MATCHING CUT

Komusiewicz, Christian & Kratsch, Dieter & Le, Van Bang. (2019). Matching Cut: Kernelization, Single-Exponential Time FPT, and Exact Exponential Algorithms. Discrete Applied Mathematics. 283. 10.1016/j.dam.2019.12.010.

Matching Cut: Kernelization, Single-Exponential Time FPT, and Exact Exponential Algorithms

Table of Contents

Problem Definition

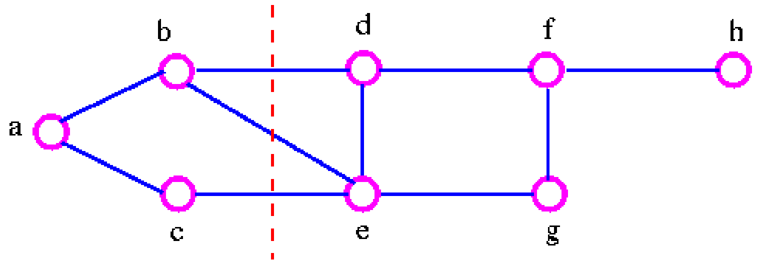
For a graph G = (V, E), a cut is a vertex partition (A, B) such that A != ∅, B != ∅, A ∩ B = ∅ and A ∪ B = V.

The set of all the edges with one endpoint in A and another in B denoted by E(A, B) is called an edge cut.

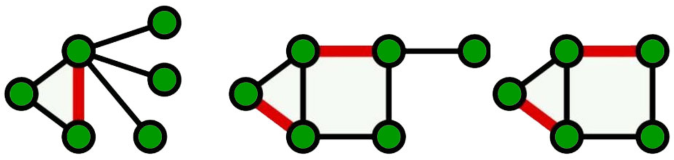
A matching is an edge set M ⊆ E such that no two edges ei , ej ∈ M shares any endpoint, A matching cut is an edge cut which is also a matching.

The MATCHING CUT problem is to decide if a given undirected graph G has a matching cut or not.

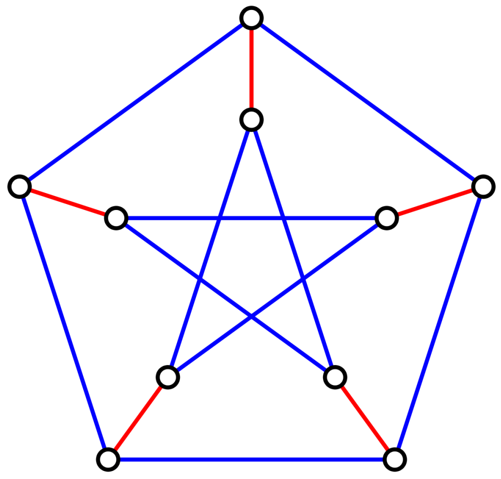
The parameterized version of the problem is to decide if a given undirected graph G having a parameter distance to cluster dc has a matching cut or not.



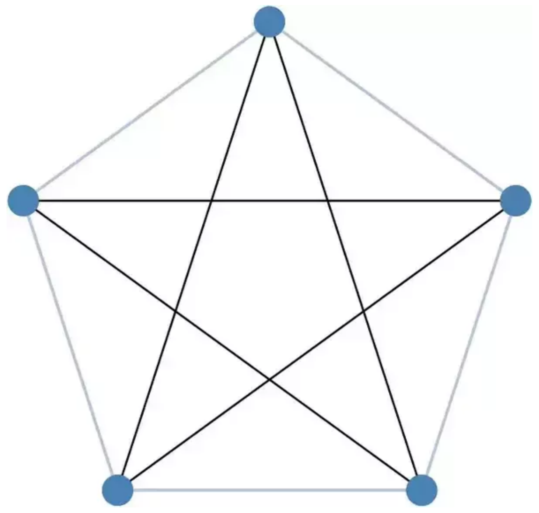
Example for edge cut



Example for Matching



Example for YES instance of Matching cut



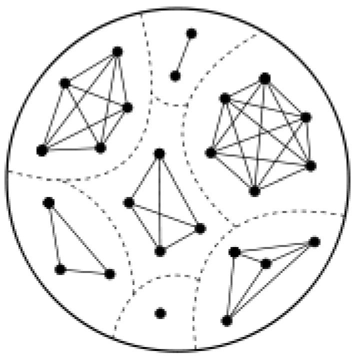
Example for NO instance of Matching cut

Classical Complexity

The Matching Cut problem is proved to be NP Complete in specific graphs, such as planar graphs of maximum degree 4, showing that it is NP Hard. