Heat Kernels in Graphs:

A Journey from Random Walks to Geometry, and Back

He Sun

University of Bristol

Notation

Let G be an undirected d-regular graph with n vertices.

Notation

Let G be an undirected d-regular graph with n vertices.

Laplacian Matrix

The normalised Laplacian matrix of G is defined by

$$\mathcal{L} \triangleq \mathbf{I} - \frac{1}{d} \cdot \mathbf{A},$$

where $\mathbf{A}$ is the adjacency matrix of G.

Notation

Let G be an undirected d-regular graph with n vertices.

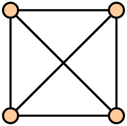
Laplacian Matrix

The normalised Laplacian matrix of G is defined by

$$\mathcal{L} \triangleq \mathbf{I} - \frac{1}{d} \cdot \mathbf{A},$$

where $\mathbf{A}$ is the adjacency matrix of G.

Example:



$$\mathcal{L}_G = \begin{pmatrix} 1 & -1/3 & -1/3 & -1/3 \\ -1/3 & 1 & -1/3 & -1/3 \\ -1/3 & -1/3 & 1 & -1/3 \\ -1/3 & -1/3 & -1/3 & 1 \end{pmatrix}$$

Notation

Let G be an undirected d-regular graph with n vertices.

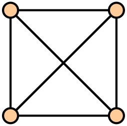
Laplacian Matrix

The normalised Laplacian matrix of G is defined by

$$\mathcal{L} \triangleq \mathbf{I} - \frac{1}{d} \cdot \mathbf{A},$$

where $\mathbf{A}$ is the adjacency matrix of G.

Example:



$$\mathcal{L}_G = \begin{pmatrix} 1 & -1/3 & -1/3 & -1/3 \\ -1/3 & 1 & -1/3 & -1/3 \\ -1/3 & -1/3 & 1 & -1/3 \\ -1/3 & -1/3 & -1/3 & 1 \end{pmatrix}$$

Matrix $\mathcal{L}$ has eigenvalues $0 = \lambda_1 \leq \ldots \leq \lambda_n$ with corresponding eigenvectors

$$f_1, \ldots, f_n.$$

Heat Kernel: a Fundamental Solution of a PDE

Let $\mathcal{M}$ be a compact Riemannian manifold, and

$$u : \mathcal{M} \times [0, \infty) \to \mathbb{R}$$

be a smooth function describing the temperature at a point in $\mathcal{M}$ and time t.

Heat Kernel: a Fundamental Solution of a PDE

Let $\mathcal{M}$ be a compact Riemannian manifold, and

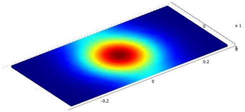
$$u : \mathcal{M} \times [0, \infty) \to \mathbb{R}$$

be a smooth function describing the temperature at a point in $\mathcal{M}$ and time t.

Heat Kernel

Let $\mathcal{M}$ be a compact Riemannian manifold and Δ the Laplacian operator. Then the heat kernel is the fundament solution of the following PDE:

$$\frac{\partial u}{\partial t} + \Delta u = 0.$$



Heat Kernel: a Fundamental Solution of a PDE

Let $\mathcal{M}$ be a compact Riemannian manifold, and

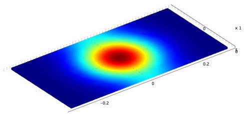
$$u : \mathcal{M} \times [0, \infty) \to \mathbb{R}$$

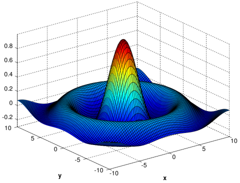
be a smooth function describing the temperature at a point in $\mathcal{M}$ and time t.

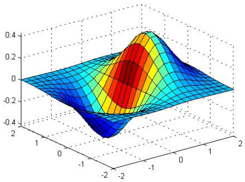
Heat Kernel

Let $\mathcal{M}$ be a compact Riemannian manifold and Δ the Laplacian operator. Then the heat kernel is the fundament solution of the following PDE:

$$\frac{\partial u}{\partial t} + \Delta u = 0.$$







Heat Kernel Defines a Continuous-Time Random Walk

Heat Kernel in Graphs

When Δ is the Laplacian matrix $\mathcal{L}$ of graph G, for any $t \geq 0$ the heat kernel of G can be written as

$$\mathbf{H}_t = \mathrm{e}^{-t\mathcal{L}} = \sum_{k=0}^{\infty} \frac{t^k \mathrm{e}^{-t}}{k!} \mathbf{P}^k,$$

where $\mathbf{P}$ is the random walk matrix of G.

Heat Kernel Defines a Continuous-Time Random Walk

Heat Kernel in Graphs

When Δ is the Laplacian matrix $\mathcal{L}$ of graph G, for any $t \geq 0$ the heat kernel of G can be written as

$$\mathbf{H}_t = \mathrm{e}^{-t\mathcal{L}} = \sum_{k=0}^{\infty} \frac{t^k \mathrm{e}^{-t}}{k!} \mathbf{P}^k,$$

where $\mathbf{P}$ is the random walk matrix of G.

Heat kernel defines a continuous-time random walk:

- Vertices choose a neighbour according to $\mathbf{P}$;
- Jumps occur after $\mathsf{Poison}(1)$ waiting times.

Heat Kernel Defines a Continuous-Time Random Walk

Heat Kernel in Graphs

When Δ is the Laplacian matrix $\mathcal{L}$ of graph G, for any $t \geq 0$ the heat kernel of G can be written as

$$\mathbf{H}_t = \mathrm{e}^{-t\mathcal{L}} = \sum_{k=0}^{\infty} \frac{t^k \mathrm{e}^{-t}}{k!} \mathbf{P}^k,$$

where $\mathbf{P}$ is the random walk matrix of G.

Heat kernel defines a continuous-time random walk:

- Vertices choose a neighbour according to $\mathbf{P}$;
- Jumps occur after $\mathrm{Poison}(1)$ waiting times.

Continuous-time Random Walks $\approx$ Discrete-time Random Walks!

Heat Kernel Defines a Continuous-Time Random Walk

Heat Kernel in Graphs

When Δ is the Laplacian matrix $\mathcal{L}$ of graph G, for any $t \geq 0$ the heat kernel of G can be written as

$$\mathbf{H}_t = \mathrm{e}^{-t\mathcal{L}} = \sum_{k=0}^{\infty} \frac{t^k \mathrm{e}^{-t}}{k!} \mathbf{P}^k,$$

where $\mathbf{P}$ is the random walk matrix of G.

Heat kernel defines a continuous-time random walk:

- Vertices choose a neighbour according to $\mathbf{P}$;
- Jumps occur after Poison(1) waiting times.

