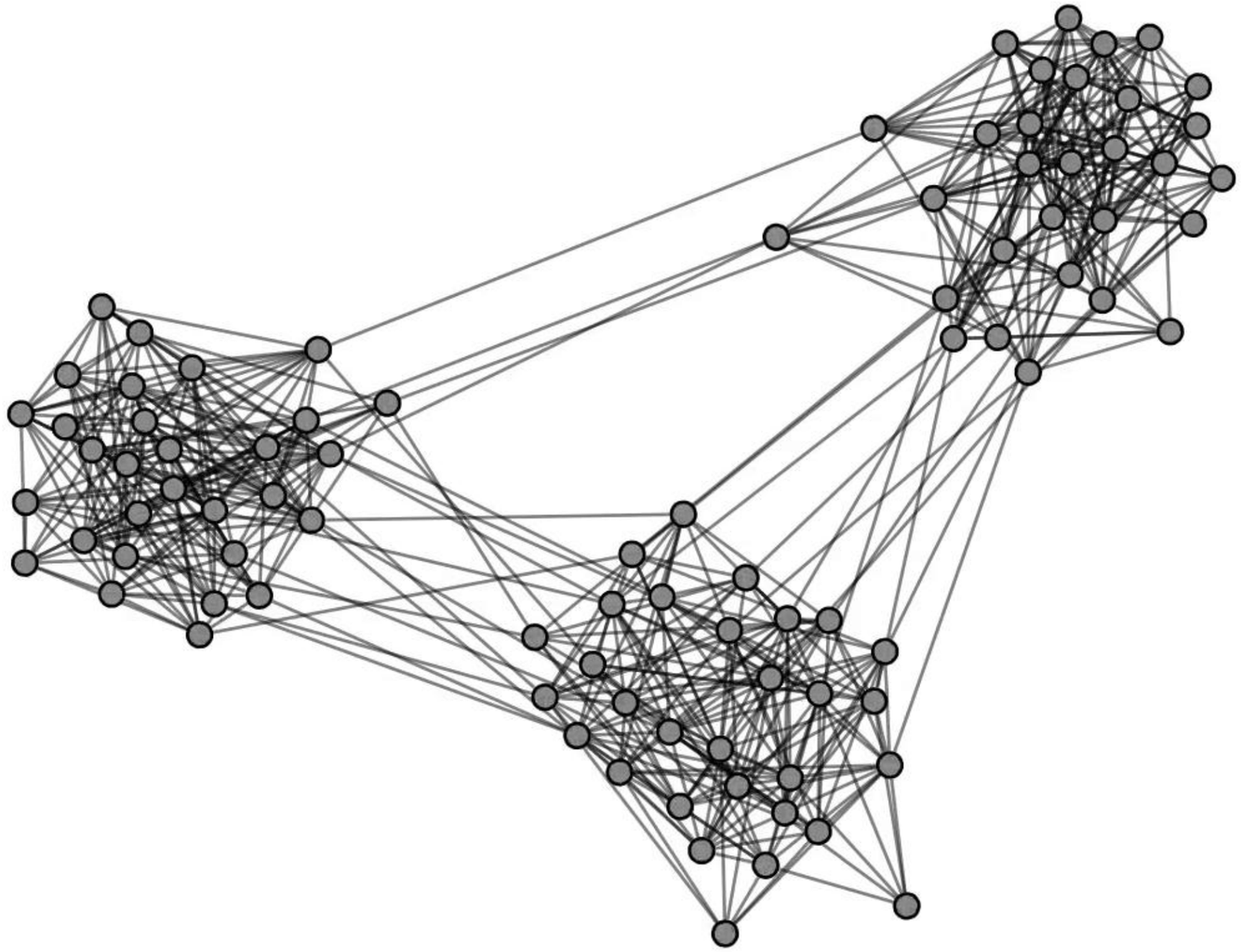
Coreset Spectral Clustering

Ben Jourdan, Gregory Schwartzman, Peter Macgregor, He Sun

THE UNIVERSITY *of* EDINBURGH

University of St Andrews

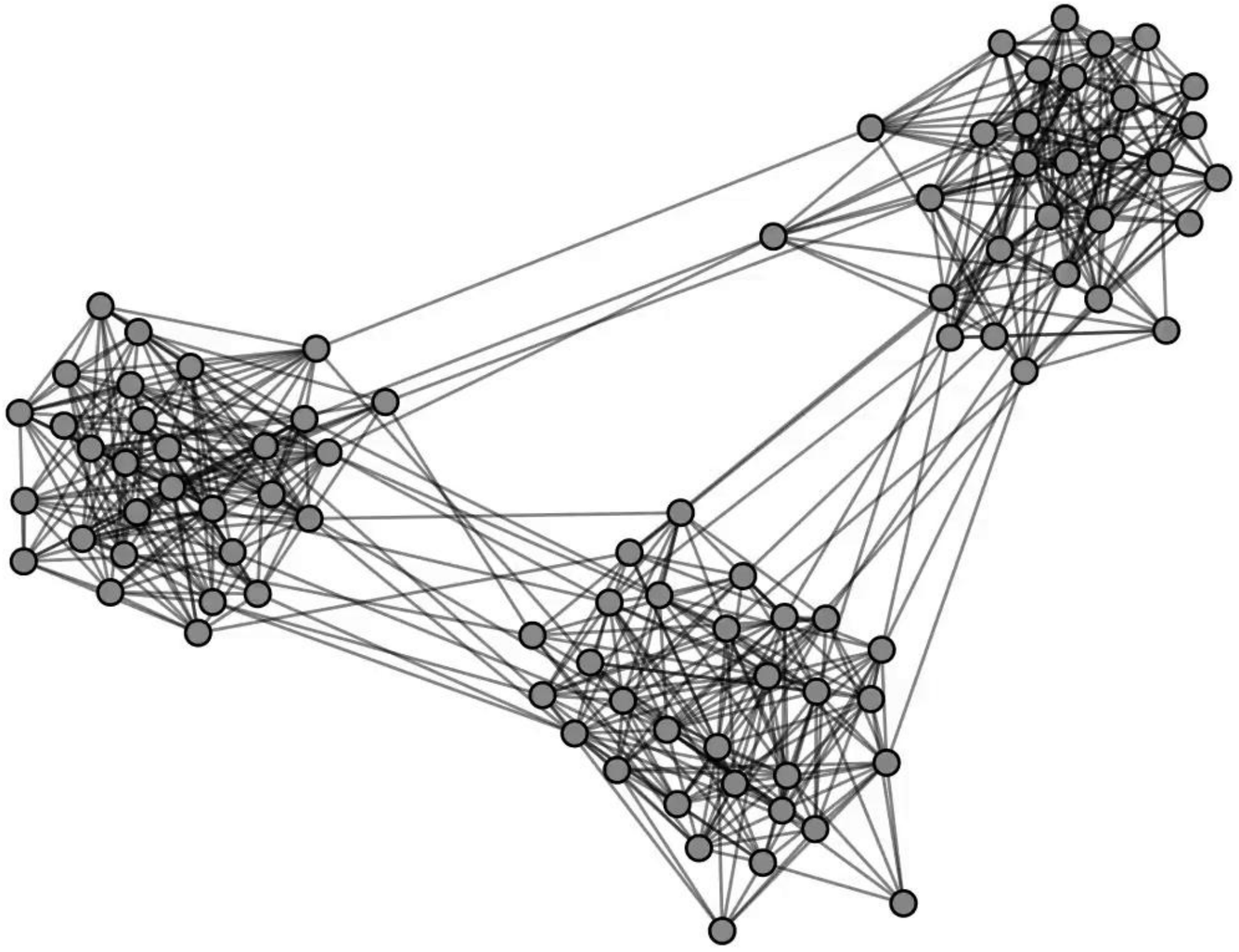
Spectral Clustering



Spectral Clustering

$$A\text{: } A_{ij} = \begin{cases} w_{ij} & \text{if } (i,j) \in E \\ 0 & \text{otherwise} \end{cases}$$

$$D\text{: } D_{ij} = \begin{cases} \sum_k A_{ik} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$



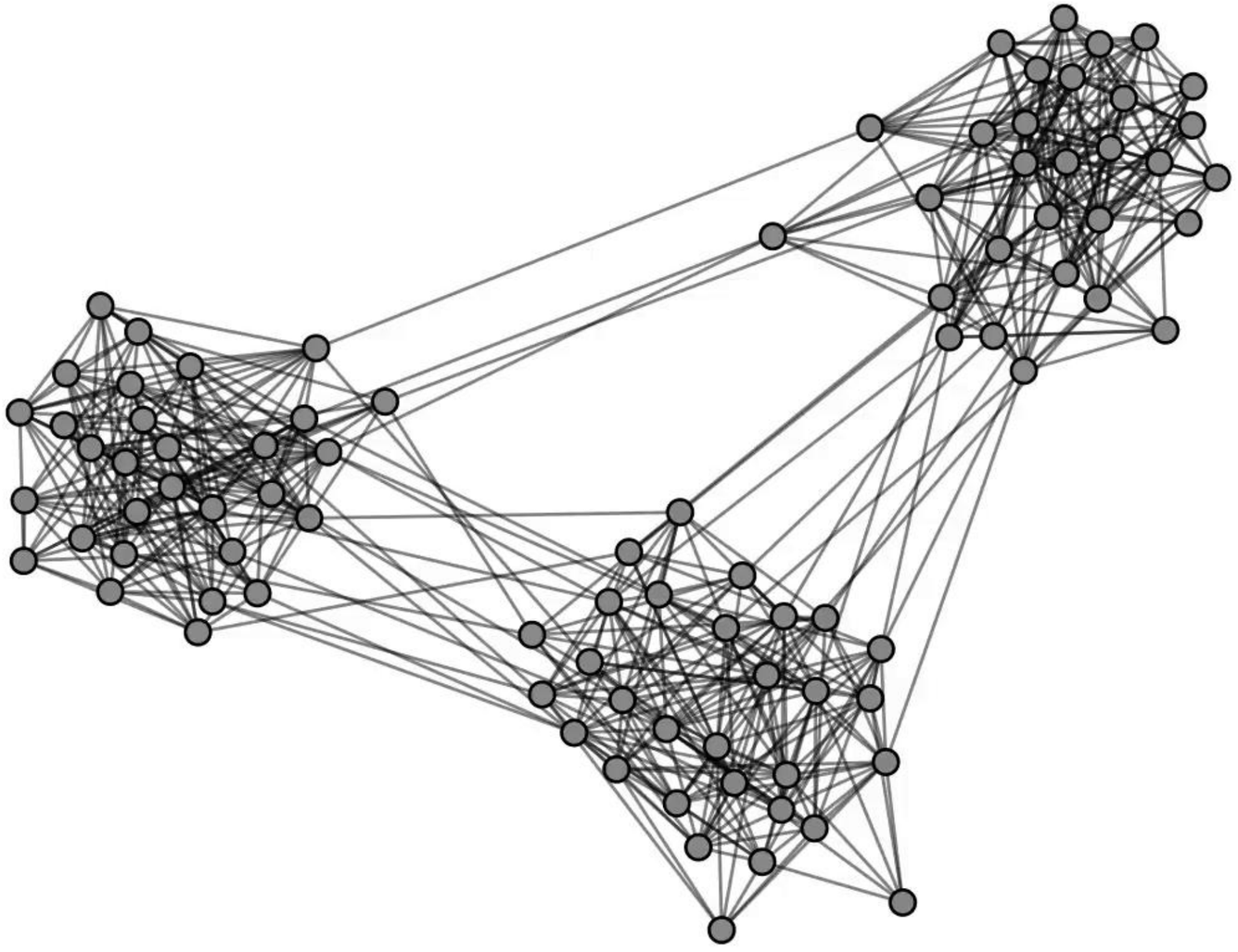
Spectral Clustering

$$A\text{: } A_{ij} = \begin{cases} w_{ij} & \text{if } (i,j) \in E \\ 0 & \text{otherwise} \end{cases}$$

$$D\text{: } D_{ij} = \begin{cases} \sum_k A_{ik} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

1. Construct Laplacian

$$\mathcal{L} = \mathbb{I} - D^{-1/2} A D^{-1/2}$$



Spectral Clustering

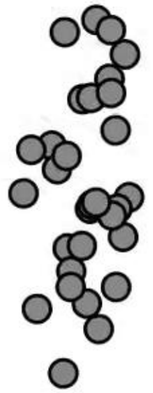
$$A\text{: } A_{ij} = \begin{cases} w_{ij} & \text{if } (i,j) \in E \\ 0 & \text{otherwise} \end{cases}$$

$$D\text{: } D_{ij} = \begin{cases} \sum_k A_{ik} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

1. Construct Laplacian

   $$\mathcal{L} = \mathbb{I} - D^{-1/2} A D^{-1/2}$$

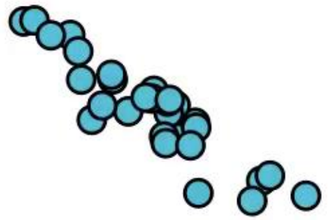
2. Embed nodes into bottom k eigenvectors
   (Slow)

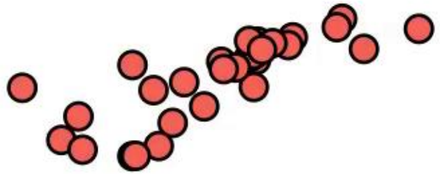


Spectral Clustering

$$A\text{: } A_{ij} = \begin{cases} w_{ij} & \text{if } (i,j) \in E \\ 0 & \text{otherwise} \end{cases}$$

$$D\text{: } D_{ij} = \begin{cases} \sum_k A_{ik} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$





