Department of Computer Science, University of Illinois at Chicago, Chicago, IL, USA
One of the key challenges in large attributed graph clustering is how to select representative attributes.

A single user may only pick out the samples that s/he is familiar with while ignoring the others, such that the selected samples are often biased.

Allows multiple individuals to select samples for a specific clustering...
Given a large attributed graph $G(V, E, F)$ with $|V| = n$ nodes and $|E| = m$ edges, where each node is associated with $|F| = d$ attributes,

we target to extract cluster $C$ from $G$ with the guidance of $K$ users. Each user independently labels the samples based on his/her own knowledge. The samples annotated by the $k$-th user are denoted as $U^k$. For each set $U^k$, we assume that nodes inside it are similar to each other, and they are dissimilar to the nodes outside the set.
Method CGMA

- combine the annotations first in an unbiased way to obtain the guidance information
- Then, use a local clustering method to cluster the graph with the guidance of combined annotations.
Since the annotations are sparse labels with little overlaps, straightforward methods like majority voting may not effectively capture the relations among the annotations.

\[
\min_{A^k} \sum_{(i,j) \in P_C^k} (f_i - f_j)^T A^k (f_i - f_j) - \gamma \log \left( \sum_{(i,j) \in P_D^k} \sqrt{(f_i - f_j)^T A^k (f_i - f_j)} \right)
\]

Here, \( P_C^k \) and \( P_D^k \) denote the similar and dissimilar set of the k-th annotation.

\[
\beta^k = \text{diag}(A^k)(k \in 1, \cdots, K)
\]

\[
\rho_k = \sum_{l=1, l \neq k}^{K} \chi(d_{kl} - d_{c})
\]

where \( \chi(x) = 1 \) if \( x < 0 \) and \( \chi(x) = 0 \) otherwise, and \( dc \) is a distance threshold. The algorithm is only sensitive to the relative magnitude of \( \rho_k \) in different points.
Algorithm 1. Combination: The Combining of Annotations

**Input:** example annotations $U^1, \ldots, U^K$

**Output:** combined attribute weights vector $\beta$

1: //Computing attribute weights vectors
2: for all $U^k$ do
3: Similar pairs $P^k_S = \emptyset$, Dissimilar pairs $P^k_D = \emptyset$
4: for all $u \in U^k$, $v \in U^k$ do
5: $P^k_S = P^k_S \cup (u, v)$
6: end for
7: repeat
8: Random sample $u$ from set $V \setminus U^k$
9: Random sample $v$ from set $V \setminus U^k$
10: $P^k_D = P^k_D \cup (u, v)$
11: until $d|P_S|$ dissimilar pairs are generated, $d = |F|$ 
12: Oversample from $P_S$ such that $|P_S| = |P_D|$ 
13: Solve objective function in Eq. (1) for diagonal $A^k$
14: $\beta^k = \text{diag}(A^k)$
15: end for
16: //Combining the attribute weights vectors
17: for all $\beta^k$ do
18: Compute $\rho_k$ by Eq. (2)
19: end for
20: Calculate $\beta = \text{norm}(\sum_k \rho_k \beta^k)$
21: return combined attribute weights vector $\beta$
Algorithm 2. Find seed set by re-sampling

Input: attributed graph $G(V, E, F)$, combined weight vector $\beta$, annotations $U^1, \ldots, U^K$

Output: seed set $S$ for expansion

1: re-weigh edges by $\beta$ getting edge re-weight $w(u, v)$
2: for all $(u, v) \in E$ do
3: \hspace{1em} $w(u, v) = 1/(\sqrt{(f_u - f_v)^T diag(\beta)(f_u - f_v) + \epsilon})$
4: end for
5: seed node set $V' = \emptyset$
6: $w_{\text{max}}(w_{\text{min}}) = \max(\min\{w(u, v) | u, v \in \{U^1 \cup \cdots \cup U^K\}\})$
7: if $w(u, v) > w_r = \lambda w_{\text{max}} + (1 - \lambda)w_{\text{min}}$ then
8: \hspace{1em} seed nodes $V' = V' \cup \{u, v\}$
9: end if
10: build seed set graphs $g(V', E', F')$ where
11: \hspace{1em} $\forall u, v \in V', (u, v) \in E, w(u, v) \geq w' \text{iff}(u, v) \in E'$
12: seed set $S \leftarrow G(V', E', F')$
13: return seed set $S$

$$\phi^w(S, G) = \frac{W_{\text{cut}}(S)}{W_{\text{vol}}(S)} = \frac{\sum_{(i,j) \in E, i \in S, j \in V \setminus S} w_{ij}}{\sum_{(i,j) \in S} \sum_{(i,j) \in E} w_{ij}}$$
Fig. 2. The $\lambda$ effects on the clustering results, 50 annotations in each of the experiment.
Experiments

**Fig. 3.** The accuracy of *CGMA*, *FocusCO* and *METIS*.

**Fig. 4.** The scalability of *CGMA*, *FocusCO* and *METIS*. 
Experiments

Fig. 5. The comparison of average NMI value in the 4 clusterings. The horizontal axis in all sub-figures represents the number of annotations we randomly selected.
Table 1. Comparisons on the Scalability of CGMA

| Dataset   | $|V|$ | $|E|$ | $|F|$ | $|C|$ | Running time (sec)         |
|-----------|-----|-----|-----|-----|-----------------------------|
| PolBlog   | 362 | 1288| 44839| 10  | $0.4772 \pm 0.0591$ (CGMA)  |
|           |     |     |      |     | $0.8772 \pm 0.0839$ (METIS) |
|           |     |     |      |     | $3.0561 \pm 0.0471$ (FocusCO)|
| Twitter   | 14078| 44619| 17839| 10  | $1.2135 \pm 0.0322$ (CGMA)  |
|           |     |     |      |     | $1.9425 \pm 0.0381$ (METIS) |
|           |     |     |      |     | $6.8772 \pm 0.0491$ (FocusCO)|