Continuous-time Random Walks $\approx$ Discrete-time Random Walks!

The heat kernel defines a semi-group, i.e.,

$$\mathbf{H}_{t+s} = \mathbf{H}_t \cdot \mathbf{H}_s, \forall t, s \geq 0 \qquad \text{and} \qquad \lim_{t \to 0} \mathbf{H}_t = \mathbf{I}.$$

Heat Kernel in Graphs: Towards a Geometric Interpretation

For any time-step $t \geq 0$, define an embedding $\psi_t : V \mapsto \mathbb{R}^n$ by

$$\psi_t(v) = \left(\mathrm{e}^{-t\lambda_1} f_1(v), \mathrm{e}^{-t\lambda_2} f_2(v), \ldots, \mathrm{e}^{-t\lambda_n} f_n(v) \right).$$

Heat Kernel in Graphs: Towards a Geometric Interpretation

For any time-step $t \geq 0$, define an embedding $\psi_t : V \mapsto \mathbb{R}^n$ by

$$\psi_t(v) = \left(\mathrm{e}^{-t\lambda_1} f_1(v), \mathrm{e}^{-t\lambda_2} f_2(v), \ldots, \mathrm{e}^{-t\lambda_n} f_n(v) \right).$$

Let the heat kernel distance between vertices u and v be

$$d_t(u,v) = \|\psi_t(u) - \psi_t(v)\|^2.$$

Heat Kernel in Graphs: Towards a Geometric Interpretation

For any time-step $t \geq 0$, define an embedding $\psi_t : V \mapsto \mathbb{R}^n$ by

$$\psi_t(v) = \left(\mathrm{e}^{-t\lambda_1} f_1(v), \mathrm{e}^{-t\lambda_2} f_2(v), \ldots, \mathrm{e}^{-t\lambda_n} f_n(v) \right).$$

Let the heat kernel distance between vertices u and v be

$$d_t(u,v) = \|\psi_t(u) - \psi_t(v)\|^2.$$

A simple calculation shows that $d_t(u,v) = \sum_{w \in V} \left(\mathbf{H}_t(w,u) - \mathbf{H}_t(w,v) \right)^2$.

Heat Kernel Distance: From Geometry to Random Walks

Meaning of the heat kernel distance, with a proper choice of t:

$$d_t(u,v) = \sum_{w \in V} \left(\mathbf{H}_t(w,u) - \mathbf{H}_t(w,v)\right)^2$$

Heat Kernel Distance: From Geometry to Random Walks

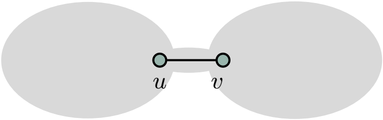
Meaning of the heat kernel distance, with a proper choice of t:

$$d_t(u,v) = \sum_{w \in V} \left(\mathbf{H}_t(w,u) - \mathbf{H}_t(w,v)\right)^2$$

Heat Kernel Distance: From Geometry to Random Walks

Meaning of the heat kernel distance, with a proper choice of t:

$$d_t(u,v) = \sum_{w \in V} \left(\mathbf{H}_t(w,u) - \mathbf{H}_t(w,v)\right)^2$$

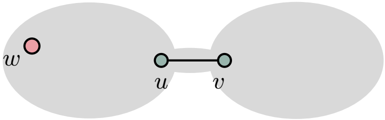


edge $\{u,v\}$ is along a sparse cut

Heat Kernel Distance: From Geometry to Random Walks

Meaning of the heat kernel distance, with a proper choice of t:

$$d_t(u,v) = \sum_{w \in V} \left(\mathbf{H}_t(w,u) - \mathbf{H}_t(w,v)\right)^2$$

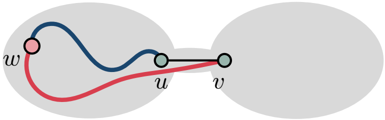


edge $\{u,v\}$ is along a sparse cut

Heat Kernel Distance: From Geometry to Random Walks

Meaning of the heat kernel distance, with a proper choice of t:

$$d_t(u,v) = \sum_{w \in V} \left(\mathbf{H}_t(w,u) - \mathbf{H}_t(w,v)\right)^2$$



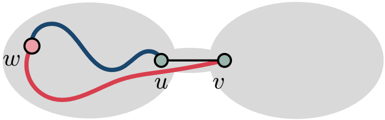
edge $\{u,v\}$ is along a sparse cut

- One of the two walks needs to go across a sparse cut.

Heat Kernel Distance: From Geometry to Random Walks

Meaning of the heat kernel distance, with a proper choice of t:

$$d_t(u,v) = \sum_{w \in V} \left(\mathbf{H}_t(w,u) - \mathbf{H}_t(w,v)\right)^2$$



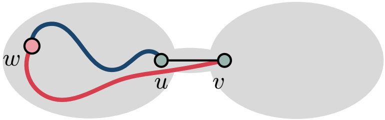
edge $\{u,v\}$ is along a sparse cut

- One of the two walks needs to go across a sparse cut.
- For any vertex w, the value of $\left(\mathbf{H}_t(w,u) - \mathbf{H}_t(w,v)\right)^2$ is big.

Heat Kernel Distance: From Geometry to Random Walks

Meaning of the heat kernel distance, with a proper choice of t:

$$d_t(u, v) = \sum_{w \in V} \left(\mathbf{H}_t(w, u) - \mathbf{H}_t(w, v)\right)^2$$



edge $\{u, v\}$ is along a sparse cut

- One of the two walks needs to go across a sparse cut.
- For any vertex w, the value of $\left(\mathbf{H}_t(w, u) - \mathbf{H}_t(w, v)\right)^2$ is big.
- Hence, $d_t(u, v)$ is big.

Heat Kernel Distance: From Geometry to Random Walks

Meaning of the heat kernel distance, with a proper choice of t:

$$d_t(u,v) = \sum_{w \in V} \left(\mathbf{H}_t(w,u) - \mathbf{H}_t(w,v)\right)^2$$

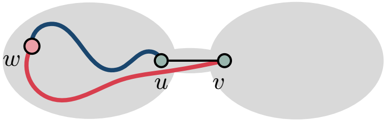
edge $\{u,v\}$ is along a sparse cut

- One of the two walks needs to go across a sparse cut.
- For any vertex w, the value of $\left(\mathbf{H}_t(w,u) - \mathbf{H}_t(w,v)\right)^2$ is big.
- Hence, $d_t(u,v)$ is big.

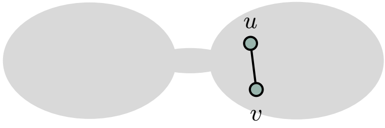
Heat Kernel Distance: From Geometry to Random Walks

Meaning of the heat kernel distance, with a proper choice of t:

$$d_t(u,v) = \sum_{w \in V} \left(\mathbf{H}_t(w,u) - \mathbf{H}_t(w,v)\right)^2$$



edge $\{u,v\}$ is along a sparse cut



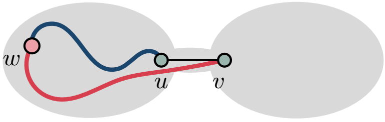
edge $\{u,v\}$ is at one side a sparse cut

- One of the two walks needs to go across a sparse cut.
- For any vertex w, the value of $\left(\mathbf{H}_t(w,u) - \mathbf{H}_t(w,v)\right)^2$ is big.
- Hence, $d_t(u,v)$ is big.

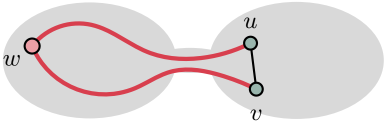
Heat Kernel Distance: From Geometry to Random Walks

Meaning of the heat kernel distance, with a proper choice of t:

$$d_t(u, v) = \sum_{w \in V} \left(\mathbf{H}_t(w, u) - \mathbf{H}_t(w, v)\right)^2$$



edge $\{u, v\}$ is along a sparse cut



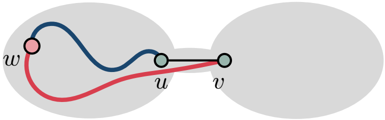
edge $\{u, v\}$ is at one side a sparse cut

- One of the two walks needs to go across a sparse cut.
- For any vertex w, the value of $\left(\mathbf{H}_t(w, u) - \mathbf{H}_t(w, v)\right)^2$ is big.
- Hence, $d_t(u, v)$ is big.

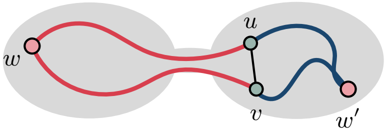
Heat Kernel Distance: From Geometry to Random Walks

Meaning of the heat kernel distance, with a proper choice of t:

$$d_t(u,v) = \sum_{w \in V} \left(\mathbf{H}_t(w,u) - \mathbf{H}_t(w,v)\right)^2$$



edge $\{u,v\}$ is along a sparse cut



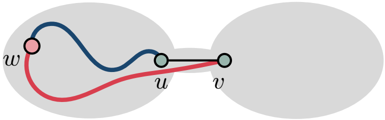
edge $\{u,v\}$ is at one side a sparse cut

- One of the two walks needs to go across a sparse cut.
- For any vertex w, the value of $\left(\mathbf{H}_t(w,u) - \mathbf{H}_t(w,v)\right)^2$ is big.
- Hence, $d_t(u,v)$ is big.

Heat Kernel Distance: From Geometry to Random Walks

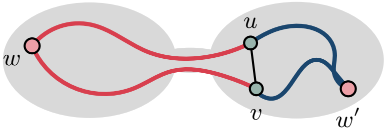
Meaning of the heat kernel distance, with a proper choice of t:

$$d_t(u,v) = \sum_{w \in V} \left(\mathbf{H}_t(w,u) - \mathbf{H}_t(w,v)\right)^2$$



edge $\{u,v\}$ is along a sparse cut

- One of the two walks needs to go across a sparse cut.
- For any vertex w, the value of $\left(\mathbf{H}_t(w,u) - \mathbf{H}_t(w,v)\right)^2$ is big.
- Hence, $d_t(u,v)$ is big.



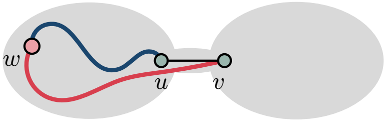
edge $\{u,v\}$ is at one side a sparse cut

- The values of two $\mathbf{H}_t(w,.)$s are close to each other.

Heat Kernel Distance: From Geometry to Random Walks

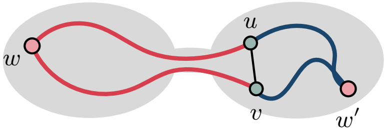
Meaning of the heat kernel distance, with a proper choice of t:

$$d_t(u,v) = \sum_{w \in V} \left(\mathbf{H}_t(w,u) - \mathbf{H}_t(w,v)\right)^2$$



edge $\{u,v\}$ is along a sparse cut

- One of the two walks needs to go across a sparse cut.
- For any vertex w, the value of $\left(\mathbf{H}_t(w,u) - \mathbf{H}_t(w,v)\right)^2$ is big.
- Hence, $d_t(u,v)$ is big.



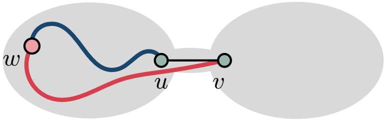
edge $\{u,v\}$ is at one side a sparse cut

- The values of two $\mathbf{H}_t(w,.)$s are close to each other.
- Hence, $\left(\mathbf{H}_t(w,u) - \mathbf{H}_t(w,v)\right)^2$ is small for any vertex w.

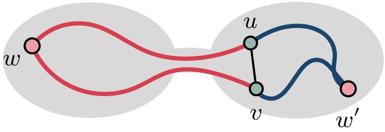
Heat Kernel Distance: From Geometry to Random Walks

Meaning of the heat kernel distance, with a proper choice of t:

$$d_t(u,v) = \sum_{w \in V} \left(\mathbf{H}_t(w,u) - \mathbf{H}_t(w,v)\right)^2$$



edge $\{u,v\}$ is along a sparse cut



edge $\{u,v\}$ is at one side a sparse cut

- One of the two walks needs to go across a sparse cut.
- For any vertex w, the value of $\left(\mathbf{H}_t(w,u) - \mathbf{H}_t(w,v)\right)^2$ is big.
- Hence, $d_t(u,v)$ is big.

- The values of two $\mathbf{H}_t(w,.)$s are close to each other.
- Hence, $\left(\mathbf{H}_t(w,u) - \mathbf{H}_t(w,v)\right)^2$ is small for any vertex w.
- Hence, $d_t(u,v)$ is small.

Key Questions

- Are our intuitions based on random walks correct?

Key Questions

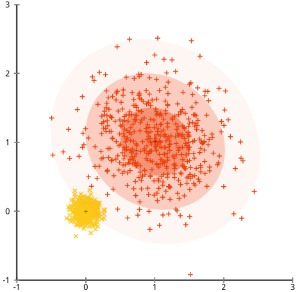
- Are our intuitions based on random walks correct?
- How do we apply these intuitions to design algorithms?