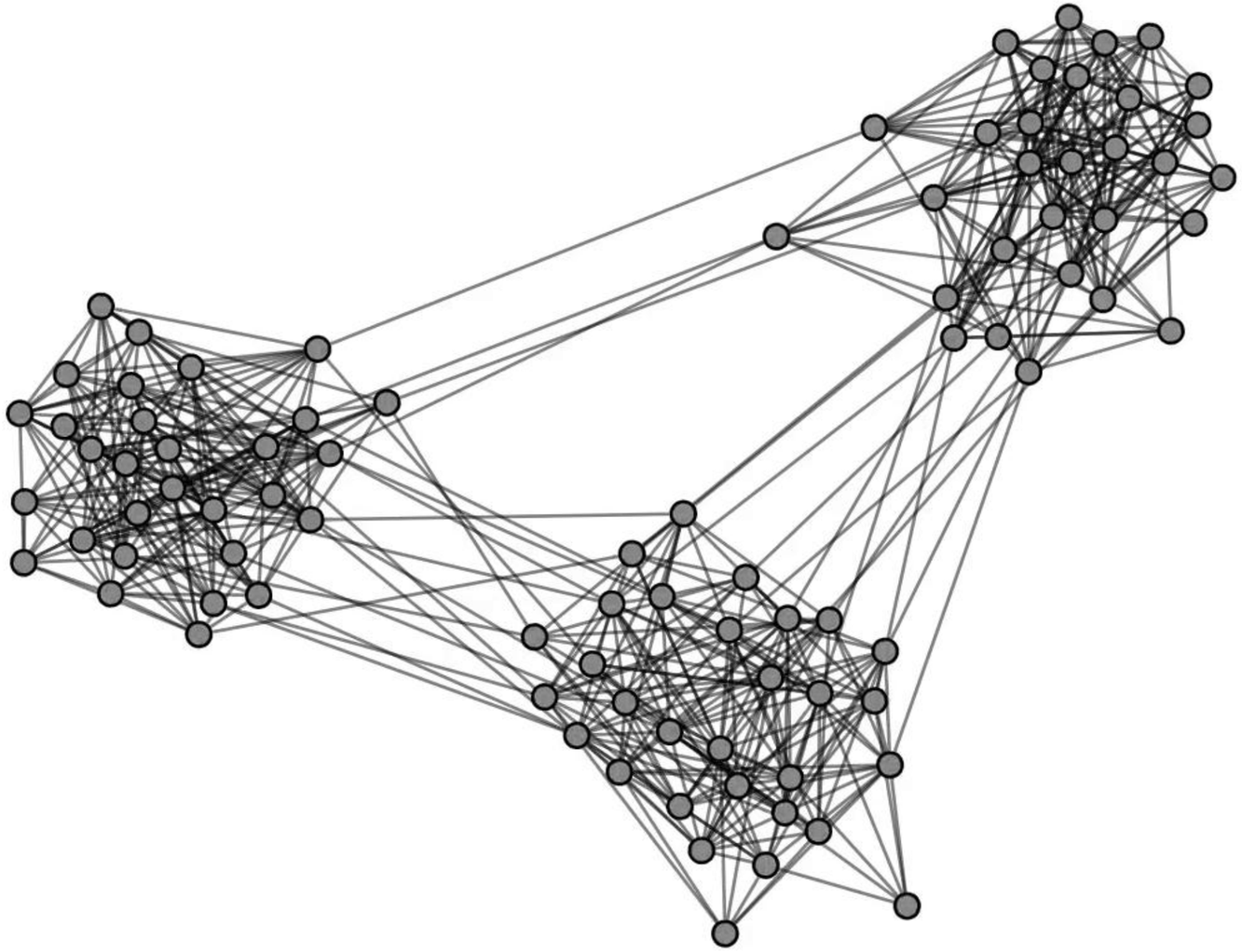
1. Construct Laplacian

   $$\mathcal{L} = \mathbb{I} - D^{-1/2} A D^{-1/2}$$

2. Embed nodes into bottom k eigenvectors
   (Slow)

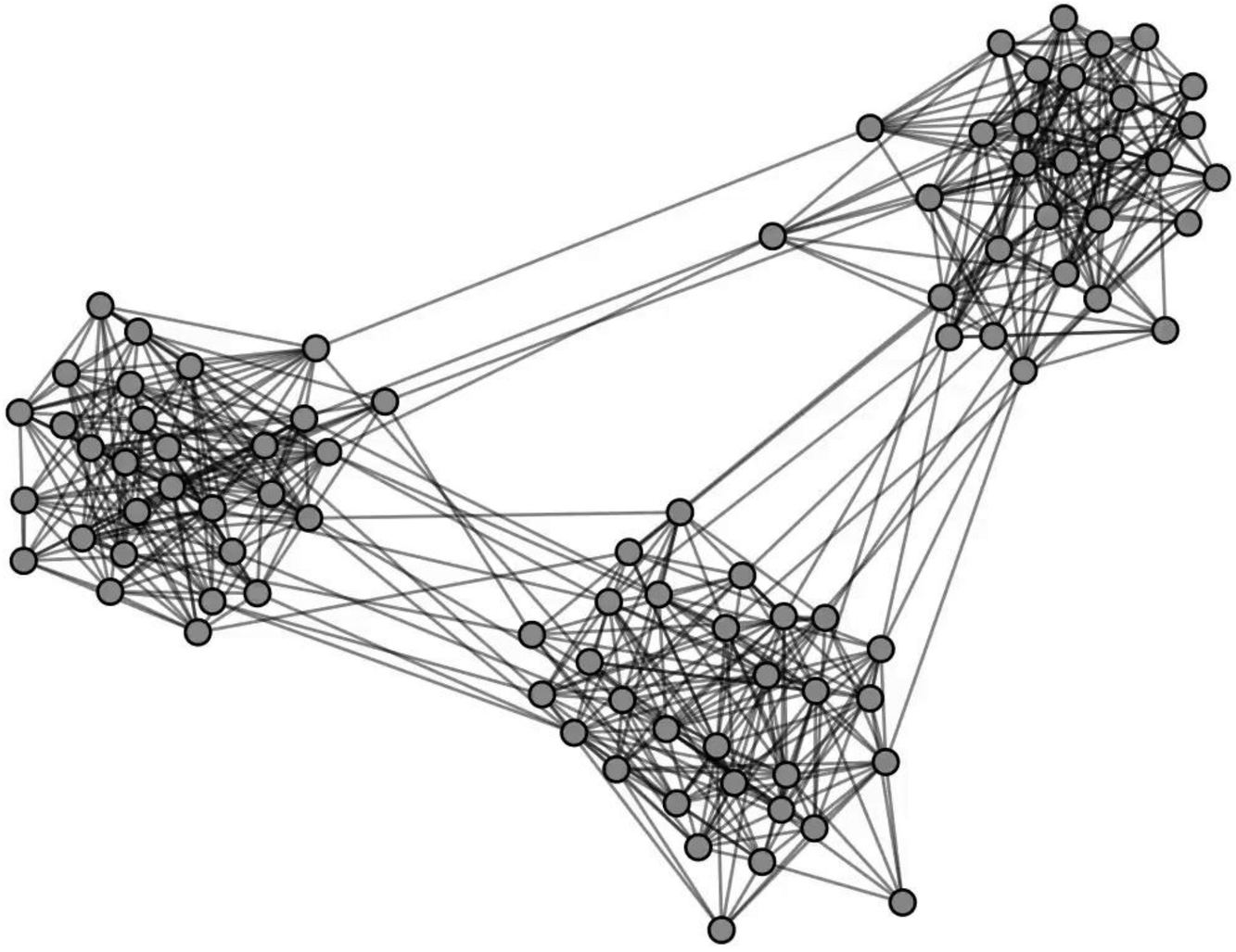
3. Run k-means

Motivation



Motivation

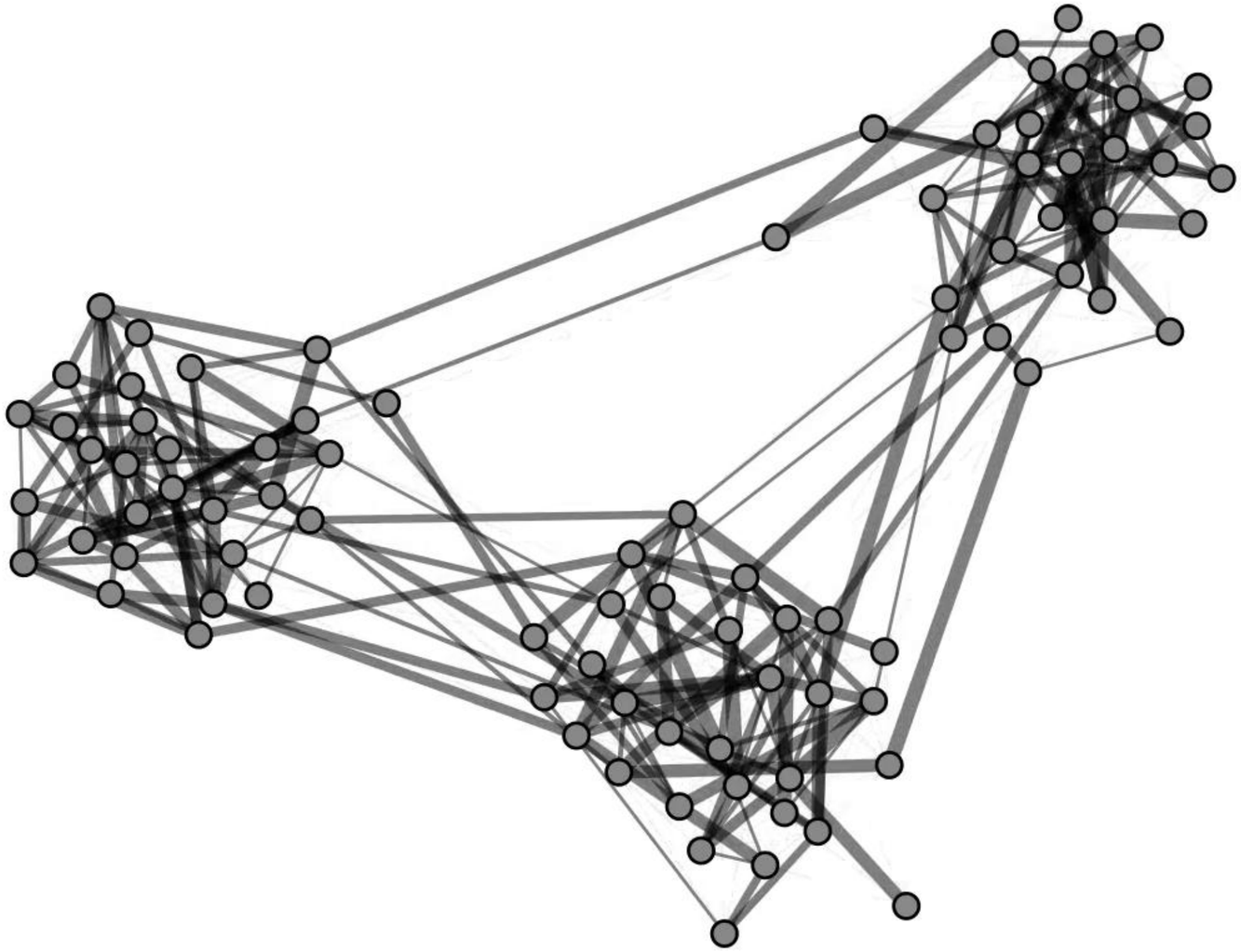
Spectral Clustering is slow



Motivation

Spectral Clustering is slow

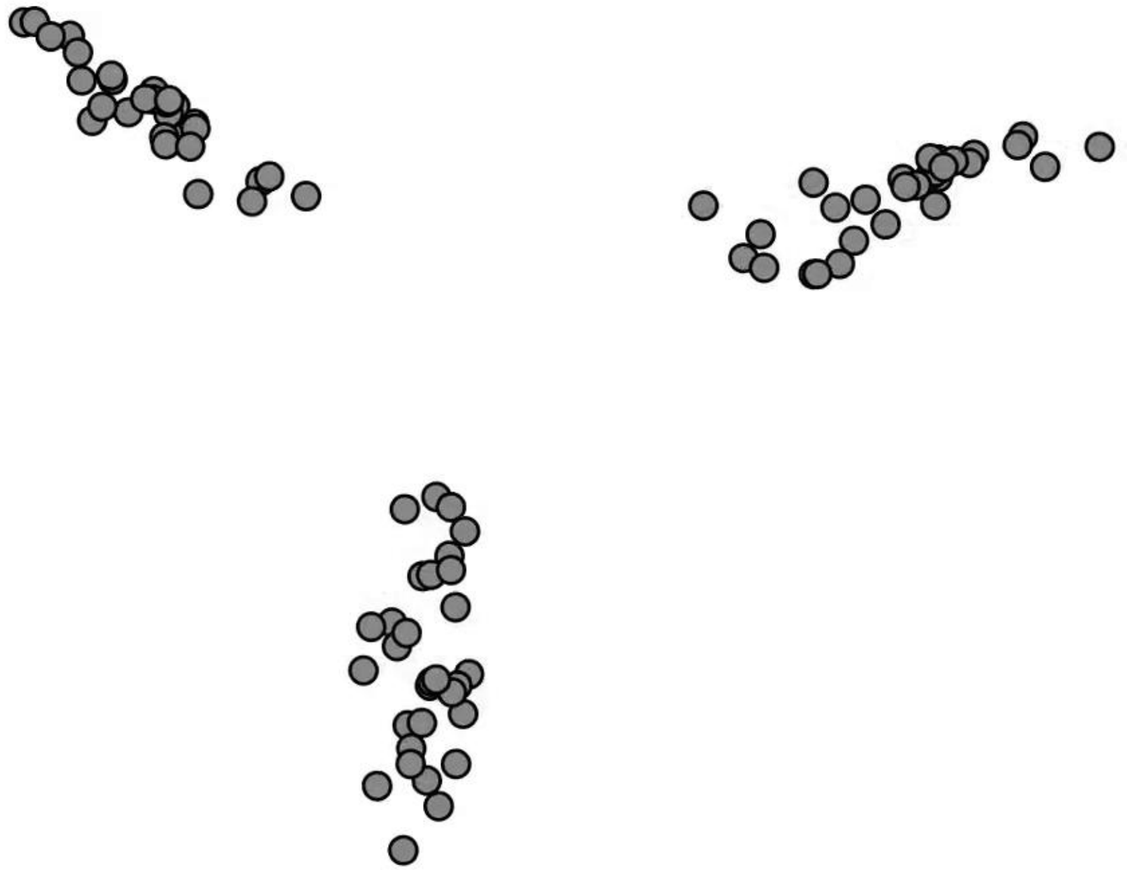
- Sparsification: Fewer edges ✓



Motivation

Spectral Clustering is slow

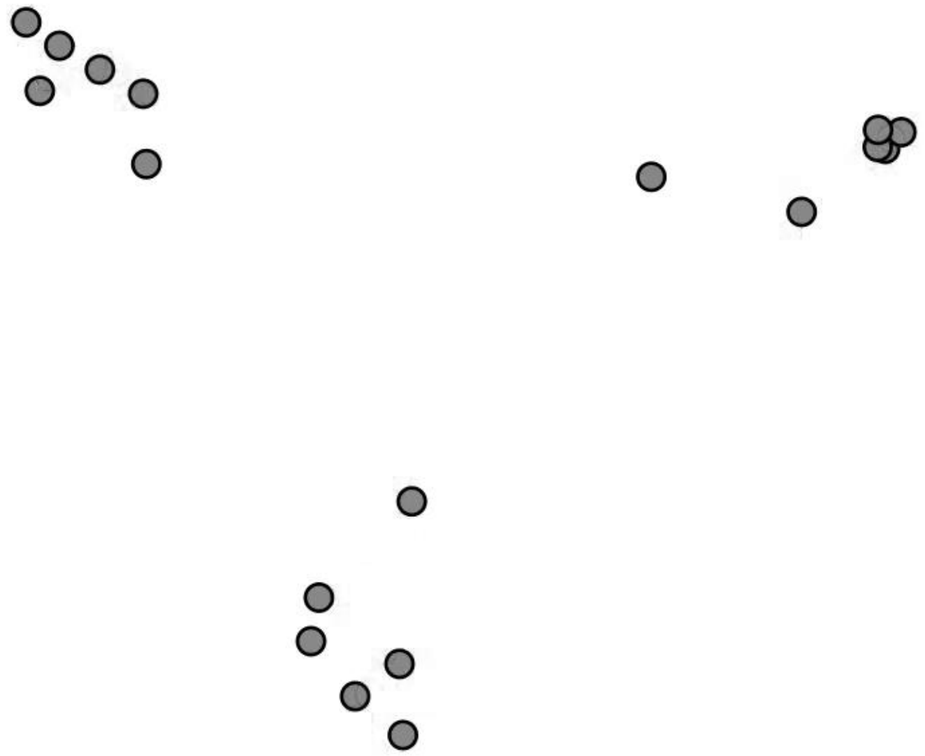
- Sparsification: Fewer edges ✓
- Fewer Eigenvectors ✓



Motivation

Spectral Clustering is slow

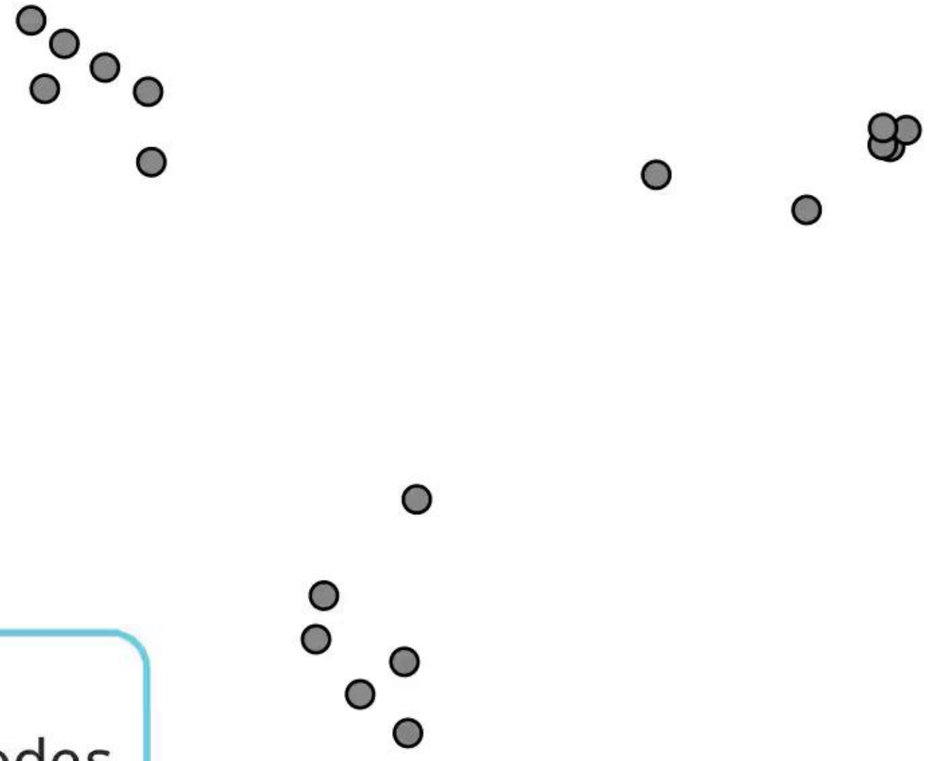
- Sparsification:
  Fewer edges ✓
- Fewer Eigenvectors ✓
- Fewer Nodes ?



Motivation

Spectral Clustering is slow

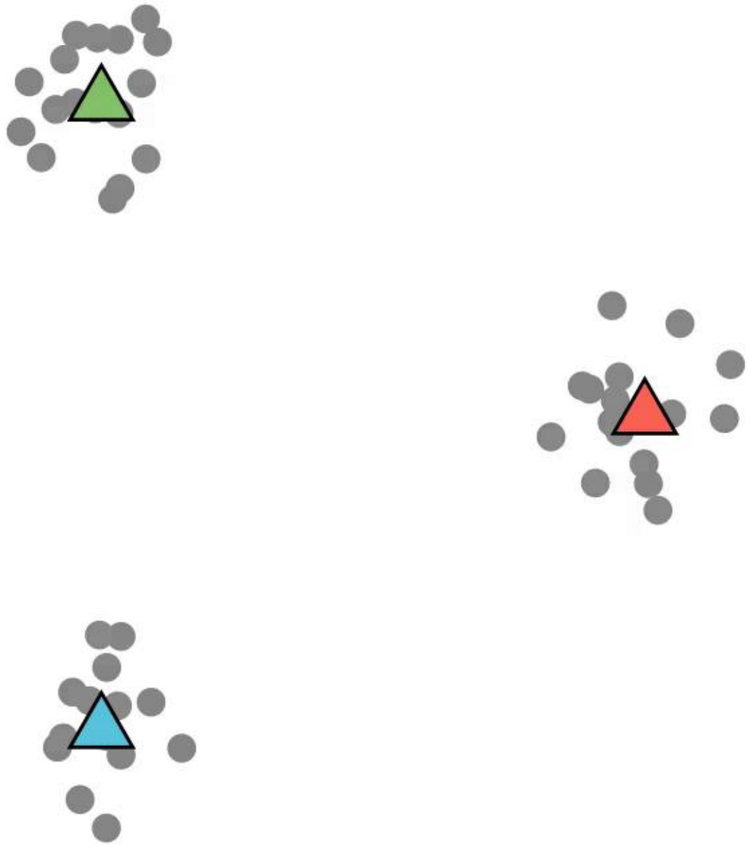
- Sparsification:
  Fewer edges ✓
- Fewer Eigenvectors ✓
- Fewer Nodes ?



