Graph Theory and Algorithms

Ph.D. Course – Marco Viviani

Graph Partitioning and Clustering
(June 18, 2021 / 15:00-17:00)

TABLE OF CONTENTS

1

Intro and Basic Notions

Cuts, Flows, and *s*-*t* Cuts

Motivations

- Cutting a graph into “smaller pieces” is one of the fundamental algorithmic operations.
- With the advent of even larger instances in applications such as scientific simulation, social networks, or road networks, **graph partitioning** and **graph clustering** therefore becomes highly important, multifaceted, and challenging.
- Partitioning or clustering large graphs is often an important subproblem for complexity reduction or parallelization.

Motivations … Cont'd

- A commonly used method to partition or cluster large graphs is the **multilevel approach** or variations thereof.
- Here, the graph is recursively contracted to create smaller graphs which somewhat should reflect the same basic structure as the input graph.
- This usually achieved by **modifying edge and node weights** of the coarser graphs.

Cuts

- In graph theory, a **cut** is a partition of the vertices of a graph into two disjoint subsets.
- Any cut determines a **cut-set**, the set of edges that have one endpoint in each subset of the partition.
- These edges are said to **cross the cut**.

Flow Networks and $s - t$ Cuts

- In graph theory, a **flow network** (also known as a transportation network) is a directed graph where each edge has a capacity, and each edge receives a flow.
- The amount of flow on an edge **cannot exceed the capacity of the edge**.
- A **flow** must satisfy the restriction that the amount of flow into a node equals the amount of flow out of it, unless it is a **source**, which has only outgoing flow, or **sink**, which has only incoming flow.
- A network can be used to model traffic in a computer network, circulation with demands, fluids in pipes, currents in an electrical circuit, or anything similar in which something travels through a network of nodes.

Flow Networks and $s-t$ Cuts ... Cont'd

- An **$s-t$ cut** is defined as tuple $(S, V \setminus S)$ with $s \in S \subset V$ and $t \in V \setminus S$.
- The **weight** of an $s-t$ cut is defined as $\sum_{\{(u,v) \in E \cap S \times V \setminus S\}} \omega(u,v)$, i.e., the weight of the edges starting in S and ending in $V \setminus S$.
- A **minimum $s-t$ cut** has the smallest weight among all $s-t$ cuts.

2

Graph Partitioning

Graph Partitioning

Given a number $k \in \mathbb{N}_{>1}$ and an undirected graph with *non-negative* edge weights, the *graph partitioning problem* asks for *blocks* of nodes $V_1,\ldots,V_k$ that partition the node set V, i.e.

1. $V_1 \cup \cdots \cup V_k = V$
2. $V_i \cap V_j = \emptyset \;\; \forall i \neq j.$

Graph Partitioning … Cont'd

A *balance constraint* demands that all blocks have about equal size. More precisely, it requires that,

$$\forall i \in \{1..k\} : |V_i| \leq L_{\max} := (1+\epsilon)\lceil |V|/k \rceil$$

for some imbalance parameter $\epsilon \in \mathbb{R}_{\geq 0}$ in the case that the cost function of the nodes is identical to one. Often after one has computed a partition of a graph one reports the *maximum imbalance* $\max_i |V_i|/\lceil |V|/k \rceil$ of the partition.

Graph Partitioning … Cont'd

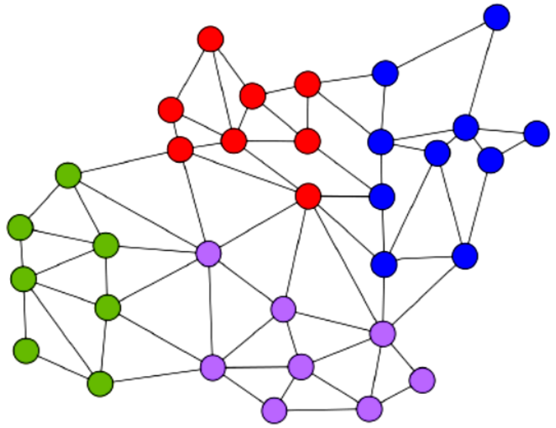


Figure 1.1: An example graph that is partitioned into four blocks. The partition has 17 cut edges. The weights of the blocks are blue(10), red(8), green(7) and purple(8). Hence, the maximum imbalance is 11%.