Key Questions

- Are our intuitions based on random walks correct?
- How do we apply these intuitions to design algorithms?
- Do PDEs lead to an entirely new technique to design algorithms for large datasets?

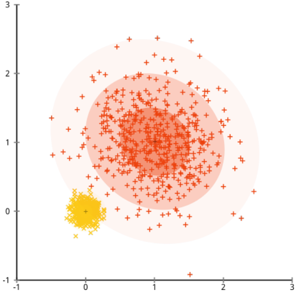
Graph Clustering

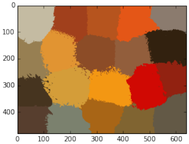
Applications in clustering:



Graph Clustering

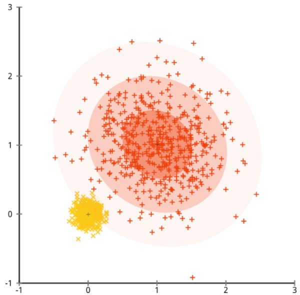
Applications in clustering:

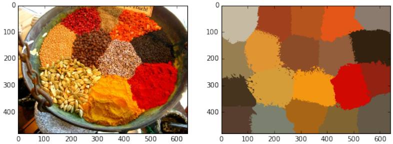


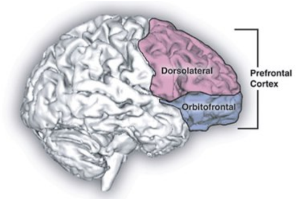


Graph Clustering

Applications in clustering:







Graph Conductance

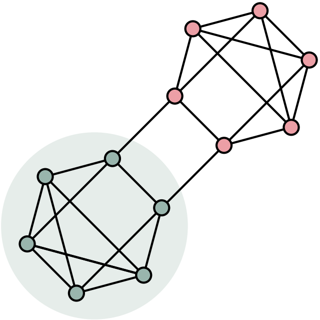
The conductance of a set S is defined by

$$\phi_G(S) \triangleq \frac{|E(S, V \setminus S)|}{d \cdot |S|}.$$

Graph Conductance

The conductance of a set S is defined by

$$\phi_G(S) \triangleq \frac{|E(S, V \setminus S)|}{d \cdot |S|}.$$



$$\phi_G(S) = \frac{2}{4 \cdot 6} = \frac{1}{12}$$

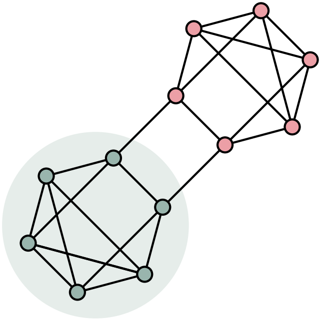
Graph Conductance

The conductance of a set S is defined by

$$\phi_G(S) \triangleq \frac{|E(S, V \setminus S)|}{d \cdot |S|}.$$

The conductance of a graph G is defined by

$$\phi_G \triangleq \min_{S:|S| \leq |V|/2} \phi_G(S).$$



$$\phi_G(S) = \frac{2}{4 \cdot 6} = \frac{1}{12}$$

Graph Conductance

The conductance of a set S is defined by

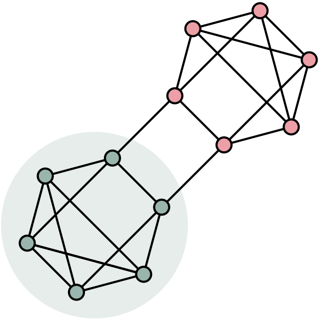
$$\phi_G(S) \triangleq \frac{|E(S, V \setminus S)|}{d \cdot |S|}.$$

The conductance of a graph G is defined by

$$\phi_G \triangleq \min_{S:|S|\leq|V|/2} \phi_G(S).$$

Cheeger's Inequality

$$\frac{\lambda_2}{2} \leq \phi_G \leq \sqrt{2\lambda_2}.$$

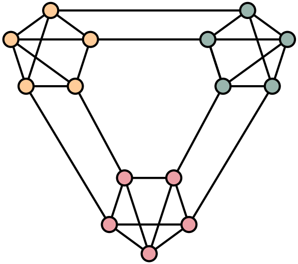


$$\phi_G(S) = \frac{2}{4\cdot 6} = \frac{1}{12}$$

k-Way Expansion

The k-way expansion constant is defined by

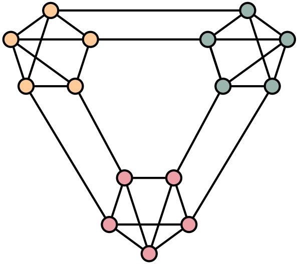
$$\rho(k) = \min_{\text{partition } A_1,\ldots,A_k} \max_{1 \leq i \leq k} \phi_G(A_i).$$



k-Way Expansion

The k-way expansion constant is defined by

$$\rho(k) = \min_{\text{partition } A_1,\ldots,A_k} \max_{1\le i\le k} \phi_G(A_i).$$



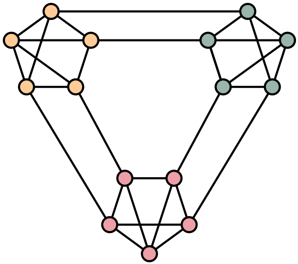
Higher-Order Cheeger's Inequality

$$\frac{\lambda_k}{2} \le \rho(k) \le O(k^3)\sqrt{\lambda_k}.$$

k-Way Expansion

The k-way expansion constant is defined by

$$\rho(k) = \min_{\text{partition } A_1,\ldots,A_k} \max_{1\leq i\leq k} \phi_G(A_i).$$



Higher-Order Cheeger's Inequality

$$\frac{\lambda_k}{2} \leq \rho(k) \leq O(k^3)\sqrt{\lambda_k}.$$

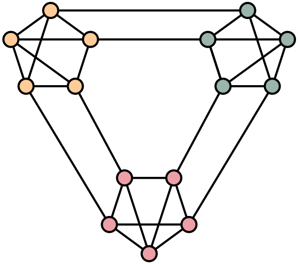
A large gap between λ_{k+1} and $\rho(k)$ implies that

- existence of a k-way partition with bounded $\rho(k)$.

k-Way Expansion

The k-way expansion constant is defined by

$$\rho(k) = \min_{\text{partition } A_1,\ldots,A_k} \max_{1\le i\le k} \phi_G(A_i).$$