Goal:
Speed up Spectral Clustering using fewer nodes

Weighted Kernel K-means

Weighted Kernel K-means



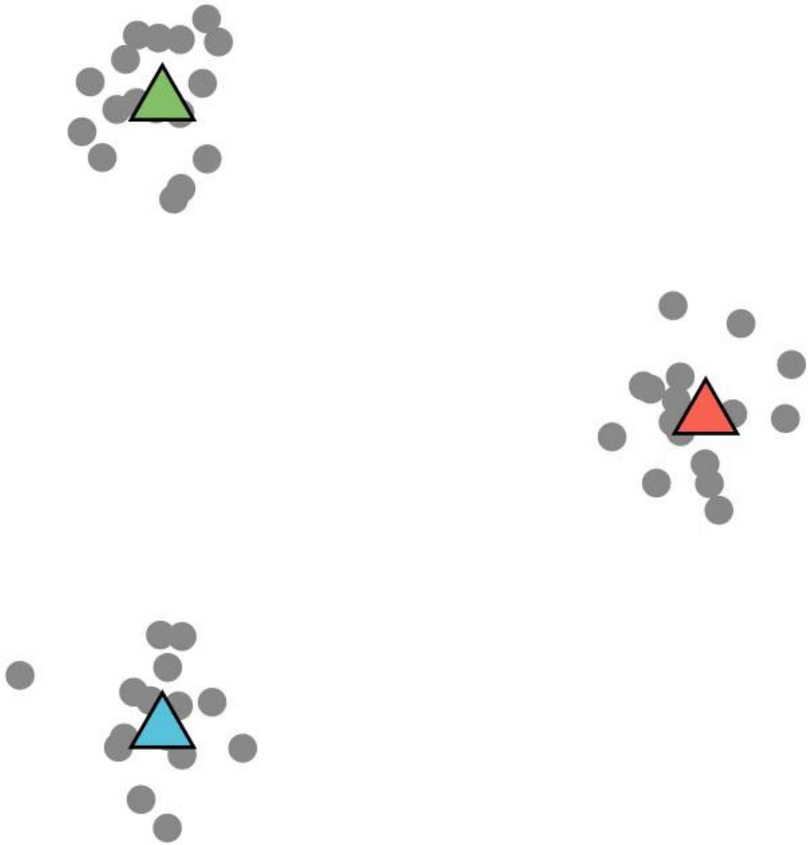
Kernel Trick for distances

$$\|\phi(x) - \phi(y)\|^2 =$$
$$\langle \phi(x), \phi(x) \rangle - 2\langle \phi(x), \phi(y) \rangle + \langle \phi(y), \phi(y) \rangle$$

$$\phi : X \to \mathcal{H} \quad \text{s.t.} \quad \langle \phi(x), \phi(y) \rangle = K(x, y)$$

Radial Basis Functions, edit distance similarity, etc.

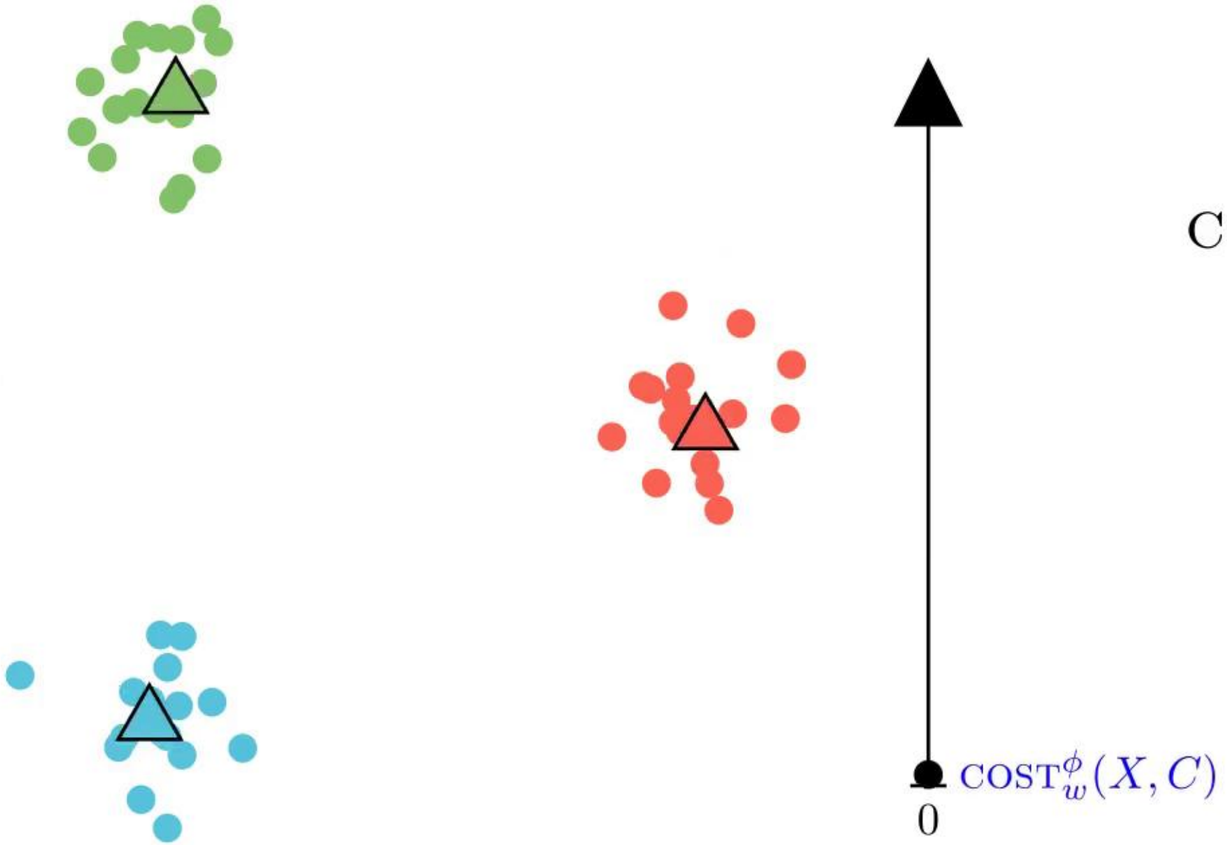
Weighted Kernel K-means



Weighted Kernel k-means objective:

$$\text{COST}_w(X, C) = \sum_{x \in X} w(x) \min_{c \in C} \|\phi(x) - c\|^2$$

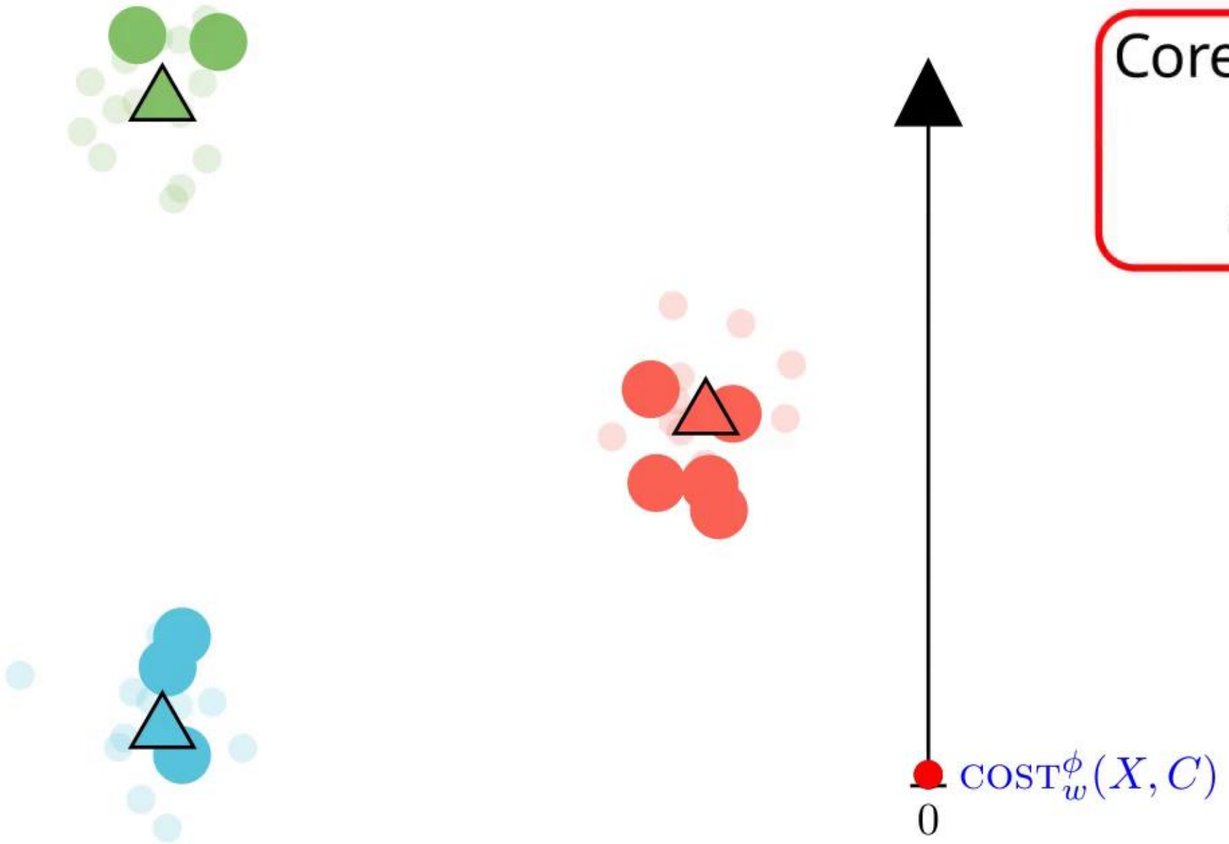
Weighted Kernel K-means



Weighted Kernel k-means objective:

$$\text{COST}_w(X, C) = \sum_{x \in X} w(x) \min_{c \in C} \|\phi(x) - c\|^2$$

Weighted Kernel K-means

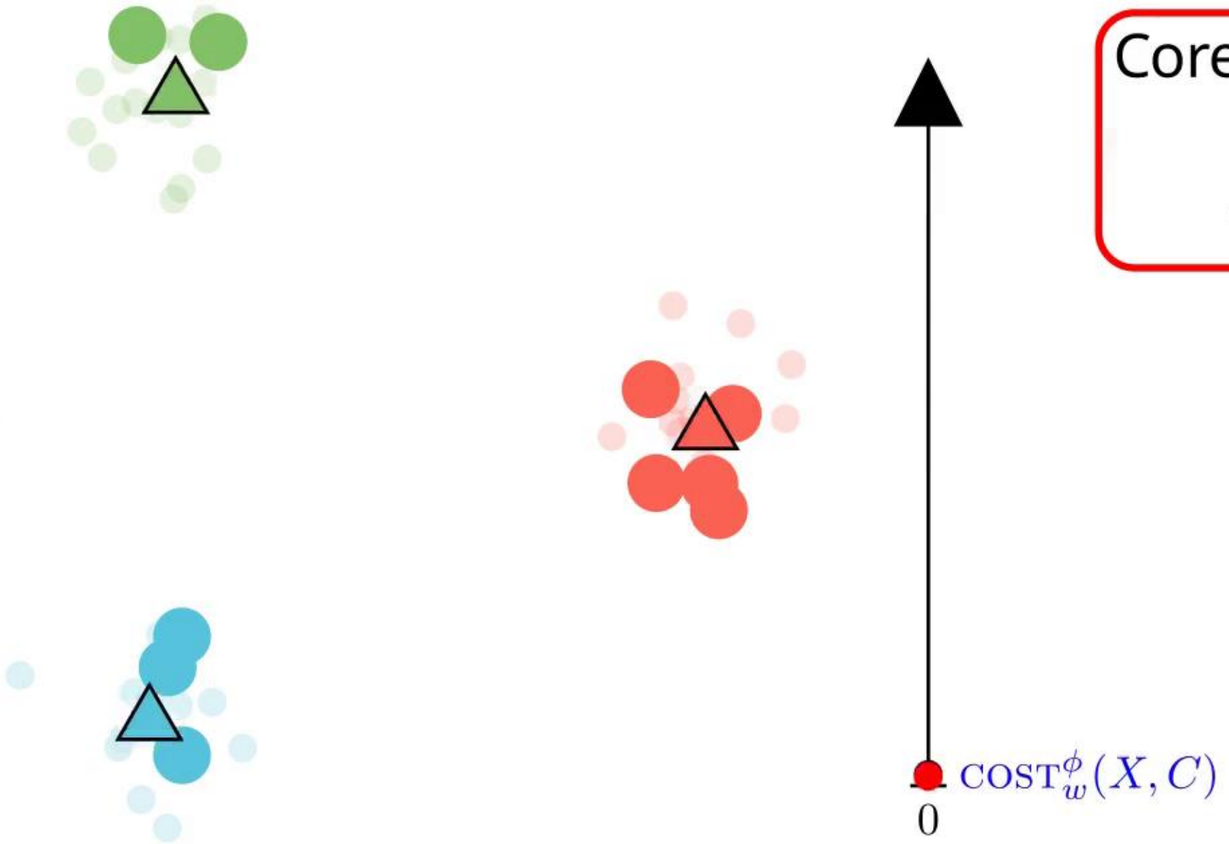


Coreset for Kernel k-means (Jiang et al., 2024):

$$\text{COST}_{w'}(S, C) \in (1 \pm \epsilon)\text{COST}_w(X, C)$$

- Coreset $S \subset X$
- $|S| = O(\text{poly}(k\epsilon^{-1}))$

Weighted Kernel K-means



Coreset for Kernel k-means (Jiang et al., 2024):

$$\text{COST}_{w'}(S, C) \in (1 \pm \epsilon)\text{COST}_w(X, C)$$

- Coreset $S \subset X$
- $|S| = O(\text{poly}(k\epsilon^{-1}))$

The Connection (Dhillon et al. 2004)

The Connection (Dhillon et al. 2004)

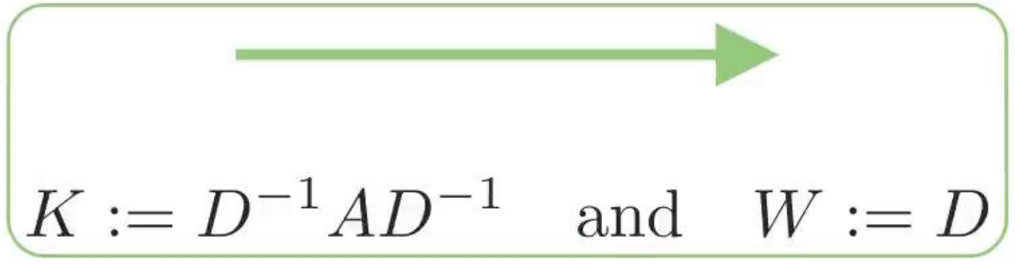
"Normalised Cut" Problem

Weighted Kernel K-Means Problem

The Connection (Dhillon et al. 2004)

