Graph Clustering

Group 2 : 김덕현 김민석
김선재 조승민
최승찬 추웅재

Min-cut based Graph Clustering

20201032 Deokhyeon Kim

Problem 1: Min-cut based graph clustering

- Definition: Construct k clusters from given component by iteratively eliminating min-cut edges.
- Algorithm: We are going to process k-1 iterations. At i-th (1 <= i <= k-1) iteration, i clusters are already found and we divide one cluster into two by eliminating min-cut edges from the component which has minimum min-cut.
- Min-cut of given component can be found by computing max-flow of all possible pair of nodes.
- There are efficient data structure to compute all-pair max-flow. (Gomory-Hu Tree)

Problem 2: 3-way min-cut

- Definition: Let w[i][j] denotes the cost when node i and node j are in the different clusters. The problem is to find 3 clusters that minimize the sum of w[i][j] given some nodes must belong to certain cluster.
- In case of 2 clusters, there are nice solution utilizing max-flow min-cut theorem. However, it is likely to be NP-hard in case of 3 (not proven yet).
- Therefore, I would like to think about an approximation algorithm that effectively solve this problem.

Questions

- 위 주제들과 관련된 선행연구가 있나요?
- 두 주제 중 어떤걸 하는게 좋나요?

(k,g)-Core Computation In a Strongly Induced Subhypergraph

20201040 Minseok Kim

Motivation

Definition 1. ((*k*, *g*)-core). *Given a hypergraph G, $k \geq 1$ and $g \geq 1$, (k, g)-core is the maximal set of nodes in which each node has at least k neighbours which appear in at least g hyperedges together in an induced subhypergraph by the (k, g)-core.*

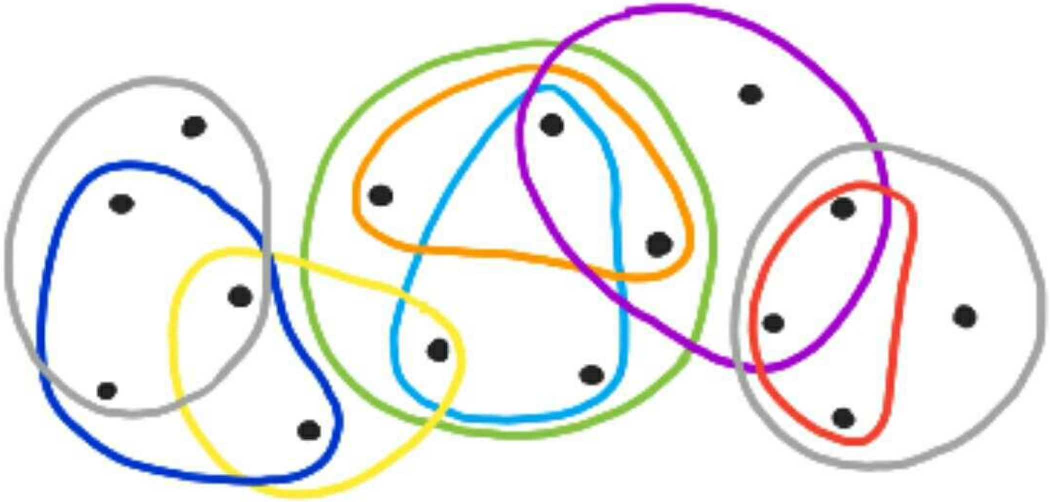
- Applications
  - Biological Systems
  - Recommendation System
  - Fraudster Detection, etc.