Higher-Order Cheeger's Inequality

$$\frac{\lambda_k}{2} \le \rho(k) \le O(k^3)\sqrt{\lambda_k}.$$

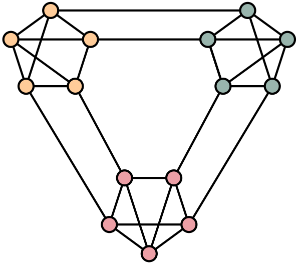
A large gap between λ_{k+1} and $\rho(k)$ implies that

- existence of a k-way partition with bounded $\rho(k)$.
- any $(k+1)$-way partition contains a set with conductance at least $\lambda_{k+1}/2$.

k-Way Expansion

The k-way expansion constant is defined by

$$\rho(k) = \min_{\text{partition } A_1,\ldots,A_k} \max_{1\leq i\leq k} \phi_G(A_i).$$



Higher-Order Cheeger's Inequality

$$\frac{\lambda_k}{2} \leq \rho(k) \leq O(k^3)\sqrt{\lambda_k}.$$

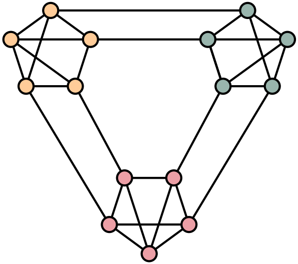
A large gap between λ_{k+1} and $\rho(k)$ implies that

- existence of a k-way partition with bounded $\rho(k)$.
- any $(k+1)$-way partition contains a set with conductance at least $\lambda_{k+1}/2$.
- Graph G has <u>exactly</u> k clusters.

k-Way Expansion

The k-way expansion constant is defined by

$$\rho(k) = \min_{\text{partition } A_1,\ldots,A_k} \max_{1\leq i\leq k} \phi_G(A_i).$$



Higher-Order Cheeger's Inequality

$$\frac{\lambda_k}{2} \leq \rho(k) \leq O(k^3)\sqrt{\lambda_k}.$$

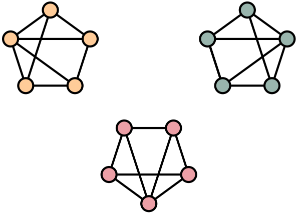
A large gap between λ_{k+1} and $\rho(k)$ implies that

- existence of a k-way partition with bounded $\rho(k)$.
- any $(k+1)$-way partition contains a set with conductance at least $\lambda_{k+1}/2$.
- Graph G has <u>exactly</u> k clusters.

The key parameter: $\Upsilon \triangleq \dfrac{\lambda_{k+1}}{\rho(k)}.$

The Structure Theorem

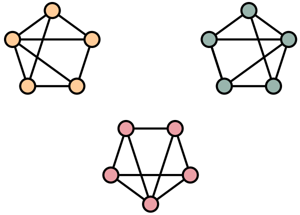
Let G be a d-regular graph with k disjoint components $S_1, \ldots, S_k$.



The Structure Theorem

Let G be a d-regular graph with k disjoint components $S_1, \ldots, S_k$. For any $1 \leq i \leq k$ let

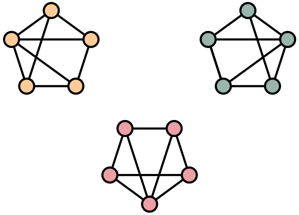
$$\chi_i(v) = \begin{cases} 1 & \text{if } v \in S_i, \\ 0 & \text{otherwise.} \end{cases}$$



The Structure Theorem

Let G be a d-regular graph with k disjoint components $S_1, \ldots, S_k$. For any $1 \leq i \leq k$ let

$$\chi_i(v) = \begin{cases} 1 & \text{if } v \in S_i, \\ 0 & \text{otherwise.} \end{cases}$$



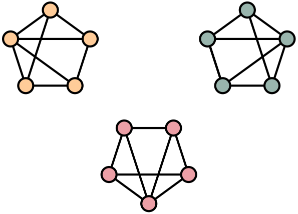
Then

$$\operatorname{span}\{f_1, \ldots, f_k\} = \operatorname{span}\{\chi_1, \ldots, \chi_k\}.$$

The Structure Theorem

Let G be a d-regular graph with k disjoint components $S_1, \ldots, S_k$. For any $1 \leq i \leq k$ let

$$\chi_i(v) = \begin{cases} 1 & \text{if } v \in S_i, \\ 0 & \text{otherwise.} \end{cases}$$



Then

$$\operatorname{span}\{f_1, \ldots, f_k\} = \operatorname{span}\{\chi_1, \ldots, \chi_k\}.$$

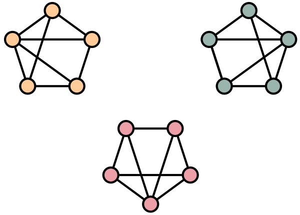
Lemma (Peng-S.-Zanetti, 2017)

$\Upsilon = \Omega(k)$ implies that $\operatorname{span}\{f_1, \ldots, f_k\} \approx \operatorname{span}\{\chi_1, \ldots, \chi_k\}$.

The Structure Theorem

Let G be a d-regular graph with k disjoint components $S_1, \ldots, S_k$. For any $1 \leq i \leq k$ let

$$\chi_i(v) = \begin{cases} 1 & \text{if } v \in S_i, \\ 0 & \text{otherwise.} \end{cases}$$



Then

$$\mathrm{span}\{f_1, \ldots, f_k\} = \mathrm{span}\{\chi_1, \ldots, \chi_k\}.$$

Lemma (Peng-S.-Zanetti, 2017)

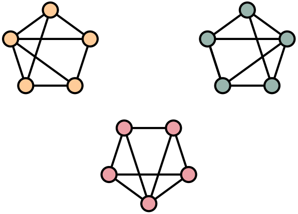
$\Upsilon = \Omega(k)$ implies that $\mathrm{span}\{f_1, \ldots, f_k\} \approx \mathrm{span}\{\chi_1, \ldots, \chi_k\}$.

Define $F(v) = (f_1(v), \ldots, f_k(v))$.

The Structure Theorem

Let G be a d-regular graph with k disjoint components $S_1, \ldots, S_k$. For any $1 \leq i \leq k$ let

$$\chi_i(v) = \begin{cases} 1 & \text{if } v \in S_i, \\ 0 & \text{otherwise.} \end{cases}$$



Then

$$\operatorname{span}\{f_1, \ldots, f_k\} = \operatorname{span}\{\chi_1, \ldots, \chi_k\}.$$

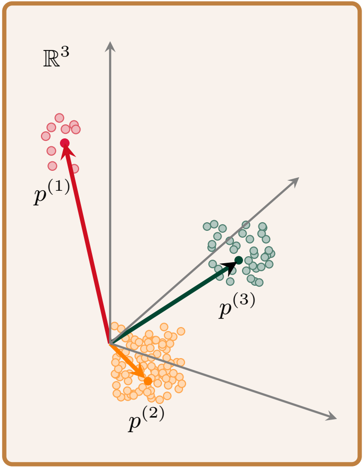
Lemma (Peng-S.-Zanetti, 2017)

$\Upsilon = \Omega(k)$ implies that $\operatorname{span}\{f_1, \ldots, f_k\} \approx \operatorname{span}\{\chi_1, \ldots, \chi_k\}$.