"Normalised Cut" Problem

Weighted Kernel K-Means Problem



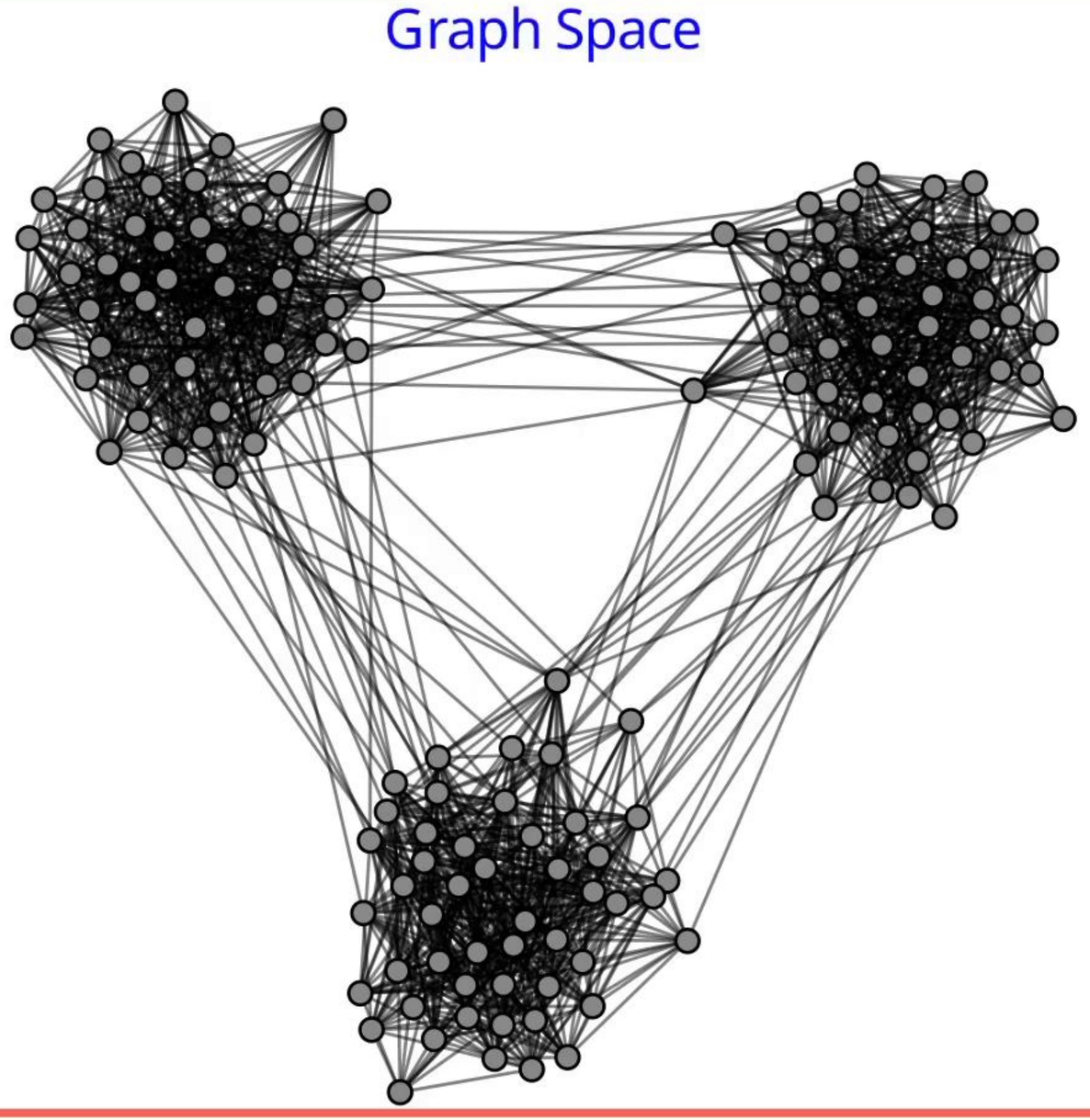
The Connection (Dhillon et al. 2004)

"Normalised Cut" Problem

Weighted Kernel K-Means Problem

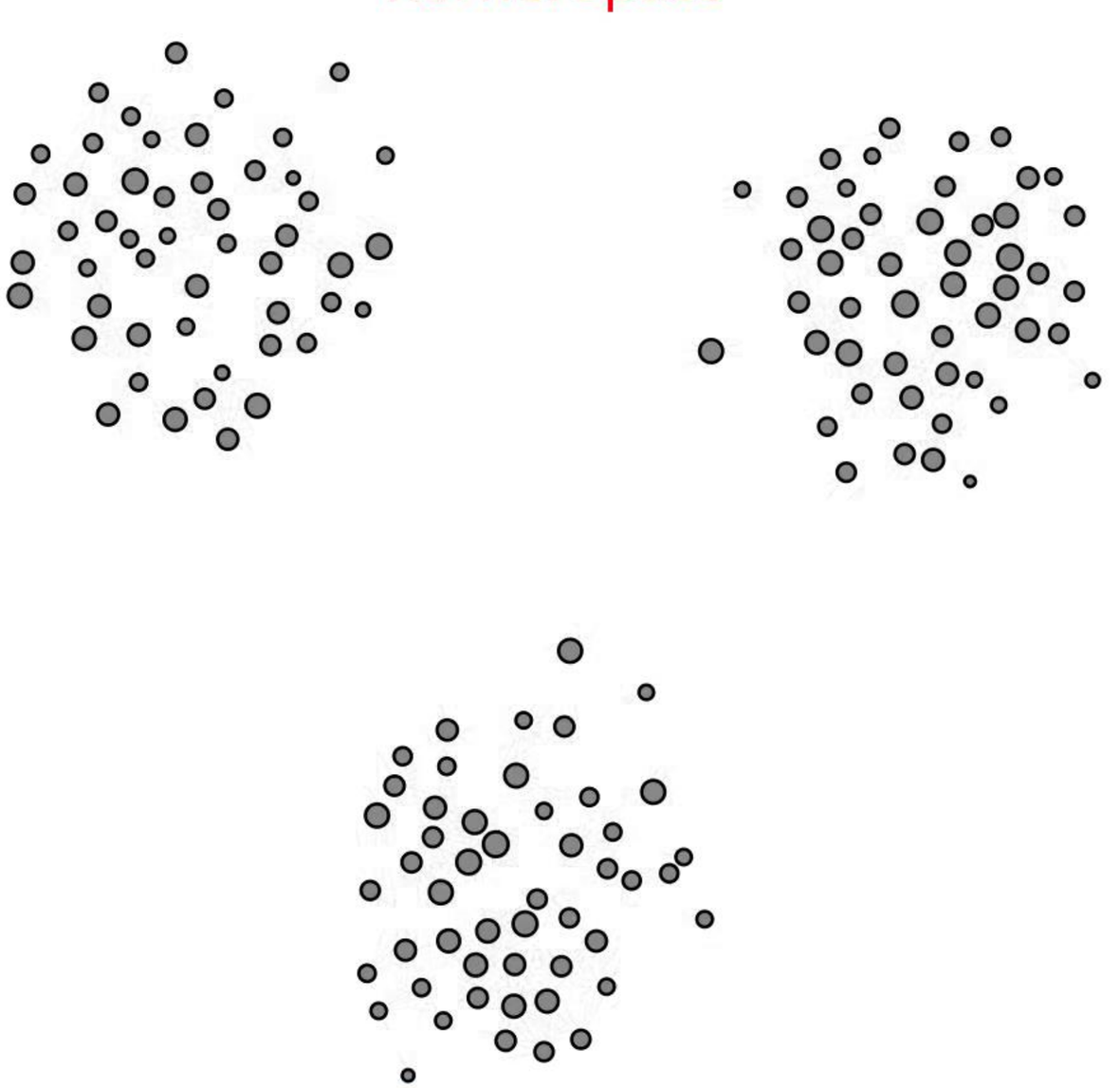
$$D := W \quad \text{and} \quad A := WKW$$

Our Algorithm



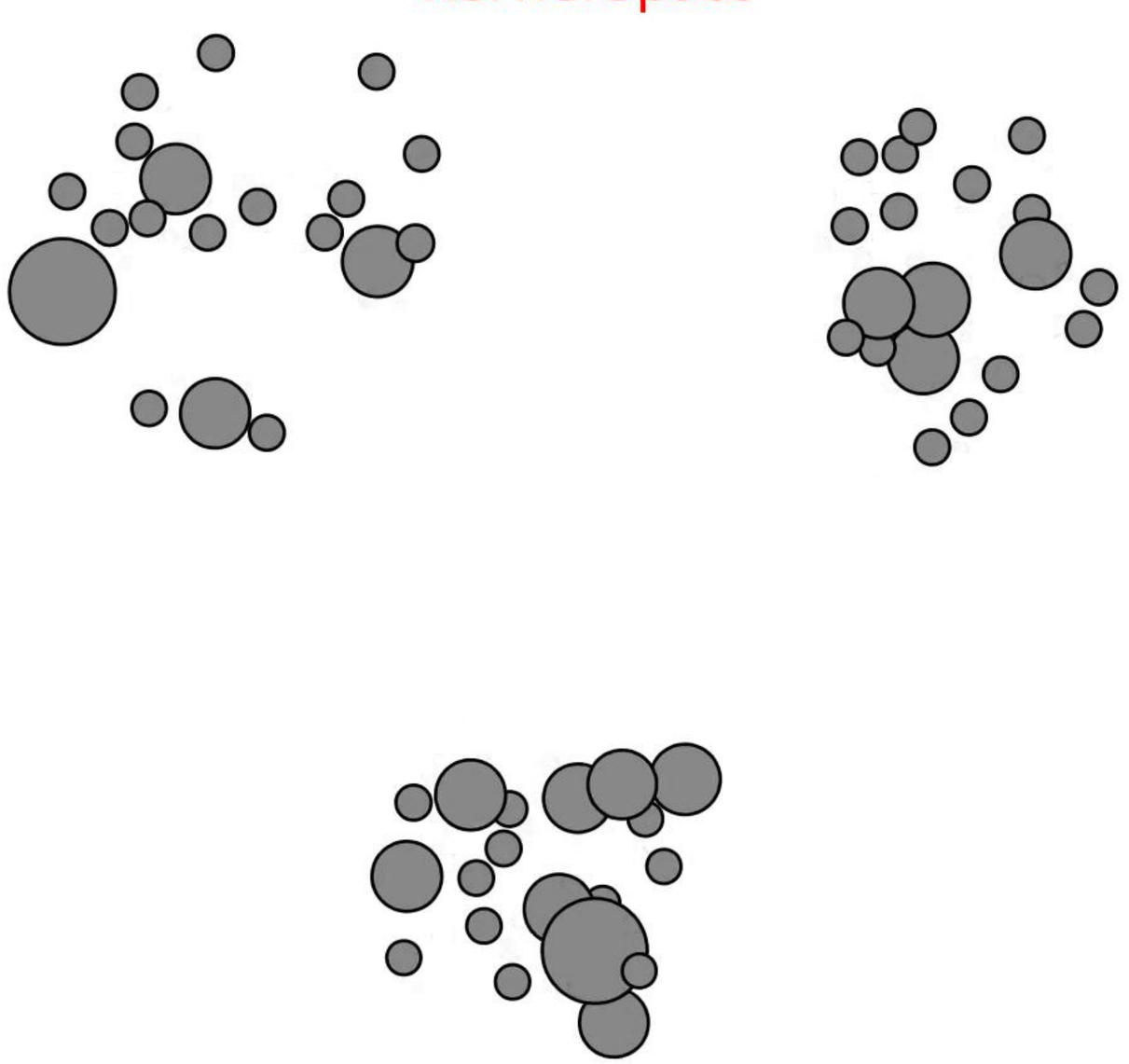
Our Algorithm

1. Translate from graph space to kernel space



Our Algorithm

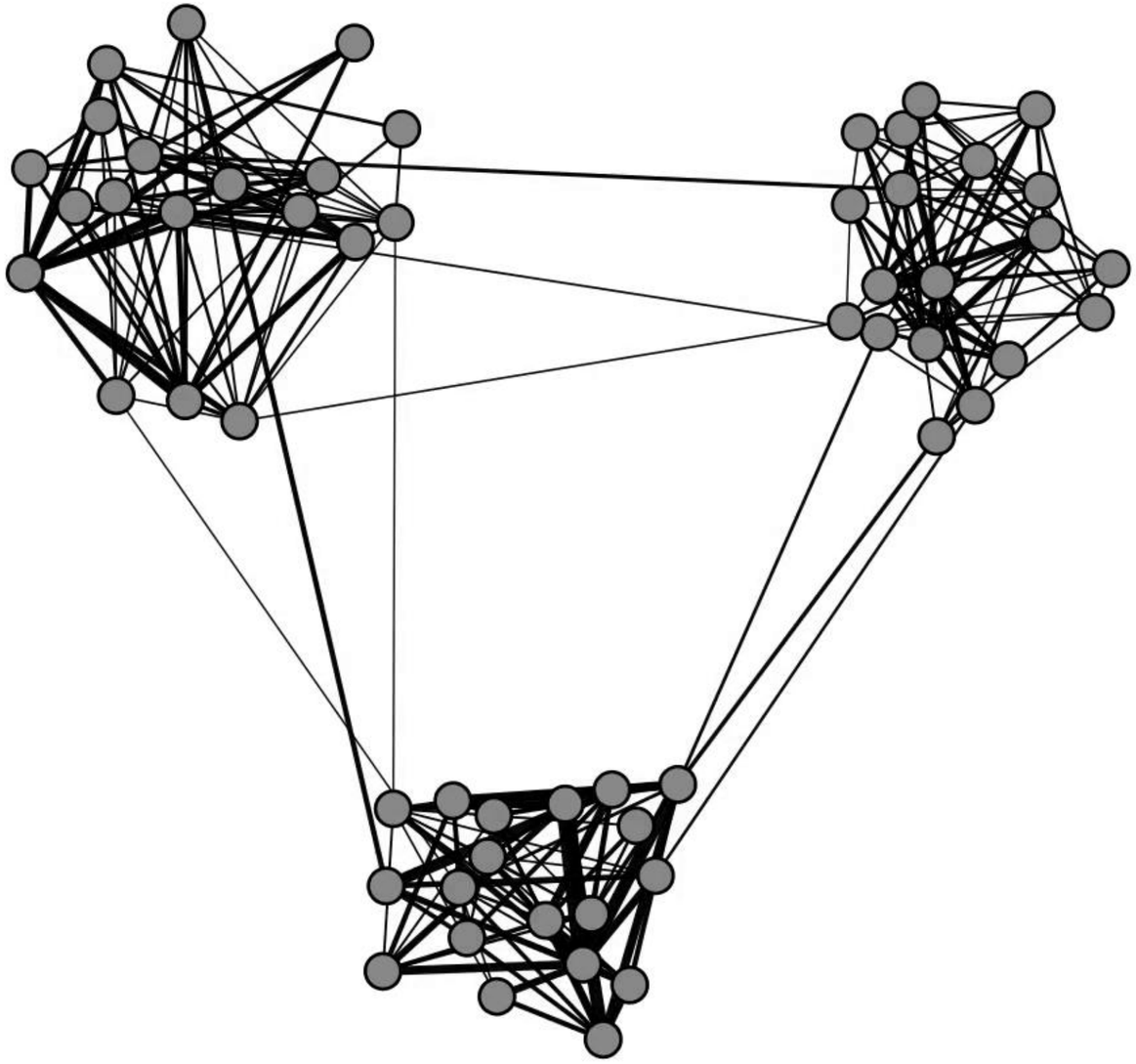
1. Translate from graph space to kernel space
2. Compute a coreset of the kernel problem



Our Algorithm

1. Translate from graph space to kernel space
2. Compute a coreset of the kernel problem
3. Translate coreset back to graph space