Graph Partitioning … Cont'd

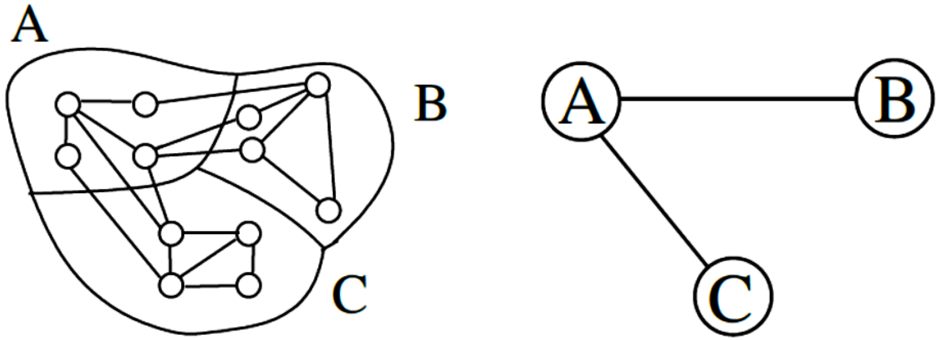


Figure 1.2: A graph that is partitioned into three blocks of size four on the left and its corresponding quotient graph on the right. There is an edge in the quotient graph if there is an edge between the corresponding blocks in the original graph.

Objective Functions

In practice, we often seek to find a partition that minimizes (or maximizes) an objective. Probably the most prominent objective function is to minimize the *total cut*

$$\sum_{i<j} \omega(E_{ij}).$$

In other words, the objective computes the sum of the weight of the cut edges.

3

Graph Clustering

Graph Clustering

A *clustering* is also a partition of the nodes, however k is usually not given in advance and the balance constraint is removed. Note that a partition is also a clustering of a graph. In both cases, the *goal* is to minimize or maximize a particular objective function. Sometimes, there is an upper bound to the cluster sizes specified. One of the main paradigms of clustering is to find groups/clusters intra-cluster density vs. inter-cluster sparsity. As we will see this can be formalized with a number of objective functions.

Graph Clustering (Example)

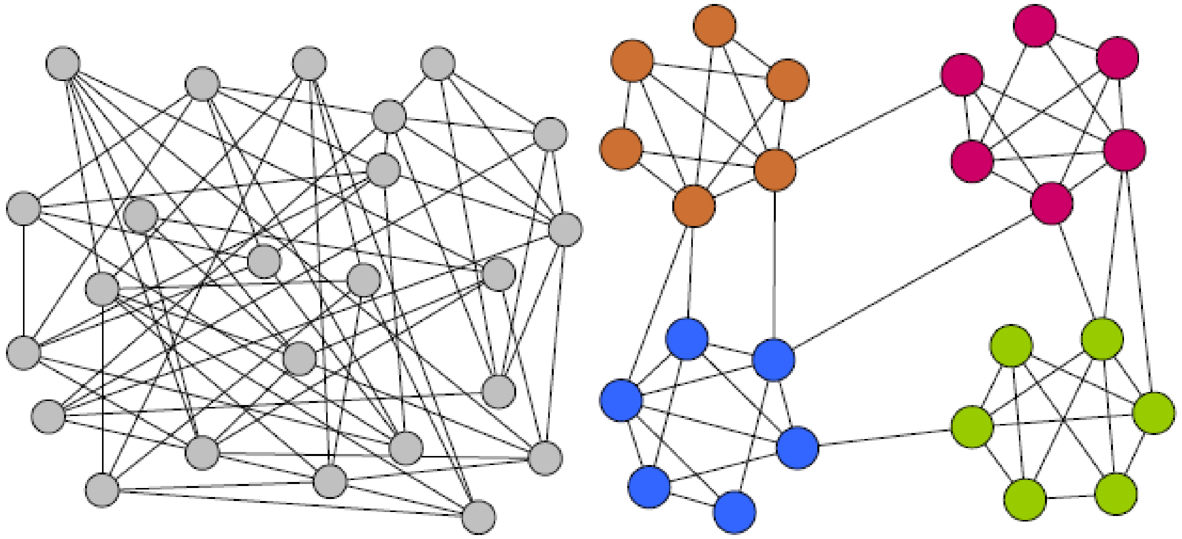


Figure 1.7: An example clustering of a graph.