Define $F(v) = (f_1(v), \ldots, f_k(v))$.

There are points $p^{(1)}, \ldots, p^{(k)}$, s.t. cluster S_i is concentrated around $p^{(i)}$.

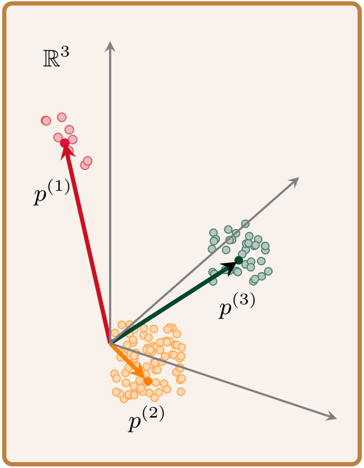
Well-Separation Property of the Embedding



$$\sum_{i=1}^{k} \sum_{u \in S_i} \left\| F(u) - p^{(i)} \right\|^2 \leq k^2/\Upsilon.$$

Points from S_i concentrate around $p^{(i)}$s.

Well-Separation Property of the Embedding



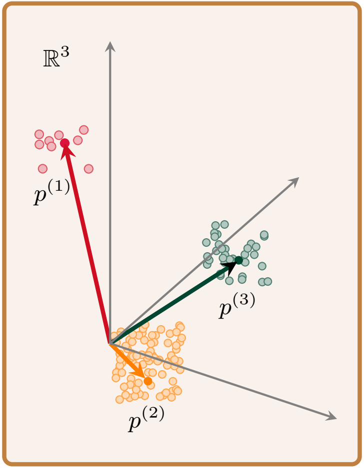
$$\sum_{i=1}^{k} \sum_{u \in S_i} \left\| F(u) - p^{(i)} \right\|^2 \leq k^2 / \Upsilon.$$

Points from S_i concentrate around $p^{(i)}s$.

$$\left\| p^{(i)} \right\|^2 \in \left(\frac{9}{10}, \frac{11}{10} \right) \cdot \frac{1}{|S_i|}$$

“Bigger” clusters are closer to the origin.

Well-Separation Property of the Embedding



$$\sum_{i=1}^{k} \sum_{u \in S_i} \left\| F(u) - p^{(i)} \right\|^2 \leq k^2/\Upsilon.$$

Points from S_i concentrate around $p^{(i)}s$.

$$\left\| p^{(i)} \right\|^2 \in \left(\frac{9}{10}, \frac{11}{10} \right) \cdot \frac{1}{|S_i|}$$

“Bigger” clusters are closer to the origin.

$$\left\| p^{(i)} - p^{(j)} \right\|^2 \geq \frac{1}{k \min\{|S_i|, |S_j|\}}$$

Distance between different clusters inversely $\approx$ the smaller cluster.

A Simple Algorithm For Graph Clustering

ASSUME we know the pairwise distances of the points for free!

A Simple Algorithm For Graph Clustering

ASSUME we know the pairwise distances of the points for free!

1. Obtain a set C of candidate centres.

Algorithm

for $i = 1$ **to** $K = \Theta(k \log k)$ **do**

  set $c_i = v$ with prob. proportional to $\|F(v)\|^2$.

return $C \triangleq \{c_1, \ldots, c_K\}$.

A Simple Algorithm For Graph Clustering

ASSUME we know the pairwise distances of the points for free!

1. Obtain a set C of candidate centres.

Algorithm

for $i = 1$ **to** $K = \Theta(k \log k)$ **do**
  set $c_i = v$ with prob. proportional to $\|F(v)\|^2$.
return $C \triangleq \{c_1, \ldots, c_K\}$.

With const. prob., each S_i has at least one vertex sampled.

A Simple Algorithm For Graph Clustering

ASSUME we know the pairwise distances of the points for free!

1. Obtain a set C of candidate centres.

Algorithm

for $i = 1$ **to** $K = \Theta(k \log k)$ **do**
  set $c_i = v$ with prob. proportional to $\|F(v)\|^2$.
return $C \triangleq \{c_1, \ldots, c_K\}$.

With const. prob., each S_i has at least one vertex sampled.

2. Delete points in C "close" to each other, until $|C| = k$.

A Simple Algorithm For Graph Clustering

ASSUME we know the pairwise distances of the points for free!

1. Obtain a set C of candidate centres.

Algorithm

for $i = 1$ **to** $K = \Theta(k \log k)$ **do**
  set $c_i = v$ with prob. proportional to $\|F(v)\|^2$.
return $C \triangleq \{c_1, \ldots, c_K\}$.

With const. prob., each S_i has at least one vertex sampled.

2. Delete points in C “close” to each other, until $|C| = k$.

With const. prob., each S_i has exactly one vertex remaining in C.

A Simple Algorithm For Graph Clustering

ASSUME we know the pairwise distances of the points for free!

1. Obtain a set C of candidate centres.

Algorithm

for $i = 1$ **to** $K = \Theta(k \log k)$ **do**
  set $c_i = v$ with prob. proportional to $\|F(v)\|^2$.
return $C \triangleq \{c_1, \ldots, c_K\}$.

With const. prob., each S_i has at least one vertex sampled.

2. Delete points in C “close” to each other, until $|C| = k$.

With const. prob., each S_i has exactly one vertex remaining in C.

3. The other $n - k$ vertices find their closest neighbours in C.

A Simple Algorithm For Graph Clustering

ASSUME we know the pairwise distances of the points for free!

1. Obtain a set C of candidate centres.

Algorithm

for $i = 1$ **to** $K = \Theta(k \log k)$ **do**
  set $c_i = v$ with prob. proportional to $\|F(v)\|^2$.
return $C \triangleq \{c_1, \ldots, c_K\}$.

With const. prob., each S_i has at least one vertex sampled.

2. Delete points in C “close” to each other, until $|C| = k$.

With const. prob., each S_i has exactly one vertex remaining in C.

3. The other $n - k$ vertices find their closest neighbours in C.

apply approximate nearest neighbour data structures.

A Simple Algorithm For Graph Clustering

ASSUME we know the pairwise distances of the points for free!