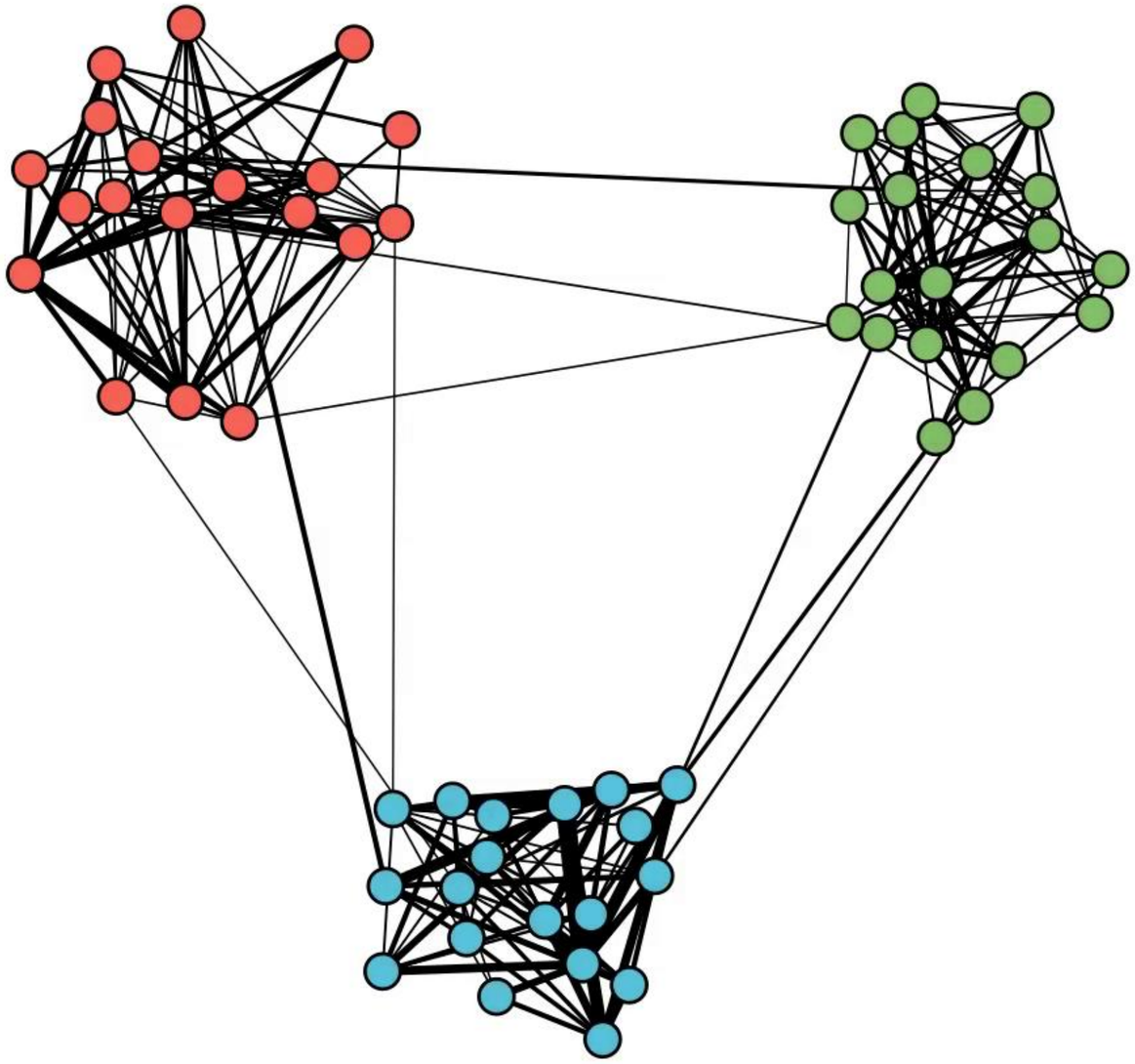
Graph Space



Our Algorithm

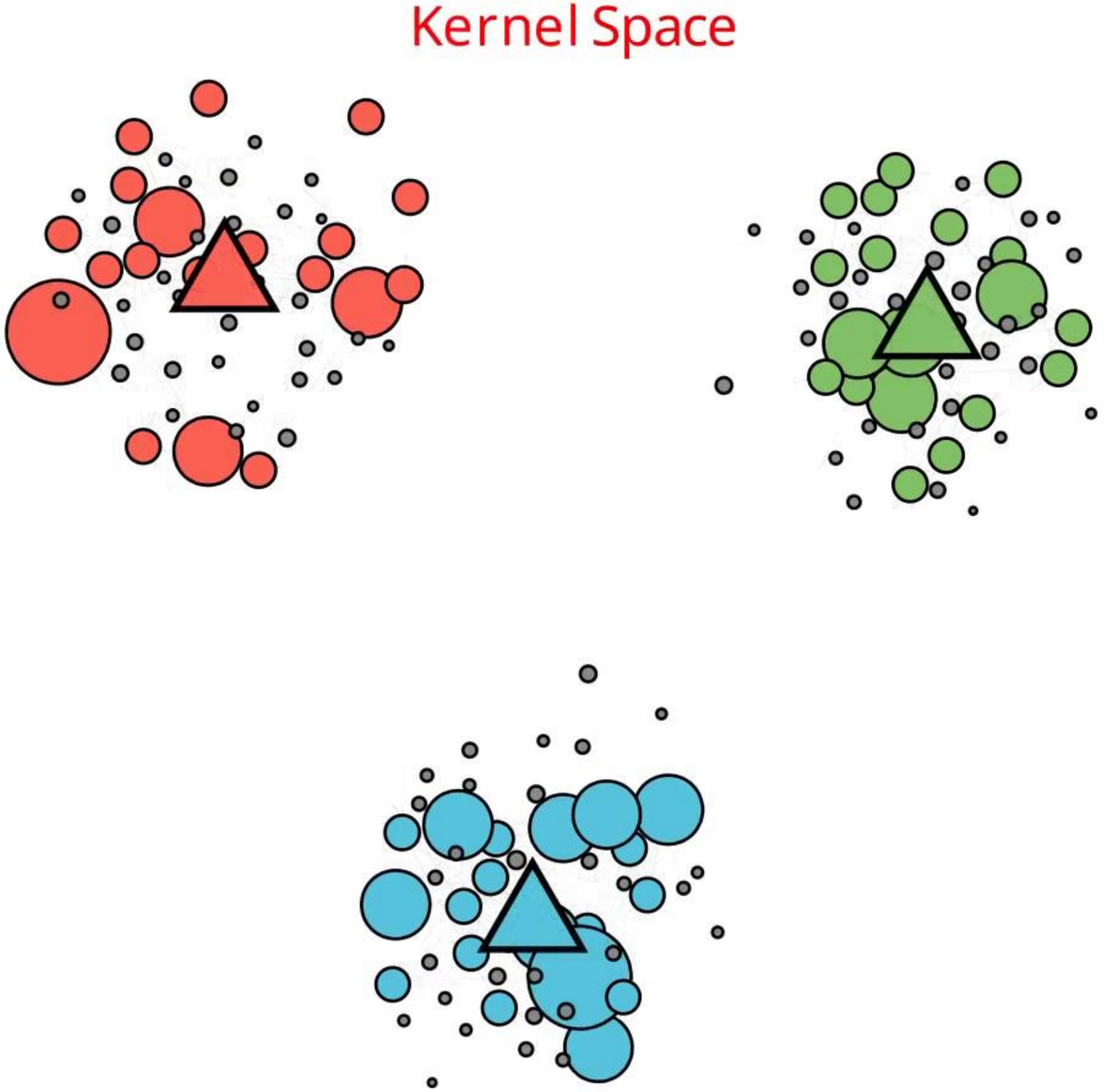
1. Translate from graph space to kernel space
2. Compute a coreset of the kernel problem
3. Translate coreset back to graph space
4. Perform spectral clustering on the coreset graph (Fast!)

Graph Space



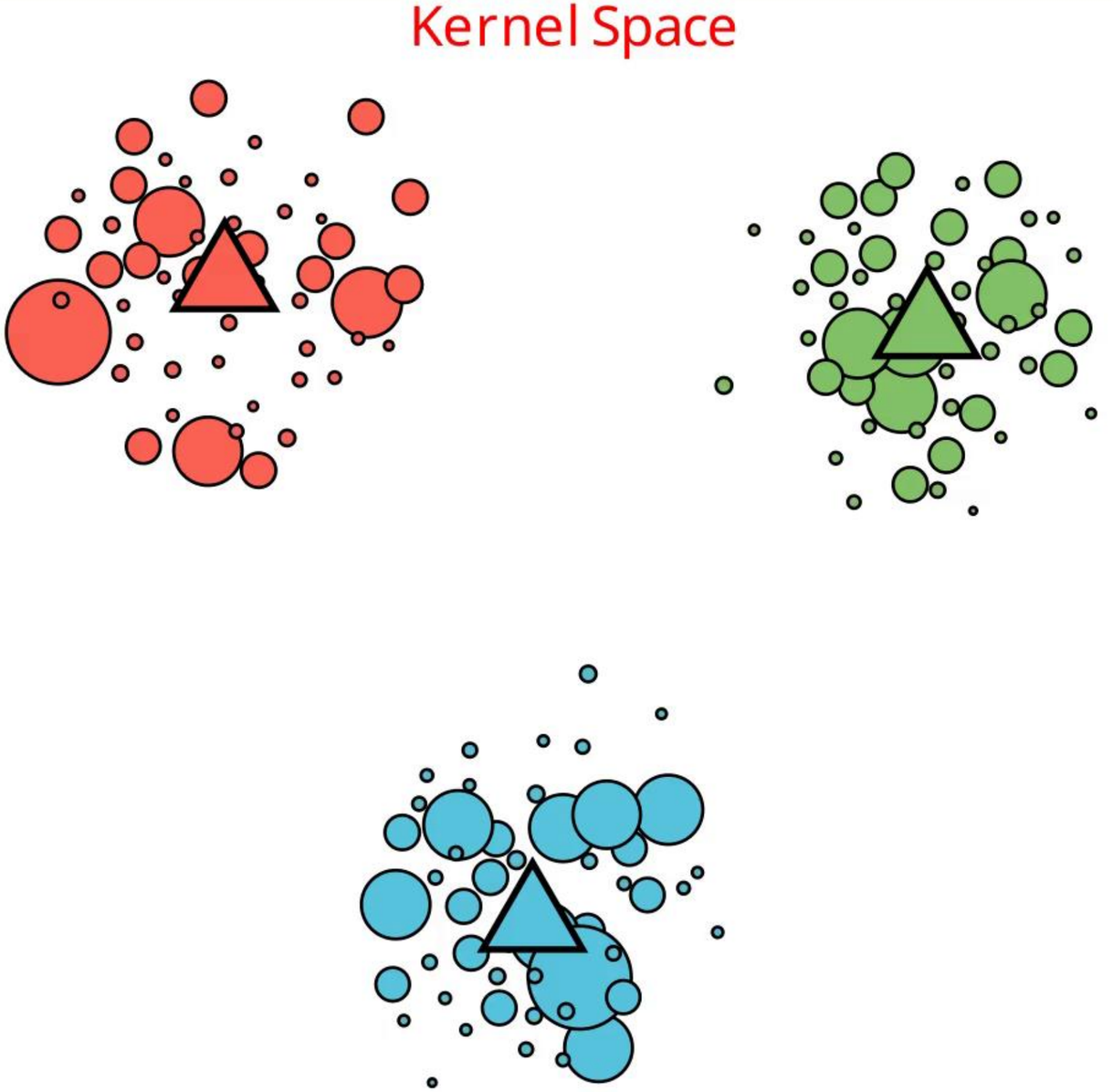
Our Algorithm

1. Translate from graph space to kernel space
2. Compute a coreset of the kernel problem
3. Translate coreset back to graph space
4. Perform spectral clustering on the coreset graph (Fast!)
5. Translate cluster partitions into centroids in kernel space



Our Algorithm

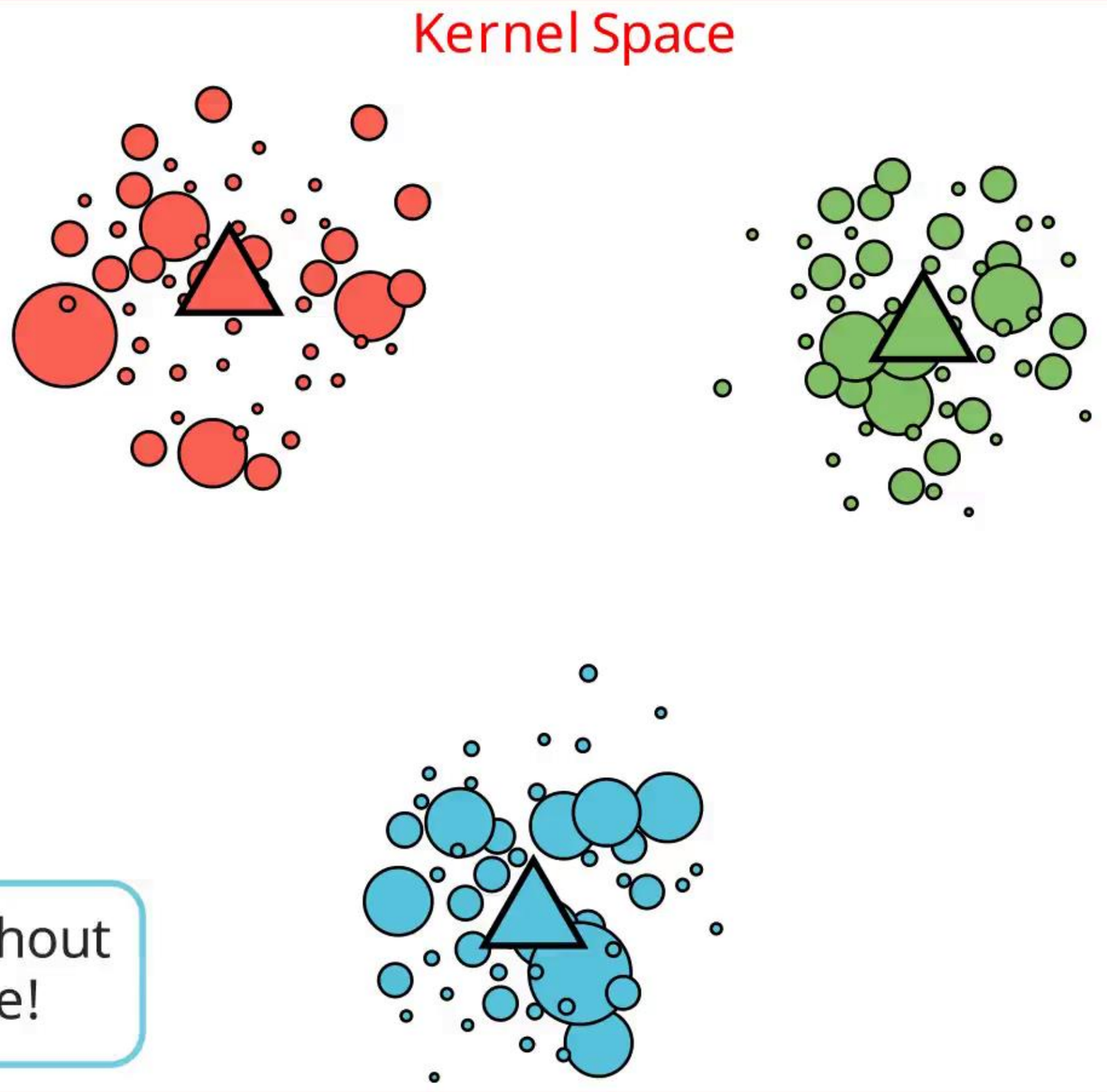
1. Translate from graph space to kernel space
2. Compute a coreset of the kernel problem
3. Translate coreset back to graph space
4. Perform spectral clustering on the coreset graph (Fast!)
5. Translate cluster partitions into centroids in kernel space
6. Label the rest of the nodes in kernel space



Our Algorithm

1. Translate from graph space to kernel space
2. Compute a coreset of the kernel problem
3. Translate coreset back to graph space
4. Perform spectral clustering on the coreset graph (Fast!)
5. Translate cluster partitions into centroids in kernel space
6. Label the rest of the nodes in kernel space

All the benefits of coreset kernel k-means without having to run Lloyd's algorithm in kernel space!



Our Results

Our Results

Performance Guarantee:

Achieves an $\alpha\frac{1+\epsilon}{1-\epsilon}$-approximation ratio for the "Normalised Cut" problem

α is the approximation ratio of the spectral clustering algorithm

ϵ comes from the $(1 \pm \epsilon)$ kernel coreset guarantee

Our Results

Fast coreset algorithm for sparse kernels

We improve the running time of the previously SOTA algorithm from $\tilde{O}(nk)$ to $\tilde{O}(n \cdot \min\{k, d_{avg}\})$

n is the number of nodes

k is the number of clusters

d_{avg} is the average number of non-zero entries in each row of the kernel matrix

Speeding up Coreset Construction

Speeding up Coreset Construction

k-means++ in kernel space (jiang et al. 2024):

1: **Input:** X
2: Draw $x \in X$ uniformly at random, and initialise $C \leftarrow \{\phi(x)\}$
3: **for** $i = 1, \ldots, k-1$ **do**
4: Draw one sample $x \in X$, using probabilities $\frac{w_X(x)\Delta(\phi(x),C)}{\sum_{x \in X} w_X(x)\Delta(\phi(x),C)}$
5: Let $C \leftarrow C \cup \{\phi(x)\}$
6: **end for**
7: **return** C

Speeding up Coreset Construction

$$
\begin{aligned}
\Delta(\phi(x), \phi(y)) &\triangleq \|\phi(x) - \phi(y)\|^2 \\
&= \langle \phi(x), \phi(x) \rangle + \langle \phi(y), \phi(y) \rangle - 2\langle \phi(x), \phi(y) \rangle \\
&= \begin{cases} \langle \phi(x), \phi(x) \rangle + \langle \phi(y), \phi(y) \rangle - 2\langle \phi(x), \phi(y) \rangle & \text{if } x \sim y \\ \langle \phi(x), \phi(x) \rangle + \langle \phi(y), \phi(y) \rangle & \text{if } x \not\sim y \end{cases}
\end{aligned}
$$

Speeding up Coreset Construction

$$\Delta(\phi(x), \phi(y)) \triangleq \|\phi(x) - \phi(y)\|^2$$
$$= \langle \phi(x), \phi(x) \rangle + \langle \phi(y), \phi(y) \rangle - 2\langle \phi(x), \phi(y) \rangle$$
$$= \begin{cases} \langle \phi(x), \phi(x) \rangle + \langle \phi(y), \phi(y) \rangle - 2\langle \phi(x), \phi(y) \rangle & \text{if } x \sim y \\ \langle \phi(x), \phi(x) \rangle + \langle \phi(y), \phi(y) \rangle & \text{if } x \not\sim y \end{cases}$$

Let $C = \{\phi(x_1), \ldots, \phi(x_l)\}$ be a set of points in kernel space. Define

$$\Delta(\phi(x), C) \triangleq \min_{c \in C} \Delta(\phi(x), c)$$
$$= \langle \phi(x), \phi(x) \rangle + \begin{cases} \min_{c \in C} \langle c, c \rangle - 2\langle \phi(x), c \rangle & x \sim C \\ \min_{c \in C} \langle c, c \rangle & x \not\sim C \end{cases}$$

Speeding up Coreset Construction

k-means++ in kernel space (jiang et al. 2024):

1: **Input:** X
2: Draw $x \in X$ uniformly at random, and initialise $C \leftarrow \{\phi(x)\}$
3: **for** $i = 1, \ldots, k-1$ **do**
4: Draw one sample $x \in X$, using probabilities $\frac{w_X(x)\Delta(\phi(x),C)}{\sum_{x \in X} w_X(x)\Delta(\phi(x),C)}$
5: Let $C \leftarrow C \cup \{\phi(x)\}$
6: **end for**
7: **return** C

Speeding up Coreset Construction

Suppose we first add the datapoint x^* such that $x^* \triangleq \arg\min_{x \in X} \langle \phi(x), \phi(x) \rangle$.

Speeding up Coreset Construction

Suppose we first add the datapoint x^* such that $x^* \triangleq \arg\min_{x \in X} \langle \phi(x), \phi(x) \rangle$.

$$\text{Then} \quad \Delta(\phi(x), C \cup \{y\}) = \begin{cases} \Delta(\phi(x), C) & x \not\sim y \\ \min(\Delta(\phi(x), C), \Delta(\phi(x), \phi(y))) & x \sim y. \end{cases}$$

Speeding up Coreset Construction

Suppose we first add the datapoint x^* such that $x^* \triangleq \arg\min_{x \in X} \langle \phi(x), \phi(x) \rangle$.

$$\text{Then} \quad \Delta(\phi(x), C \cup \{y\}) = \begin{cases} \Delta(\phi(x), C) & x \not\sim y \\ \min(\Delta(\phi(x), C), \Delta(\phi(x), \phi(y))) & x \sim y. \end{cases}$$

This implies the total number of checks is $O(\min(d_{avg}, k) \cdot n))$

Speeding up Coreset Construction

Speeding up Coreset Construction

How do we sample proportional to and update $w(\cdot)\Delta(\cdot, C)$, efficiently?