Motivation

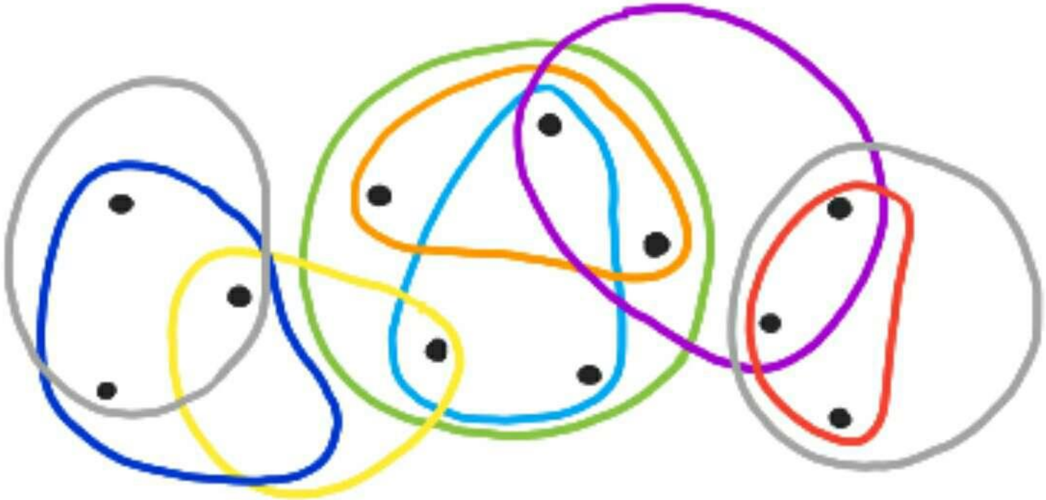
Strongly induced subhypergraph [6, 16, 28]. A strongly induced subhypergraph $H[S]$ of a hypergraph $H = (V, E)$, induced by a node set $S \subseteq V$, is a hypergraph with the node set S and the hyperedge set $E[S] \subseteq E$, consisting of all the hyperedges that are subsets of S.

$$H[S] = (S, E[S]), \text{ where } E[S] = \{e \mid e \in E \wedge e \subseteq S\} \tag{1}$$

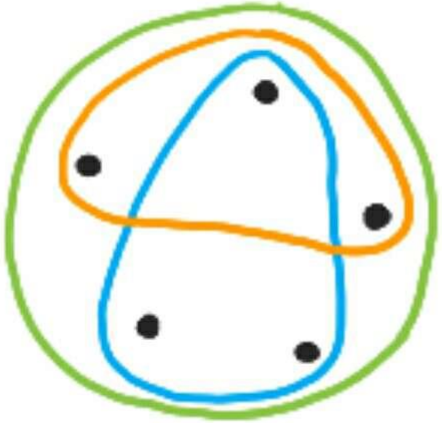
- There are some applications where the concept of a strongly induced subhypergraph is effectively utilized
  - Models such as *nbr-k-core* and *(k,d)-core* are proposed to deal with it
- However, existing *(k,g)-core* model doesn't consider it



Input Hypergraph



(k,g)-core and its induced hypergraph
where k = 2 and g = 2



Strongly induced (k,g)-core
where k = 2 and g = 2

Naïve approach

1. Count the number g-neighbors for each node
2. Iteratively peel the nodes which has less than k g-neighbors
3. Each node deletion, remove its incident edges and update its neighbors' g-neighbor counts
4. Repeat the above process until every node in the subgraph satisfies the (k,g) constraint

Discussion (Challenges)

- Although the naïve approach can be run in a polynomial time, it may be still impractical to be used in a very large hypergraph
  - Compute a nodes’ neighbors every time a node is deleted
    - Not time-efficient
  - Store the neighbors-list of all nodes in the graph
    - Not memory-efficient
- Therefore, some techniques to improve time/memory efficiency are needed
  - Prevent redundant computations
  - Pruning techniques (e.g. Upper bound pruning)
  - Parallelizing some tasks, etc.

Alternative Topic: nbr-(k,n)-core

- **Motivation**: nbr-k-core returns very large hypergraph once a single large-hyperedge exists
- **nbr-(k,n)-core**
  - Node strength: $S(u) = \sum_{e \in E(u)} \frac{1}{|e|^c}$
  - Definition: The maximal strongly induced subhypergraph such that every node *u* has at least *k* neighbors and its node strength is larger than *n*
  - Nodes with naïve edges(i.e. only contained in a few very large hyperedges) will have low node strength, and they will be filtered out
- **Discussion**
  - Lacking motivation: The meaning of the returned set of nodes
  - Techniques to make the algorithm efficient

Improving Efficiency of JSON Index Extraction Using Sampling-based Approaches

| Student Name: Sunjae Kim | Student ID: 20201343 |

Research question: Can <u>sampling-based approaches</u> improve the <u>efficiency of JSON index extraction</u> without significantly compromising the <u>accuracy or completeness</u> of the extracted <u>index information</u>?