Coverage

Coverage. The most simple index realizing a traditional measure of clustering quality is coverage. The coverage($\mathcal{C}$) of a graph clustering $\mathcal{C}$ is defined as the fraction of intra-cluster edges (or $\omega(C)$) within the complete set of edges (or W):

$$\text{cov}(C) := \frac{m(\mathcal{C})}{m}$$

$$\text{cov}_\omega := \frac{\omega(\mathcal{C})}{W}$$

Intuitively, large values of coverage correspond to a good quality of a clustering.

Modularity

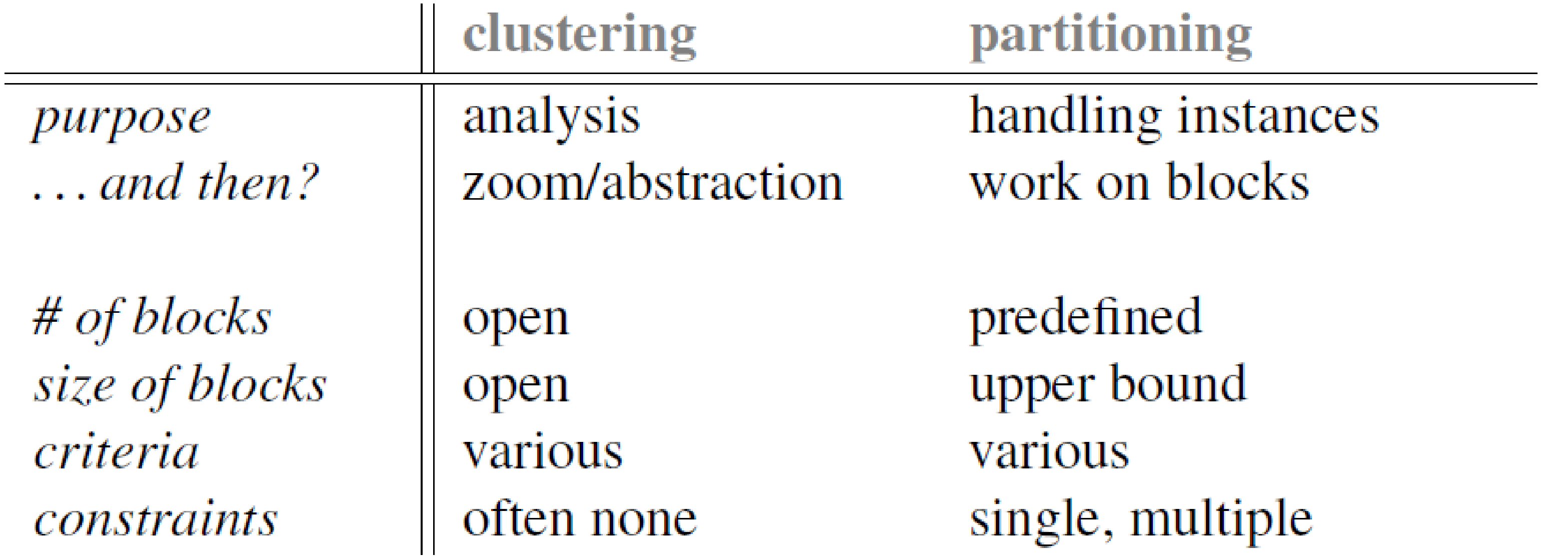
Modularity. Modularity has been proposed [101] in an attempt to find an apt remedy to the disadvantages of coverage. Citing the authors, the driving idea for modularity was to take coverage "minus the expected value of the same quantity in a network with the same community divisions, but random connections between the vertices.". The commonly used formula is as follows:

$$\mathrm{mod}(\mathcal{C}) := cov(\mathcal{C}) - \mathbb{E}[cov(\mathcal{C})] = \frac{m(\mathcal{C})}{m} - \frac{1}{4m^2} \sum_{C \in \mathcal{C}} \left(\sum_{v \in C} \deg(v) \right)^2$$

The original formulation did not take into account loops and miscounted coverage for non-simple graphs.

[101] M. EJ Newman and M. Girvan. Finding and Evaluating Community Structure in Networks. *Physical review E*, 69(2):026113, 2004.

Partitioning VS Clustering



	clustering	partitioning
purpose	analysis	handling instances
… and then?	zoom/abstraction	work on blocks
# of blocks	open	predefined
size of blocks	open	upper bound
criteria	various	various
constraints	often none	single, multiple

4

Graph Partitioning Paradigms

Spectral Graph Partitioning

- A **partition** is derived from approximate eigenvectors of the adjacency matrix.
- Deepen/revise the concepts of eigenvectors and eigenvalues → Possible assignment → Next slides.