Speeding up Coreset Construction

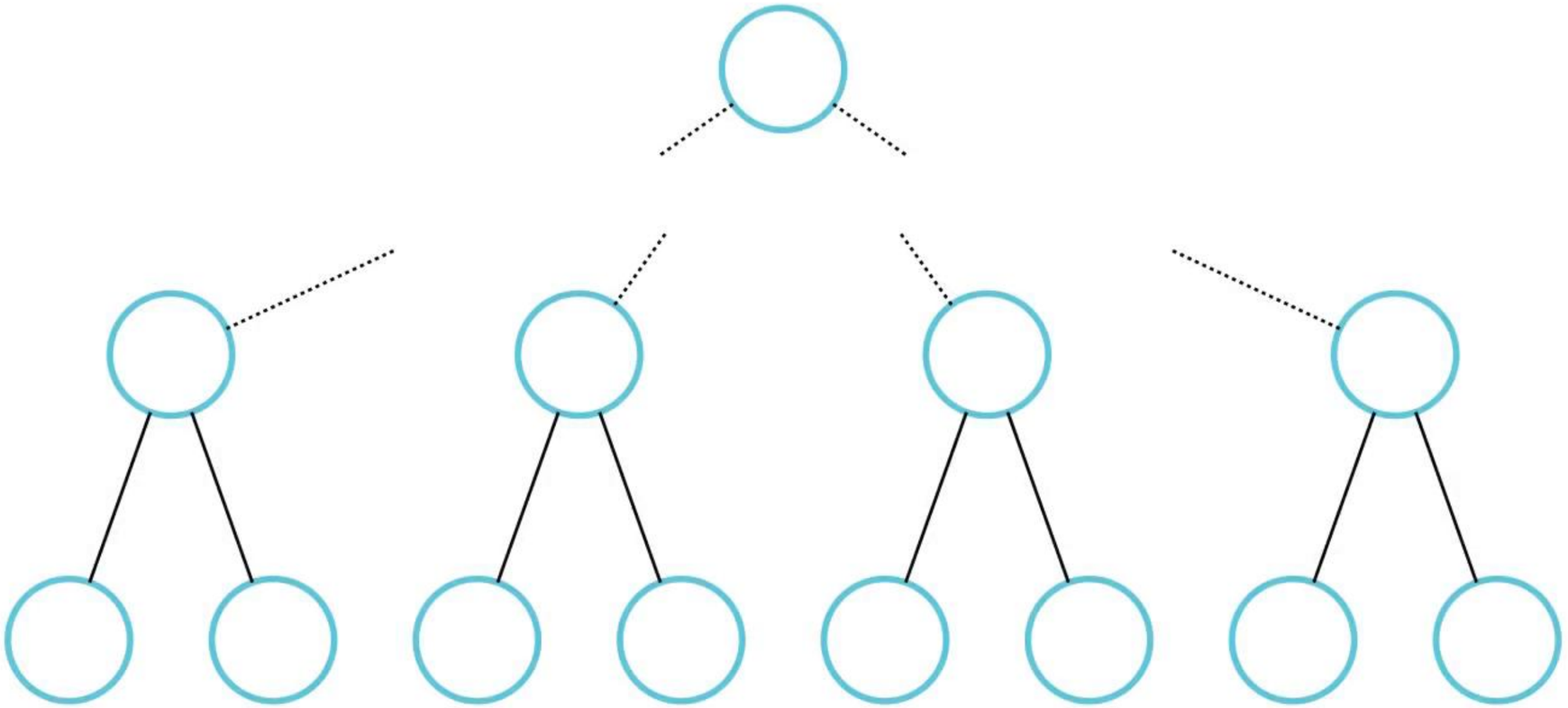
How do we sample proportional to and update $w(\cdot)\Delta(\cdot, C)$, efficiently?

Answer: Sampling trees! Sampling and updating in time $O(\log n)$.

Speeding up Coreset Construction

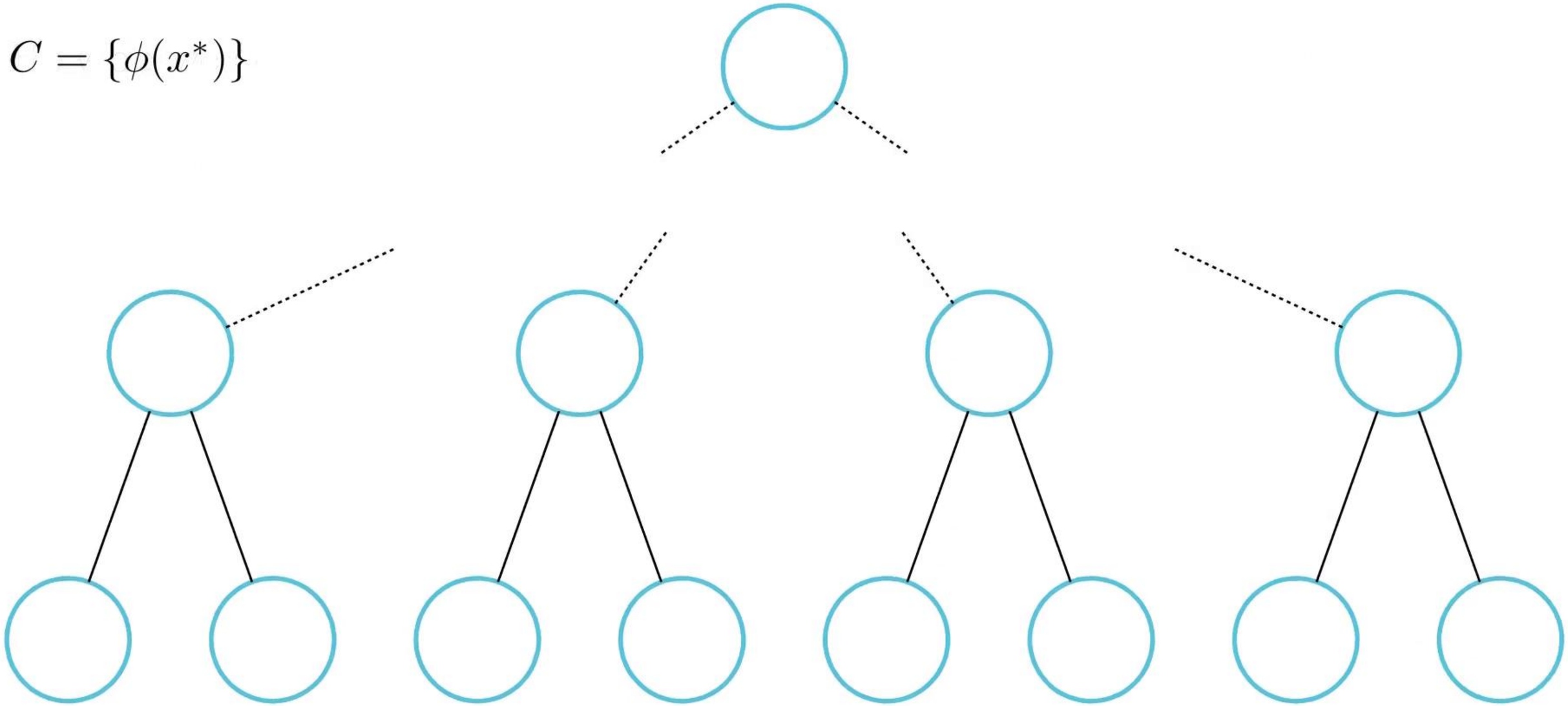
How do we sample proportional to and update $w(\cdot)\Delta(\cdot, C)$, efficiently?

Answer: Sampling trees! Sampling and updating in time $O(\log n)$.



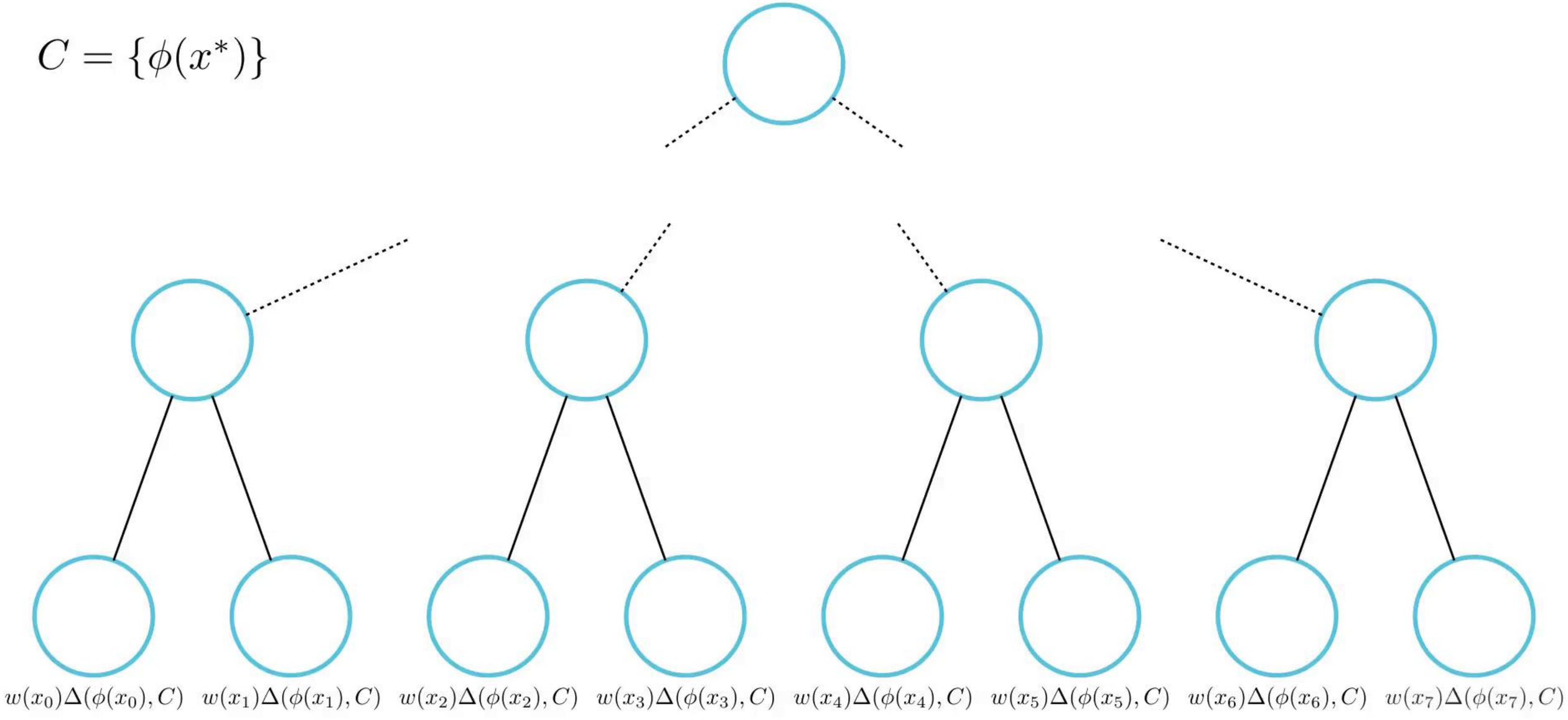
Speeding up Coreset Construction

$$C = \{\phi(x^*)\}$$



Speeding up Coreset Construction

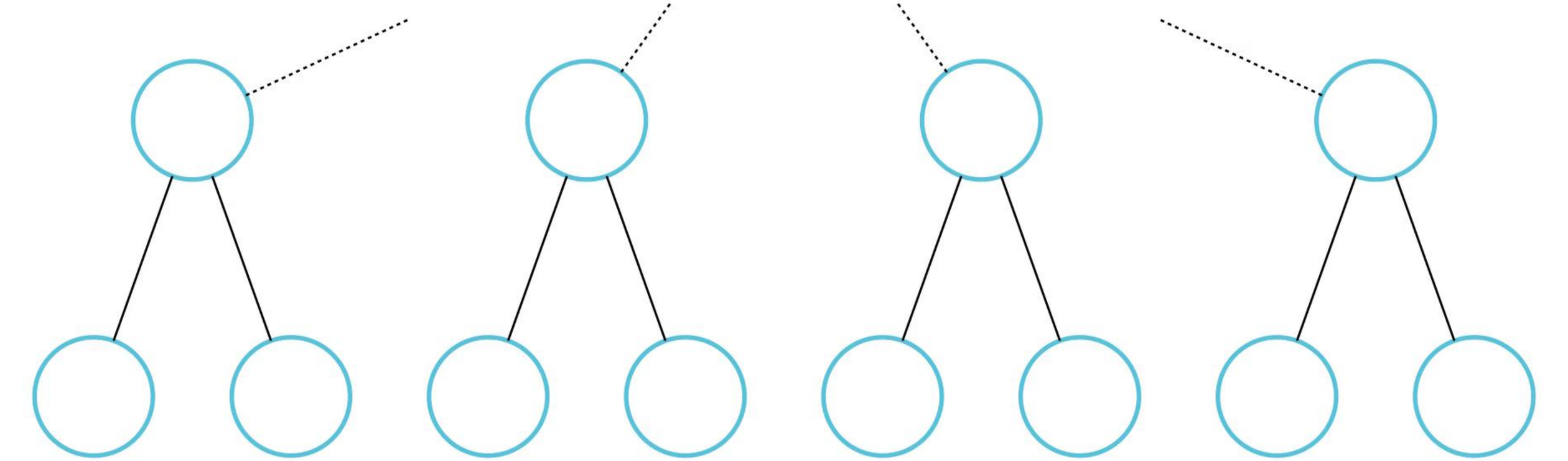
$$C = \{\phi(x^*)\}$$



Speeding up Coreset Construction

$C = \{\phi(x^*)\}$

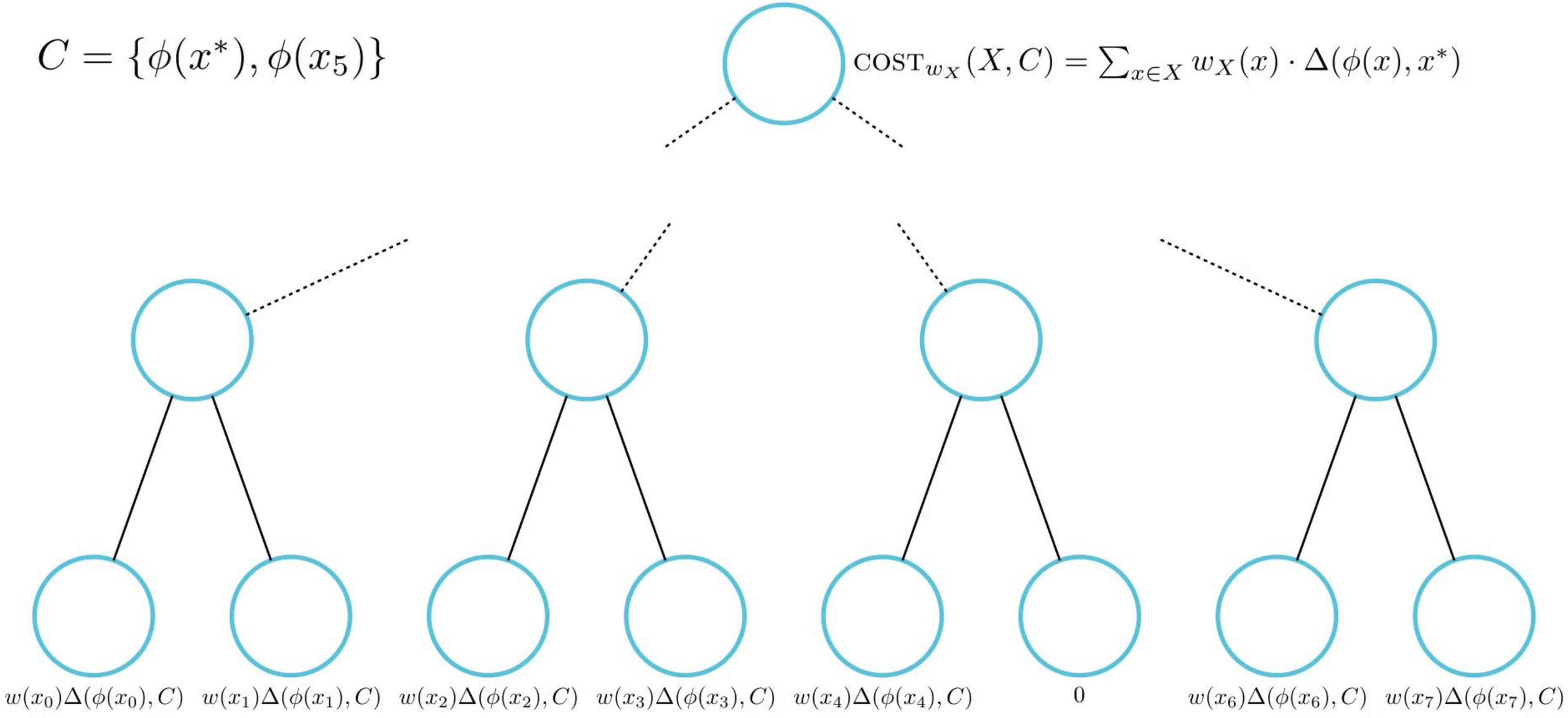
$\text{COST}_{w_X}(X, C) = \sum_{x \in X} w_X(x) \cdot \Delta(\phi(x), x^*)$