Proposed Method

- Apply random, stratified, or adaptive sampling techniques
- Extract index keys and structures from the sampled subset
- Infer global index patterns based on the sample
- Evaluate trade-offs between speed, memory usage, and accuracy of the inferred indexes

Experiment Plan

Explore and devise algorithms affecting

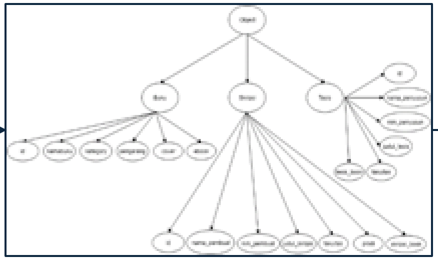
- Accuracy of the extracted index structure compared to full extraction (precision, recall),
- Index extraction time (runtime)
- Memory consumption
- Impact on downstream query execution performance.

Expected Contribution

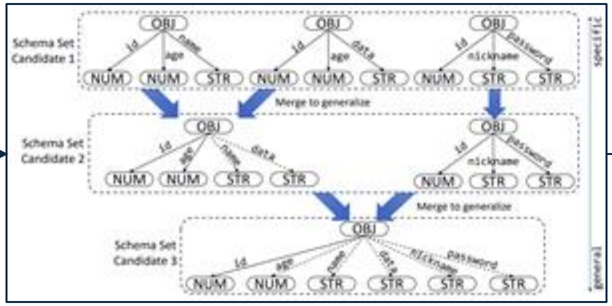
- A novel sampling-based algorithm for efficient JSON index extraction.
- Analysis of trade-offs between performance and accuracy.
- Open-source implementation and benchmarking tools for reproducibility.

Visual Demo

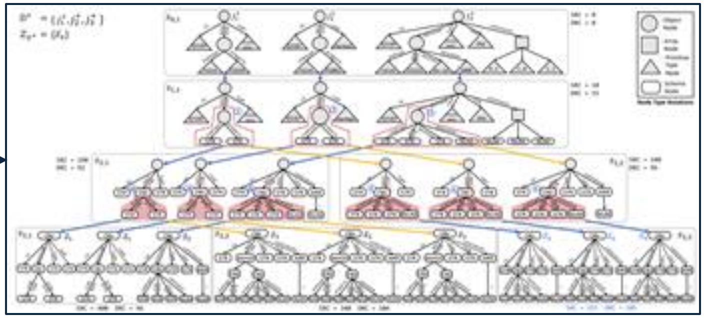
JSON Data Prep



Tree Conversion



Sampling



Apply Index

Optimized Force-directed placement using Laplacian-based von Neumann Entropy

20201290 Seungmin Cho

Motivation 1. Force-directed

- Graph embedding by **force-directed** placement
- Simulates a intuitive physical system

- Nodes **repel**, Edges act as **springs**
- Stable position in Euclidean space by **iteration**

- Use **temperature** to clustering with the proper number of iteration

- Limitation: expensive($O(n^3)$), ambiguous,
  designed for unweighted graphs

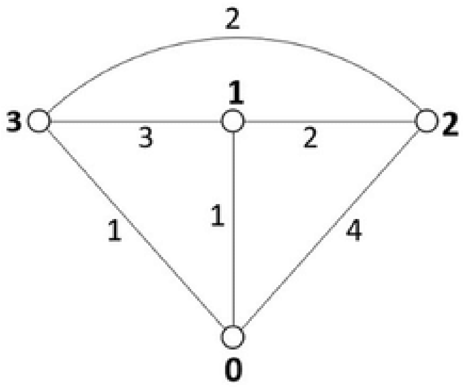
```
area := W * L; { W and L are the width and length of the frame }
G := (V, E); { the vertices are assigned random initial positions }
k := √(area/|V|);
function f_a(x) := begin return x²/k end;
function f_r(x) := begin return k²/x end;

for i := 1 to iterations do begin
    { calculate repulsive forces }
    for v in V do begin
        { each vertex has two vectors: .pos and .disp }
        v.disp := 0;
        for u in V do
            if (u ≠ v) then begin
                { Δ is short hand for the difference}
                { vector between the positions of the two vertices }
                Δ := v.pos − u.pos;
                v.disp := v.disp + (Δ/|Δ|) * f_r(|Δ|)
            end
    end

    { calculate attractive forces }
    for e in E do begin
        { each edge is an ordered pair of vertices .v and .u }
        Δ := e.v.pos − e.u.pos;
        e.v.disp := e.v.disp − (Δ/|Δ|) * f_a(|Δ|);
        e.u.disp := e.u.disp + (Δ/|Δ|) * f_a(|Δ|)
    end

    { limit the maximum displacement to the temperature t }
    { and then prevent from being displaced outside frame }
    for v in V do begin
        v.pos := v.pos + (v.disp/|v.disp|) * min(v.disp, t);
        v.pos.x := min(W/2, max(−W/2, v.pos.x));
        v.pos.y := min(L/2, max(−L/2, v.pos.y))
    end
    { reduce the temperature as the layout approaches a better configuration }
    t := cool(t)
end
```