1. Obtain a set C of candidate centres.

Algorithm

for $i = 1$ **to** $K = \Theta(k \log k)$ **do**
  set $c_i = v$ with prob. proportional to $\|F(v)\|^2$.
return $C \triangleq \{c_1, \ldots, c_K\}$.

With const. prob., each S_i has at least one vertex sampled.

2. Delete points in C "close" to each other, until $|C| = k$.

With const. prob., each S_i has exactly one vertex remaining in C.

3. The other $n - k$ vertices find their closest neighbours in C.

apply approximate nearest neighbour data structures.

Runtime is $O(n \cdot \operatorname{poly} \log n)$, even for a large value of k!

Obtaining the Pairwise Distances via Heat Kernels

Recall the two embeddings discussed so far:

- $F(v) = (f_1(v), \ldots, f_k(v))$
- $\psi_t(v) = \left(\mathrm{e}^{-t\lambda_1} f_1(v), \ldots, \mathrm{e}^{-t\lambda_n} f_n(v)\right)$

Obtaining the Pairwise Distances via Heat Kernels

Recall the two embeddings discussed so far:

- $F(v) = (f_1(v), \ldots, f_k(v))$
- $\psi_t(v) = \left(\mathrm{e}^{-t\lambda_1} f_1(v), \ldots, \mathrm{e}^{-t\lambda_n} f_n(v)\right)$

Lemma (Peng-S.-Zanetti, 2017)

We can compute in $O\left(nd \cdot \log^{10} n\right)$ time an embedding such that, with hight probability, it holds that

$$(1-\varepsilon)\|F(u)-F(v)\|^2 \leq \|\psi_t(u)-\psi_t(v)\|^2 \leq \|F(u)-F(v)\|^2 + n^{-10}.$$

Obtaining the Pairwise Distances via Heat Kernels

Recall the two embeddings discussed so far:

- $F(v) = (f_1(v), \ldots, f_k(v))$
- $\psi_t(v) = \left(\mathrm{e}^{-t\lambda_1} f_1(v), \ldots, \mathrm{e}^{-t\lambda_n} f_n(v)\right)$

Lemma (Peng-S.-Zanetti, 2017)

We can compute in $O\left(nd \cdot \log^{10} n\right)$ time an embedding such that, with hight probability, it holds that

$$(1-\varepsilon)\|F(u) - F(v)\|^2 \leq \|\psi_t(u) - \psi_t(v)\|^2 \leq \|F(u) - F(v)\|^2 + n^{-10}.$$

Proof Sketch

- Johnson-Lindenstrauss transformation
- Algorithm for approximating matrix exponential.

Main Result

Theorem (Peng-S.-Zanetti, 2017)

There is a linear-time algorithm that, for a graph G with k clusters $S_1, \ldots, S_k$ and $\Upsilon = \Omega(k^3)$, outputs a partition $A_1, \ldots, A_k$ such that

$$|A_i \triangle S_i| = O(k^3 \cdot \Upsilon^{-1} \cdot |S_i|).$$

Main Result

Theorem (Peng-S.-Zanetti, 2017)

There is a linear-time algorithm that, for a graph G with k clusters $S_1, \ldots, S_k$ and $\Upsilon = \Omega(k^3)$, outputs a partition $A_1, \ldots, A_k$ such that

$$|A_i \triangle S_i| = O(k^3 \cdot \Upsilon^{-1} \cdot |S_i|).$$

- The heat kernel distances

$$d_t(u, v) = \sum_w (H_t(w, u) - H_t(w, v))^2$$

do behave differently among edges inside a cluster and edges crossing different clusters.

Main Result

Theorem (Peng-S.-Zanetti, 2017)

There is a linear-time algorithm that, for a graph G with k clusters $S_1, \ldots, S_k$ and $\Upsilon = \Omega(k^3)$, outputs a partition $A_1, \ldots, A_k$ such that

$$|A_i \triangle S_i| = O(k^3 \cdot \Upsilon^{-1} \cdot |S_i|).$$

- The heat kernel distances

$$d_t(u, v) = \sum_w (H_t(w, u) - H_t(w, v))^2$$

  do behave differently among edges inside a cluster and edges crossing different clusters.
- This gives us the first linear-time algorithm for graph clustering.

Main Result

Theorem (Peng-S.-Zanetti, 2017)

There is a linear-time algorithm that, for a graph G with k clusters $S_1, \ldots, S_k$ and $\Upsilon = \Omega(k^3)$, outputs a partition $A_1, \ldots, A_k$ such that

$$|A_i \triangle S_i| = O(k^3 \cdot \Upsilon^{-1} \cdot |S_i|).$$

- The heat kernel distances

$$d_t(u,v) = \sum_w (H_t(w,u) - H_t(w,v))^2$$

  do behave differently among edges inside a cluster and edges crossing different clusters.
- This gives us the first linear-time algorithm for graph clustering.
- Our intuitions are from random walk theory, but our analysis is based on geometry.

Main Result

Theorem (Peng-S.-Zanetti, 2017)

There is a linear-time algorithm that, for a graph G with k clusters $S_1, \ldots, S_k$ and $\Upsilon = \Omega(k^3)$, outputs a partition $A_1, \ldots, A_k$ such that

$$|A_i \triangle S_i| = O(k^3 \cdot \Upsilon^{-1} \cdot |S_i|).$$

- The heat kernel distances

$$d_t(u, v) = \sum_w (H_t(w, u) - H_t(w, v))^2$$

  do behave differently among edges inside a cluster and edges crossing different clusters.
- This gives us the first linear-time algorithm for graph clustering.
- Our intuitions are from random walk theory, but our analysis is based on geometry.
- A direct proof based on random walks?

Beyond Graph Clustering

What is the limit of this technique?

Revisit the Graph Expansion Problem

Graph Expansion

Given a d-regular graph $G = (V, E)$ as input, find a set $S \subseteq V$ of size $|S| \leq n/2$ of minimum conductance, i.e.,

$$\phi_G(S) = \min_{S' : |S'| \leq n/2} \phi_G(S').$$

Revisit the Graph Expansion Problem

Graph Expansion

Given a d-regular graph $G = (V, E)$ as input, find a set $S \subseteq V$ of size $|S| \leq n/2$ of minimum conductance, i.e.,

$$\phi_G(S) = \min_{S' : |S'| \leq n/2} \phi_G(S').$$

- This is the simplified version of graph clustering ($k = 2$ clusters).

Revisit the Graph Expansion Problem

Graph Expansion

Given a d-regular graph $G = (V, E)$ as input, find a set $S \subseteq V$ of size $|S| \leq n/2$ of minimum conductance, i.e.,

$$\phi_G(S) = \min_{S' : |S'| \leq n/2} \phi_G(S').$$

- This is the simplified version of graph clustering ($k = 2$ clusters).
- NP-hard to approximate, and there is no constant-factor approximation algorithms assuming the small-set expansion conjecture holds.