Speeding up Coreset Construction

$C = \{\phi(x^*), \phi(x_5)\}$

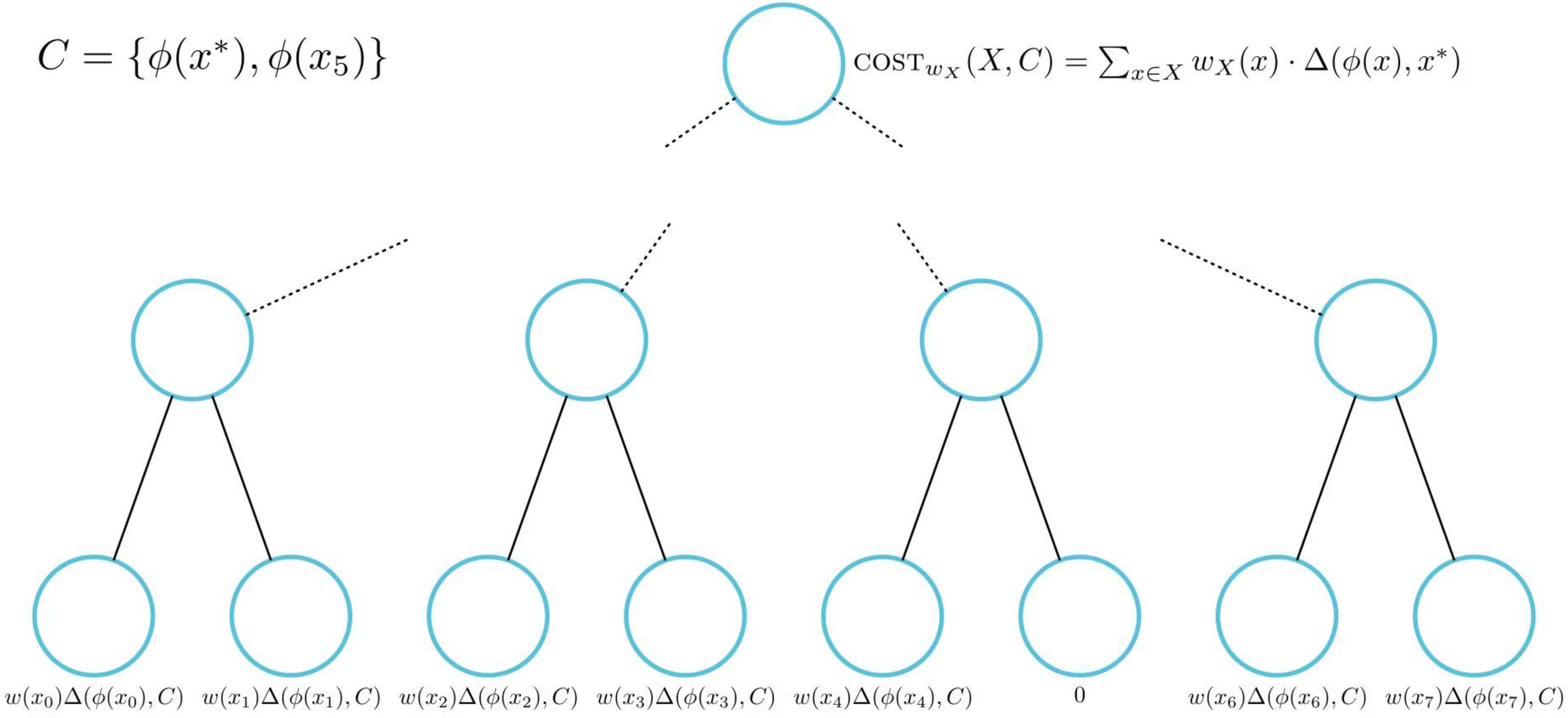
$\text{COST}_{w_X}(X, C) = \sum_{x \in X} w_X(x) \cdot \Delta(\phi(x), x^*)$



Speeding up Coreset Construction

$C = \{\phi(x^*), \phi(x_5)\}$

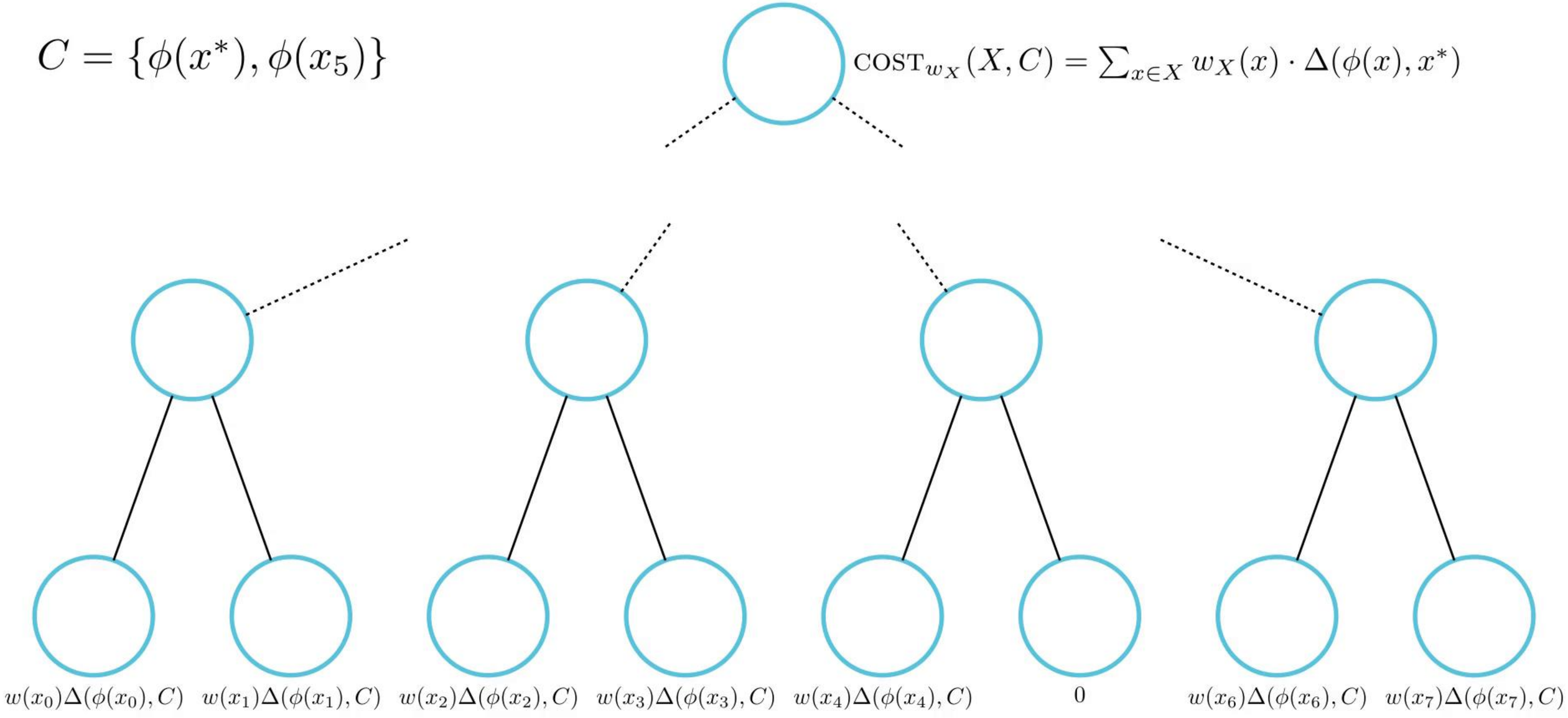
$\text{COST}_{w_X}(X, C) = \sum_{x \in X} w_X(x) \cdot \Delta(\phi(x), x^*)$



Speeding up Coreset Construction

$C = \{\phi(x^*), \phi(x_5)\}$

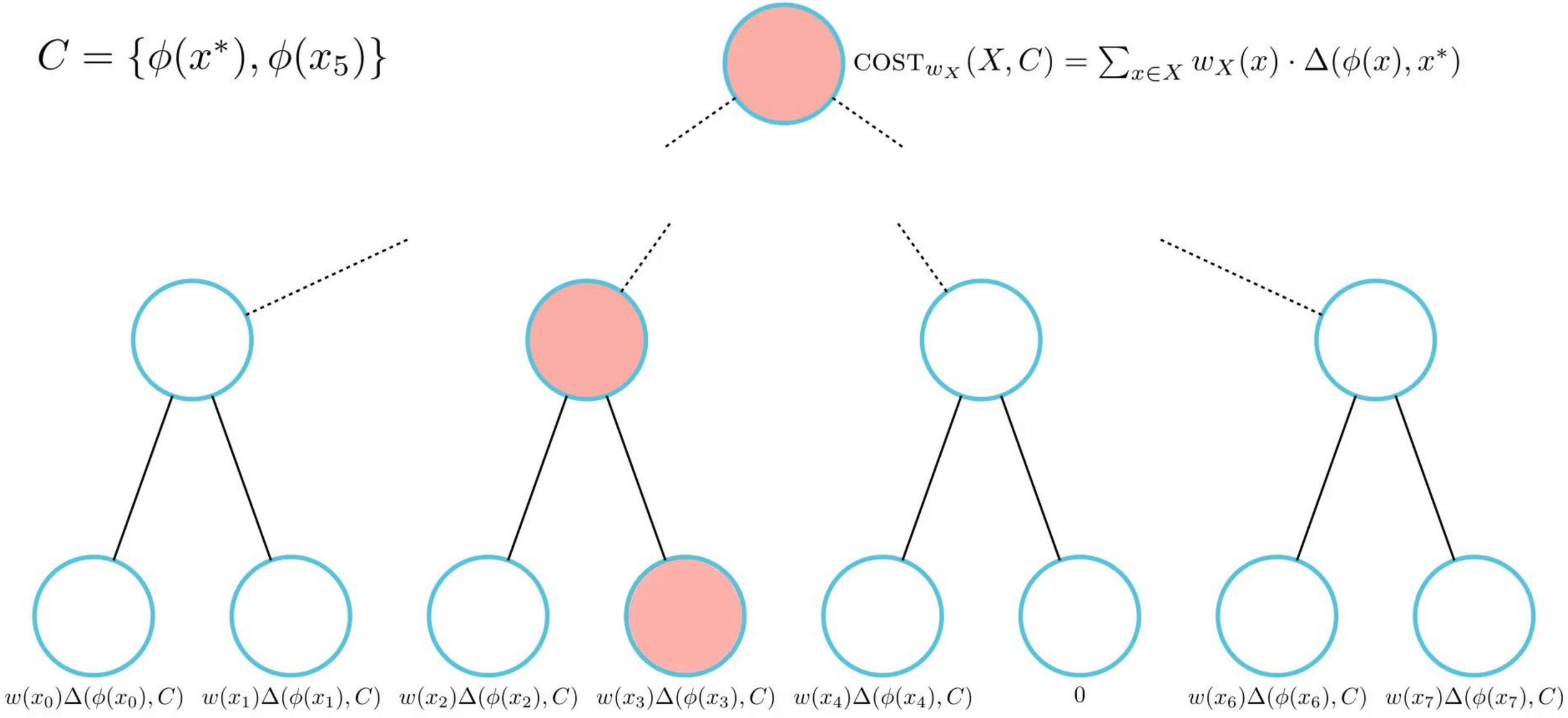
$\text{COST}_{w_X}(X, C) = \sum_{x \in X} w_X(x) \cdot \Delta(\phi(x), x^*)$



Speeding up Coreset Construction

$C = \{\phi(x^*), \phi(x_5)\}$

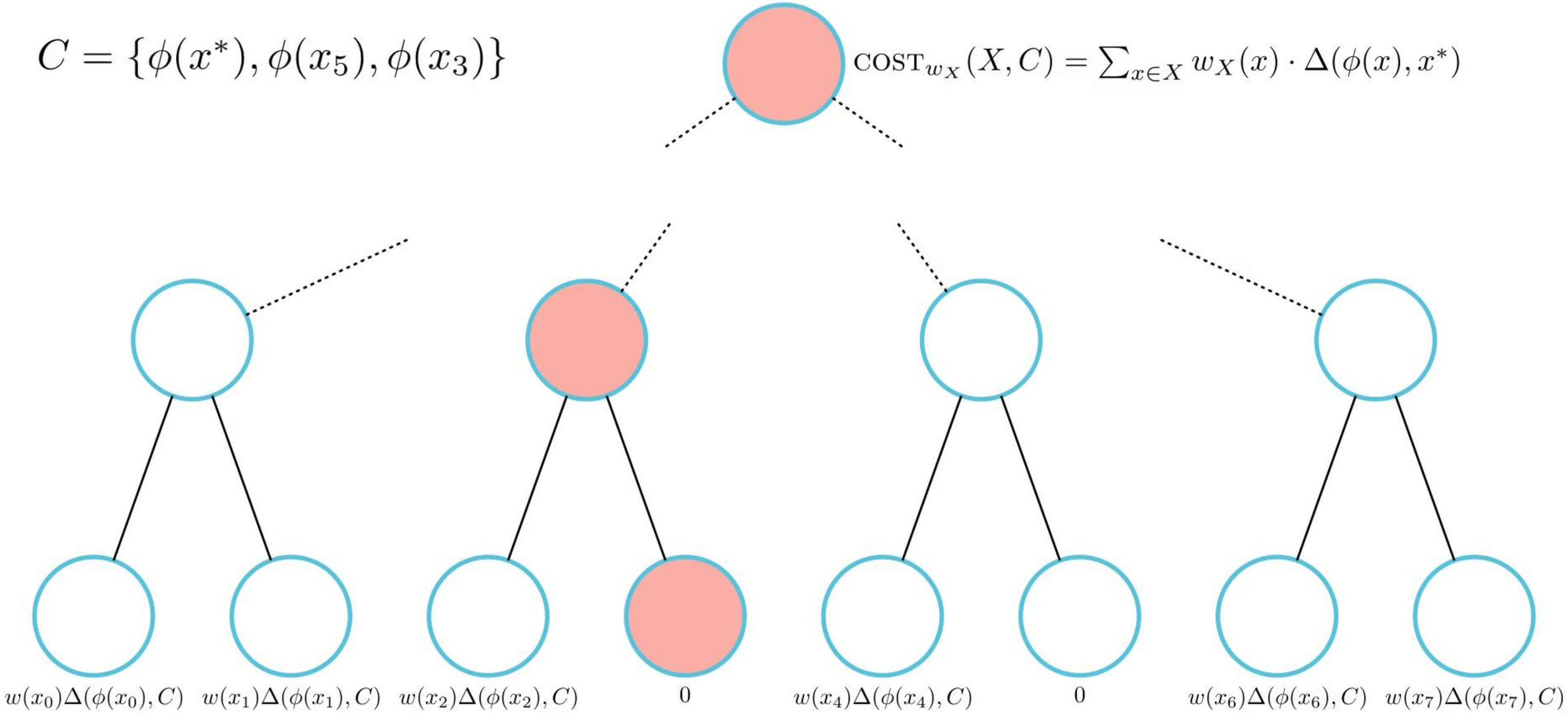
$\text{COST}_{w_X}(X, C) = \sum_{x \in X} w_X(x) \cdot \Delta(\phi(x), x^*)$



Speeding up Coreset Construction

$$C = \{\phi(x^*), \phi(x_5), \phi(x_3)\}$$

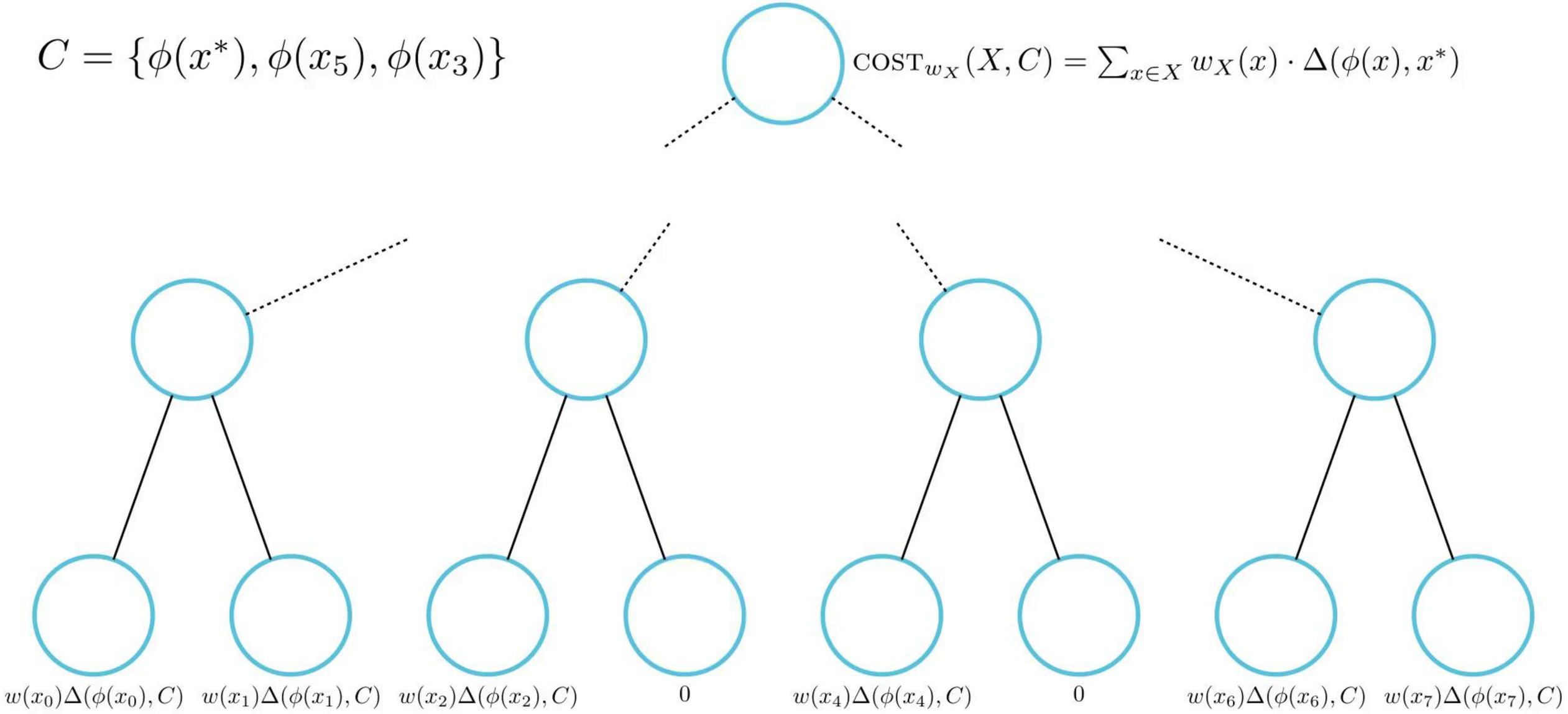
$$\text{COST}_{w_X}(X, C) = \sum_{x \in X} w_X(x) \cdot \Delta(\phi(x), x^*)$$