Figure 1. Force-directed placement

Motivation 2. Laplacian

- Laplacian mtx. L = D – A
  s.t. D is degree mtx. & A is adjacency mtx.

$$(L_G)_{(i,j)} = \begin{cases} -w(x_i, x_j) & \text{if } (x_i, x_j) \in E, \\ 0 & \text{if } (x_i, x_j) \notin E \text{ and } i \neq j \\ \sum_{(x_i, x_k) \in E} w(x_i, x_k) & \text{if } i = j. \end{cases}$$



$$Y = \begin{array}{c} \\ \mathbf{1} \\ \mathbf{2} \\ \mathbf{3} \\ \mathbf{0} \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 0 \end{array} \\ \begin{bmatrix} 6 & -2 & -3 & -1 \\ -2 & 8 & -2 & -4 \\ -3 & -2 & 6 & -1 \\ -1 & -4 & -1 & 6 \end{bmatrix} \end{array}$$

- Powerful tool to analyze the **flow** of graph.
- Symmetric => Eigen decomposition
  - e.g. Spectral clustering with Fiedler pair(i.e. second minimum eigenpair)

- Able to define **Heat Operator** $f(t) \coloneqq e^{-tL_G}$ for $t \in [0, \infty)$
  - Heat equation $\frac{\partial u}{\partial t} = \Delta u$ <=> $\frac{df(t)}{dt} = -L_G f(t)$ w/ sol'n $f(t) = e^{-tL_G}$

Motivation 3. von Neumann Entropy

- Amount of Information: measure of **uncertainty**.
  - e.g. Sun will rise from the east(No information). Bitcoin will rise(Informaton).
- Entropy: **average** amount of information with considering all possible outcomes.

- Shannon Entropy: $H(X) \coloneqq E\big(I(X)\big) = -\sum_{x \in X} P(x) \lg P(x)$
  - Shannon Information $I(\mathrm{x}) \coloneqq -\lg P(x)$ (e.g. $I_{coin}(head) = -\lg\frac{1}{2} = 1$, $I_{die}(4) = -\lg\frac{1}{6} \approx 2.58$)

- In quantum system, states are not represented by prob. dist., but by **density mtx.** ρ.
- von Neumann Entropy: $S(\rho) \coloneqq -\mathrm{Tr}(\rho \log \rho)$
- By heat operator, we can define density mtx. $\rho \coloneqq \frac{e^{-L_G}}{Tr(e^{-L_G})}$

Intuitive Glimpse

- Force-directed Placement gives intuitions of **Heat system**.
  - They use repulsive force, temperature.
- Laplacian mtx. can represent **heat operator** of graph.
- von Neumann entropy shows the graph system has its own **entropy**.
- It is convincing to combine them with aspect of thermodynamics.

- It looks possible to **optimize** the force-directed placement using von Neumann entropy from Laplacian mtx.
  - Laplacian may measure the power of springs on weighted edge.
  - Heat operator may measure the temperature of whole system.
  - Entropy may measure the reliability of the clustering.

Discussion

- How can this approach be distinguished from other similar models?
- Has this approach been explored already?
- Is it sure to improve the method by this approach? Can this approach be optimized?
- Is the improvement of this approach theoretically provable on general stage, or just showable to empirical efficiency on selected cases?

Extension to SCAN

20231393 최승찬

DBSCAN : Density based clustering

- Can we apply DBSCAN to “Graph”?