Revisit the Graph Expansion Problem

Graph Expansion

Given a d-regular graph $G = (V, E)$ as input, find a set $S \subseteq V$ of size $|S| \leq n/2$ of minimum conductance, i.e.,

$$\phi_G(S) = \min_{S' : |S'| \leq n/2} \phi_G(S').$$

- This is the simplified version of graph clustering ($k = 2$ clusters).
- NP-hard to approximate, and there is no constant-factor approximation algorithms assuming the small-set expansion conjecture holds.
- The current best approximation algorithm is based on SDP + geometric embedding. *Arora-Rao-Vazirani, JACM, 2009*

Revisit the Graph Expansion Problem

Graph Expansion

Given a d-regular graph $G = (V, E)$ as input, find a set $S \subseteq V$ of size $|S| \leq n/2$ of minimum conductance, i.e.,

$$\phi_G(S) = \min_{S' : |S'| \leq n/2} \phi_G(S').$$

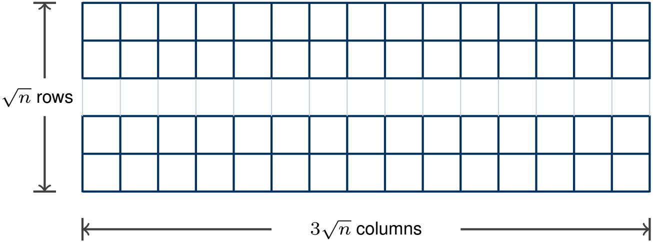
- This is the simplified version of graph clustering ($k = 2$ clusters).
- NP-hard to approximate, and there is no constant-factor approximation algorithms assuming the small-set expansion conjecture holds.
- The current best approximation algorithm is based on SDP + geometric embedding. *Arora-Rao-Vazirani, JACM, 2009*

Improve the state-of-the-art algorithm by heat kernels?

Grid Graphs

We define a family of graphs $\{G\}_n$ as follows:

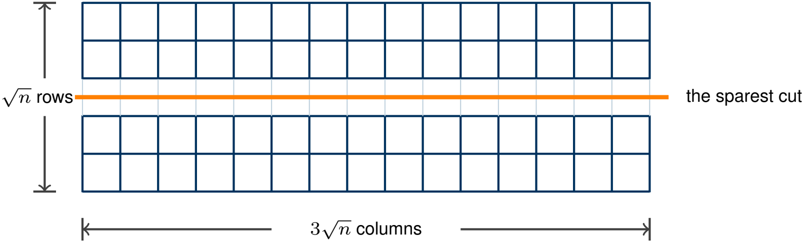
- Every G_n has $3n$ vertices, which form a grid of size $\sqrt{n} \times 3\sqrt{n}$.
- The weight of every edge in the middle row has weight $1/\sqrt{n}$, and all the other edges have weight 1.



Grid Graphs

We define a family of graphs $\{G\}_n$ as follows:

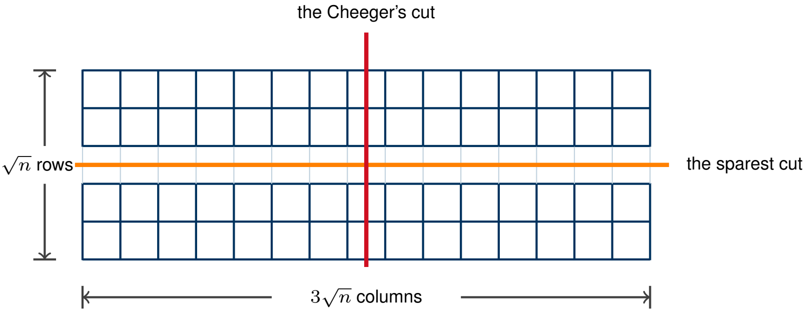
- Every G_n has $3n$ vertices, which form a grid of size $\sqrt{n} \times 3\sqrt{n}$.
- The weight of every edge in the middle row has weight $1/\sqrt{n}$, and all the other edges have weight 1.



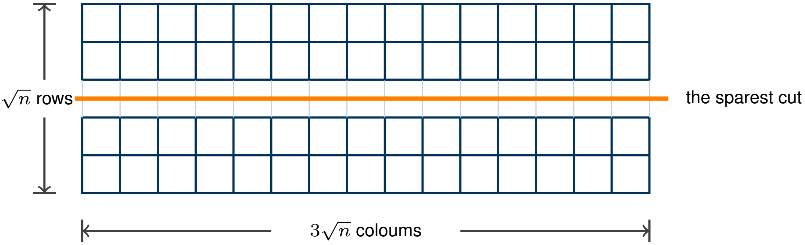
Grid Graphs

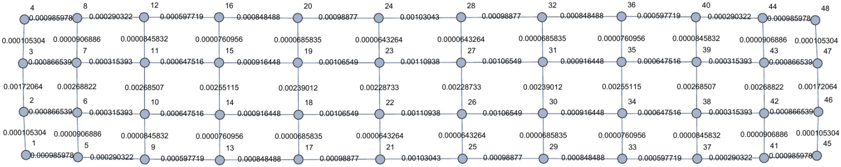
We define a family of graphs $\{G\}_n$ as follows:

- Every G_n has $3n$ vertices, which form a grid of size $\sqrt{n} \times 3\sqrt{n}$.
- The weight of every edge in the middle row has weight $1/\sqrt{n}$, and all the other edges have weight 1.



Heat Kernel Distances in the Grid Graphs





Summary

- Heat kernel is a basic notion in PDEs.

Summary

- Heat kernel is a basic notion in PDEs.

- We studied its connections to random walks and geometry, which allows us to design the first linear-time algorithm for graph clustering.

Summary

- Heat kernel is a basic notion in PDEs.
- We studied its connections to random walks and geometry, which allows us to design the first linear-time algorithm for graph clustering.
- This leaves us a number of interesting questions, including the powers and limits of PDEs for processing large datasets.

Summary

- Heat kernel is a basic notion in PDEs.
- We studied its connections to random walks and geometry, which allows us to design the first linear-time algorithm for graph clustering.
- This leaves us a number of interesting questions, including the powers and limits of PDEs for processing large datasets.

THANK YOU!

Reference: Richard Peng, He Sun, and Luca Zanetti: Partitioning Well-Clustered Graphs: Spectral Clustering Works! SIAM Journal on Computing, 46(2):710-743, 2017.