Multilevel Graph Partitioning

- A **multilevel graph partitioning** algorithm works by applying one or more stages.
- Each stage reduces the size of the graph by **collapsing vertices and edges** (**coarsening**), partitions the smaller graph, then maps back and refines this partition of the original graph.
- One widely used example of such an approach is **METIS**, a graph partitioner, and **hMETIS**, the corresponding partitioner for hypergraphs.

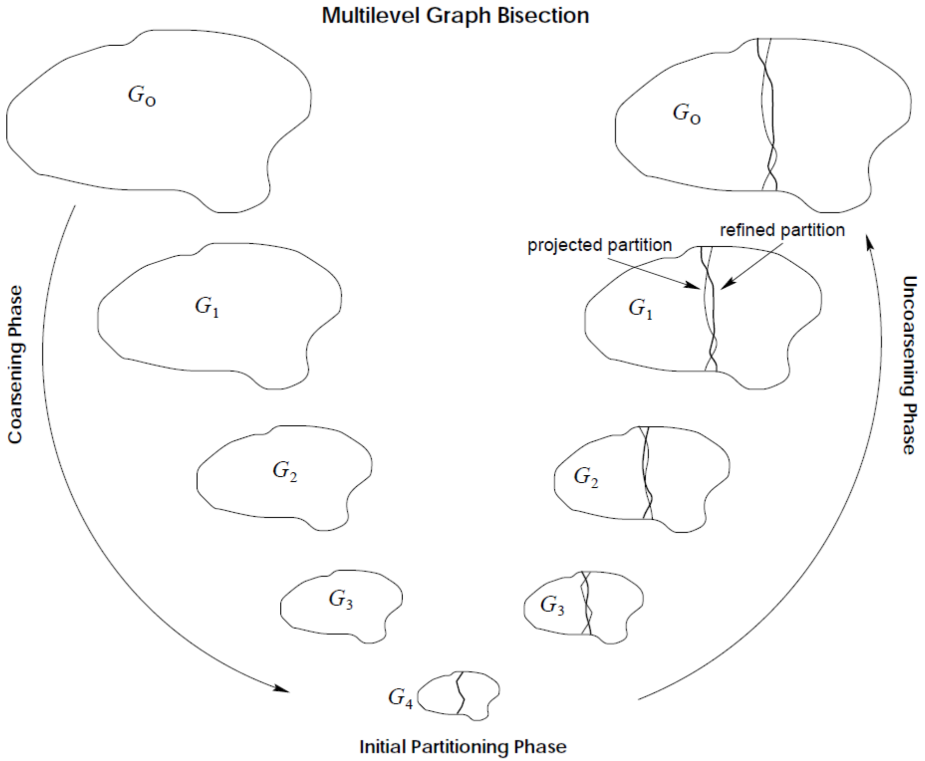
Karypis, G., & Kumar, V. (1997). METIS: A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices.
Karypis, G., Aggarwal, R., Kumar, V., & Shekhar, S. (1999). Multilevel hypergraph partitioning: Applications in VLSI domain. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 7(1), 69-79.

Multilevel Graph Partitioning … Cont'd

- The multilevel approach to graph partitioning consists of **three main phases**.

  1. Coarsening phase
  2. Initial partitioning phase
  3. Uncoarsening phase

Multilevel Graph Partitioning (Example)



Graph Coarsening

- In the **coarsening (contraction) phase**, a hierarchy of graphs is created.

- Contraction should quickly reduce the size of the input and each computed level should reflect the global structure of the input network.

Graph Coarsening (Example)

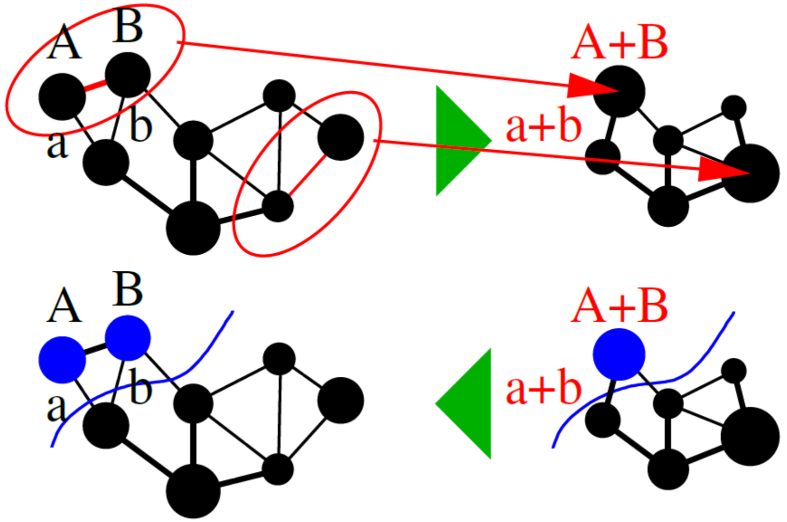
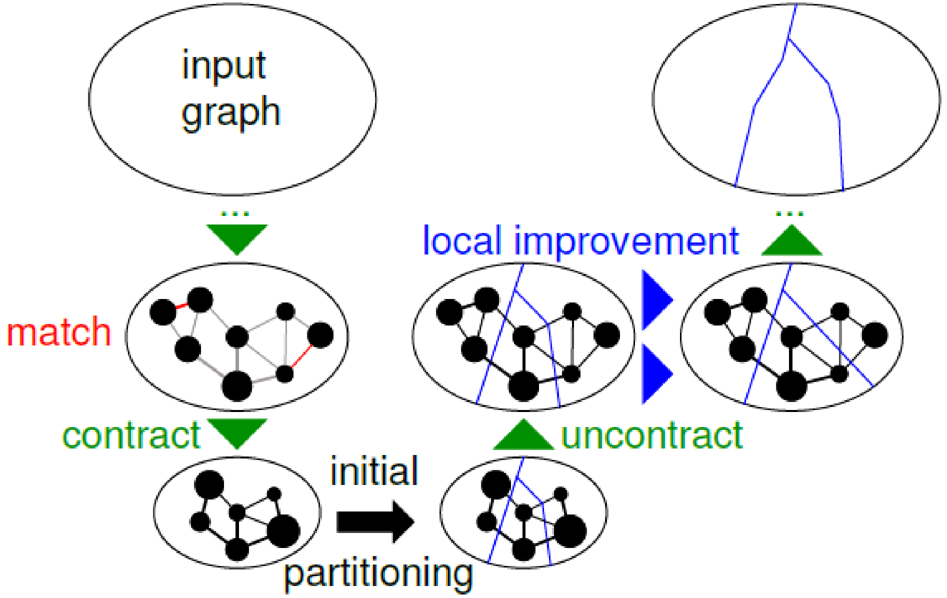


Figure 5.2: A example matching highlighted in red and contraction of matched edges. After a partition has been computed on the coarser graph, it can be transferred to the finer level by putting the finer nodes into the partition of their coarse representative. Due to the way the contraction is defined, edge cut and balance are the same after solution transfer. Source: [122].

Graph Uncoarsening

- In the **uncoarsening (or local improvement) phase**, the matchings are iteratively uncontracted.
- After uncontracting a matching, a local improvement algorithm moves nodes between blocks in order to improve the cut size or balance.