-> SCAN : A Structural Clustering Algorithm for Networks

SCAN

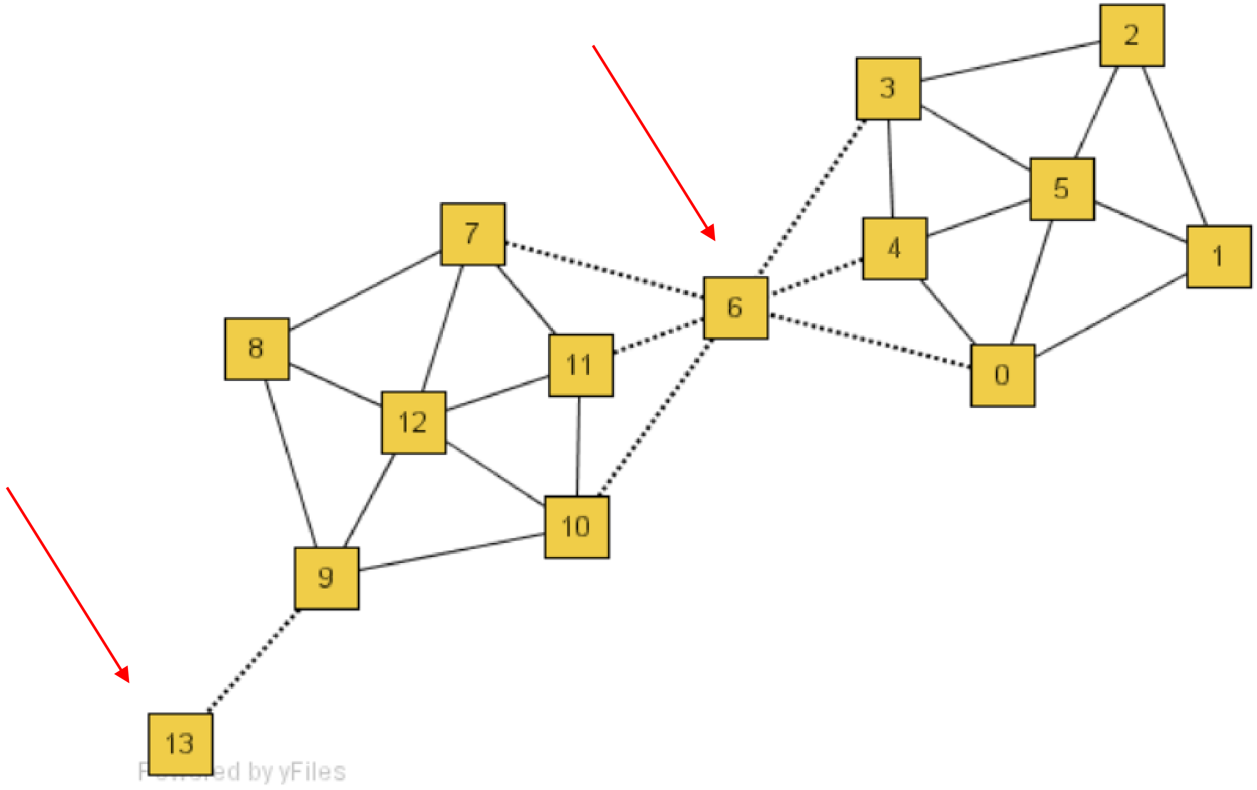


Figure 1. A Network with 2 Clusters, a Hub and an Outlier.

SCAN – "How they share neighbors"

- Neighborhood

Let $v \in V$, the structure of v is defined by its neighborhood, denoted by $\Gamma(v)$

$$\Gamma(v) = \{w \in V \mid (v,w) \in E\} \cup \{v\}$$

- Structural Similarity

DEFINITION 2 (STRUCTURAL SIMILARITY)

$$\sigma(v,w) = \frac{|\Gamma(v) \bigcap \Gamma(w)|}{\sqrt{|\Gamma(v)||\Gamma(w)|}}$$

SCAN

DEFINITION 3 (ε-NEIGHBORHOOD)

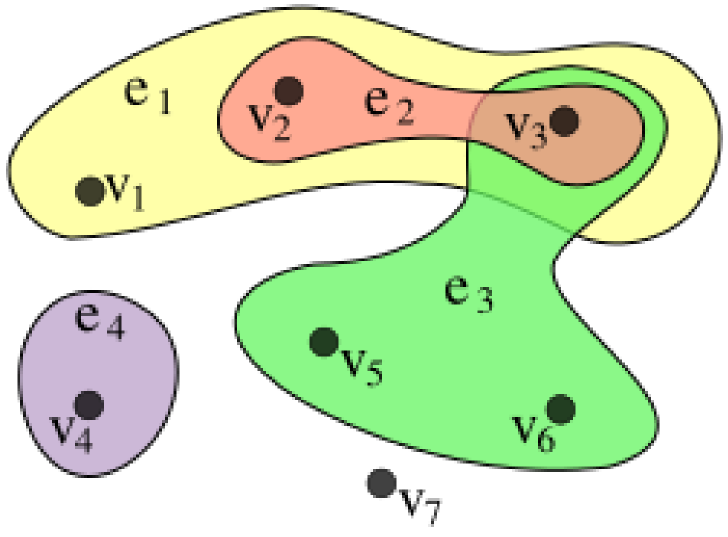
$$N_\varepsilon(v) = \{w \in \Gamma(v) \mid \sigma(v,w) \geq \varepsilon\}$$

$$CORE_{\varepsilon,\mu}(v) \Leftrightarrow |N_\varepsilon(v)| \geq \mu$$

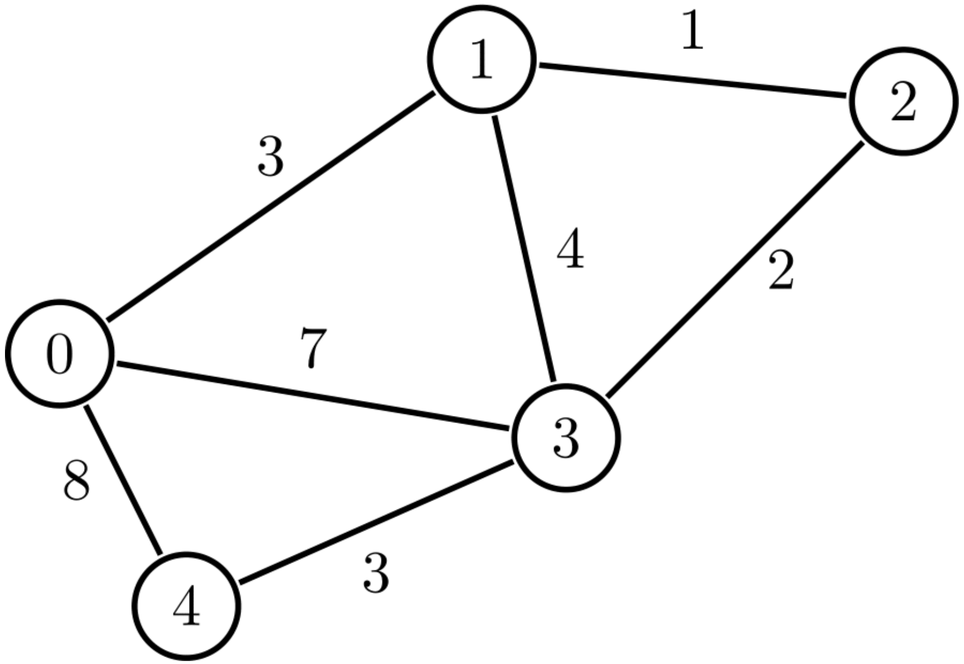
->reachability -> connectivity -> cluster

Extension

1) Hypergraph



2) Weighted graph



Discussion

1) Which Topic is more attractive? (or accurate)

2) In hypergraph, how should I consider "multiple neighbor?"
   (two or more edges connecting v and w)

3) The key in SCAN is "how much neighbors they share".
What should be the key (and motivation) in weighted SCAN?
   "how much neighbors they share"
   +
   "how hardly connected"(?)

A Clustering Game on Graphs: Conductance vs Density

20201317 Woungjae Choo

A Clustering Game on Graphs

- Given a graph and designated seed node
- Two players take turns playing the game.
- On each turn, a player chooses one of two possible actions to play.
  1. Terminate the clustering process.
  2. Expand the cluster by including one neighboring node not yet in the cluster.

Goal of the two players

- There are various metrics for evaluating partial clusters.
- Among these, conductance and density, which are commonly used, will be chosen as the goals of the two players.
- Player A's goal is to minimize the conductance value
- Player B's goal is to maximize the density value.

Expected Outcome and Questions

- The expected outcome is the formation of a cluster that incorporates the advantages of both conductance and density-based approaches.

- The question here is: since it seems difficult to obtain the result of both players playing perfectly, how should we approach such a situation?