Speeding up Coreset Construction

$$C = \{\phi(x^*), \phi(x_5), \phi(x_3)\}$$

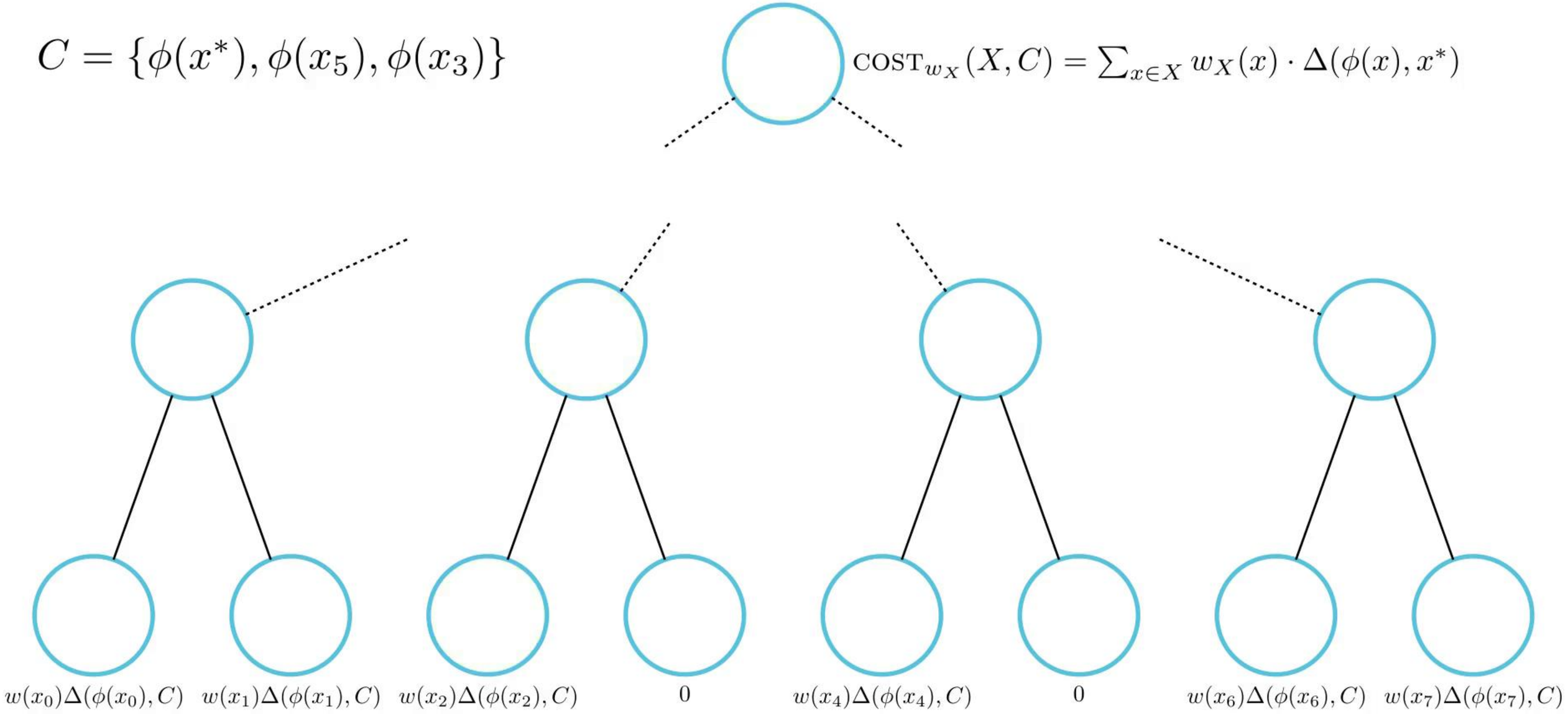
$$\text{COST}_{w_X}(X, C) = \sum_{x \in X} w_X(x) \cdot \Delta(\phi(x), x^*)$$



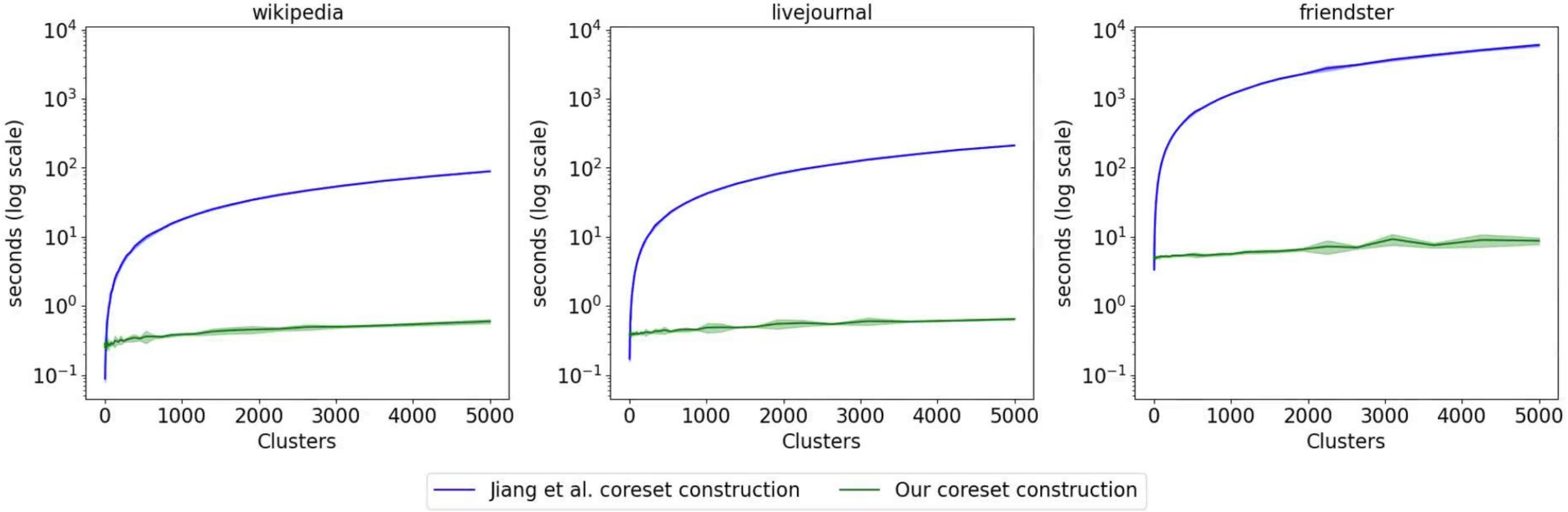
Speeding up Coreset Construction

$$C = \{\phi(x^*), \phi(x_5), \phi(x_3)\}$$

$$\text{COST}_{w_X}(X, C) = \sum_{x \in X} w_X(x) \cdot \Delta(\phi(x), x^*)$$

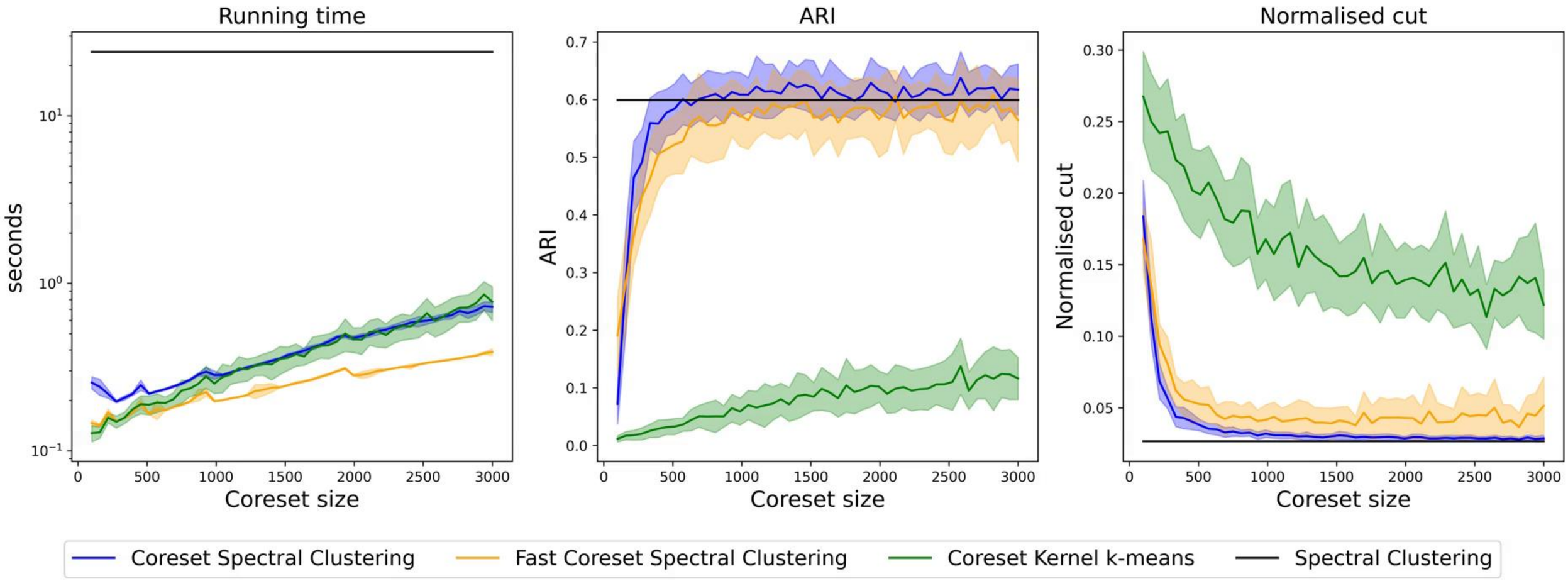


Experiments



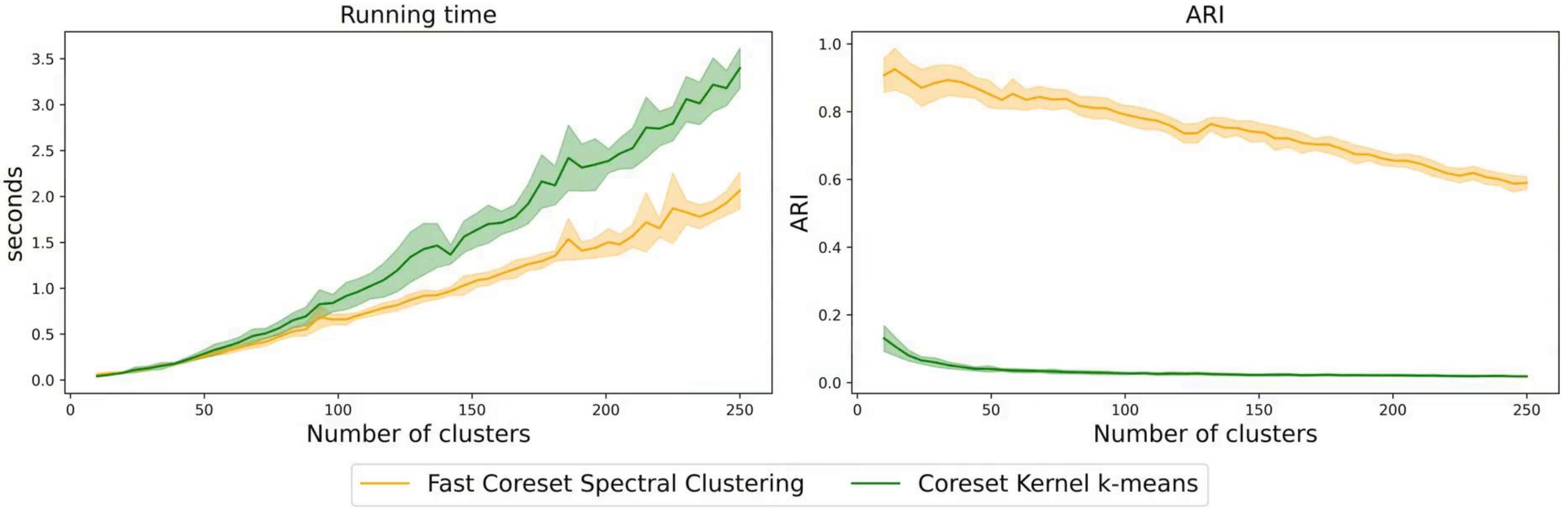
Comparison of running times for coreset construction
using either Jiang et al.'s approach for kernel kmeans++ or our method

Experiments



Results on Pendigits 250-nearest neighbour graph

Experiments



Results on synthetic stochastic block model with a coreset size of 1%
($p = 1/2$, $q = 1/(1000k)$, 1000 nodes per cluster)

Code

Install with `pip install coreset-sc`

```python
from coreset_sc import CoresetSpectralClustering

csc = CoresetSpectralClustering(
    num_clusters=5, coreset_ratio=0.1
)
# A is a sparse scipy CSR matrix of a
# symmetric adjacency matrix
pred_labels = csc.fit_predict(A)
```

https://github.com/BenJourdan/coreset-sc

Code

Install with `pip install coreset-sc`

```python
from coreset_sc import CoresetSpectralClustering

csc = CoresetSpectralClustering(
    num_clusters=5, coreset_ratio=0.1
)
# A is a sparse scipy CSR matrix of a
# symmetric adjacency matrix
pred_labels = csc.fit_predict(A)
```

https://github.com/BenJourdan/coreset-sc

The Connection (Dhillon et al. 2004)

"Normalised Cut" Problem

$$\begin{aligned} \min \quad & \operatorname{tr}(D^{-1}A) - \operatorname{tr}(Z^T D^{-\frac{1}{2}} A D^{-\frac{1}{2}} Z) \\ \text{s.t.} \quad & \mathcal{X} \in \{0,1\}^{n \times k}, \\ & \mathcal{X} 1_k = 1_n, \\ & Z = D^{\frac{1}{2}} \mathcal{X} (\mathcal{X}^T D \mathcal{X})^{-\frac{1}{2}} \end{aligned}$$

Weighted Kernel K-Means Problem

$$\begin{aligned} \min \quad & \operatorname{tr}(WK) - \operatorname{tr}(Y^T W^{\frac{1}{2}} K W^{\frac{1}{2}} Y) \\ \text{s.t.} \quad & \mathcal{X} \in \{0,1\}^{n \times k}, \\ & \mathcal{X} 1_k = 1_n, \\ & Y = W^{\frac{1}{2}} \mathcal{X} (\mathcal{X}^T W \mathcal{X})^{-\frac{1}{2}} \end{aligned}$$

The Connection (Dhillon et al. 2004)

"Normalised Cut" Problem

$$\begin{aligned} \min \quad & \operatorname{tr}(D^{-1}A) - \operatorname{tr}(Z^T D^{-\frac{1}{2}} A D^{-\frac{1}{2}} Z) \\ \text{s.t.} \quad & \mathcal{X} \in \{0,1\}^{n \times k}, \\ & \mathcal{X} 1_k = 1_n, \\ & Z = D^{\frac{1}{2}} \mathcal{X} (\mathcal{X}^T D \mathcal{X})^{-\frac{1}{2}} \end{aligned}$$

Weighted Kernel K-Means Problem

$$\begin{aligned} \min \quad & \operatorname{tr}(WK) - \operatorname{tr}(Y^T W^{\frac{1}{2}} K W^{\frac{1}{2}} Y) \\ \text{s.t.} \quad & \mathcal{X} \in \{0,1\}^{n \times k}, \\ & \mathcal{X} 1_k = 1_n, \\ & Y = W^{\frac{1}{2}} \mathcal{X} (\mathcal{X}^T W \mathcal{X})^{-\frac{1}{2}} \end{aligned}$$