5

Graph Clustering Paradigms

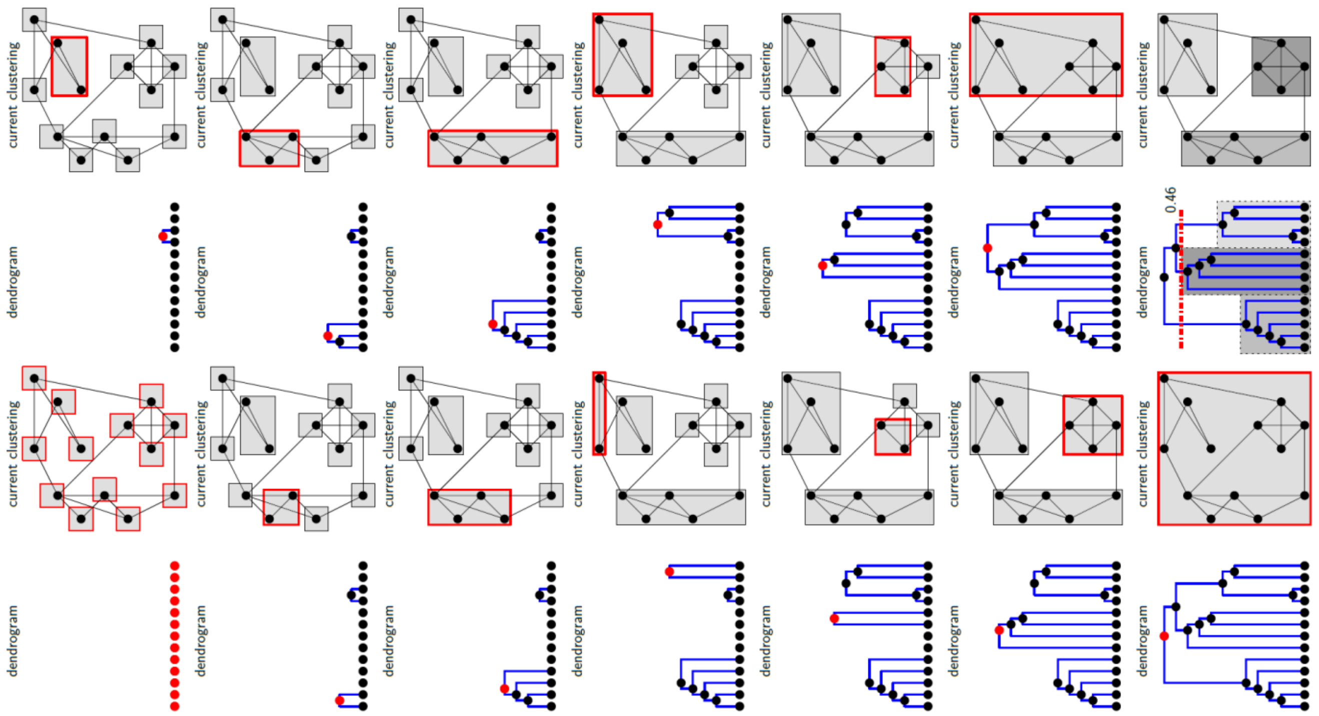
Greedy Agglomeration

- Roughly speaking, the algorithm starts with a **singleton clustering** and iteratively merges/joins clusters.
- To describe the network's community structure, a **dendrogram** is used.
- The **hierarchical decomposition** of the network is presented at all levels.

Greedy Agglomeration … Cont'd

- At each iteration of this method, pairs of communities are joined, and the **modularity** (naïve solution) is subsequently calculated.
- More precisely, in each iteration the pair of communities yielding the highest gain in modularity is merged.

Greedy Agglomeration (Example)



Spectral Clustering

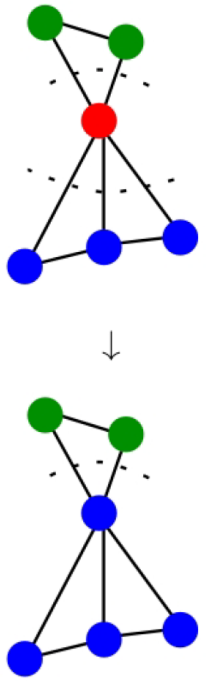
- It **clusters** graph vertices using the eigen decomposition of the graph Laplacian matrix.
- Deepen/revise the concepts of eigenvectors and eigenvalues → Possible assignment → Next slides.

Louvain Method

- Simple heuristic method which is based on **modularity optimization** and works fast.
- The basic techniques involved are the **local movement and multi-level clustering**.
- At the beginning, each node is a singleton cluster, then the nodes are traversed in random order and move to the neighboring cluster yielding the highest modularity increase.

Louvain Method … Cont'd

- The algorithm runs in **two phases**.
  - In the first phase, the algorithm is firstly initialized, so that each node is assigned to a different cluster (community).
  - Then, we iteratively proceed with following steps, targeting to find local maxima:
    1. pick a node u at random
    2. remove u from cluster, compute removal gain ΔQ
    3. for each cluster, compute add gain $\Delta Q'$
    4. if highest total gain is positive (i.e., $\Delta Q' - \Delta Q > 0$), then move node



6

Possible Assignments

Graph Partitioning / Clustering (Spectral, METIS, FluidC)

Possible Assignments

- Deepen a specific graph partitioning/clustering algorithm based on the spectral partitioning/clustering paradigm.
- Deepen a specific graph partitioning algorithm based on the multilevel partitioning paradigm (e.g., METIS).
- Deepen a specific graph partitioning algorithm based on the concepts of flows (e.g., FluidC).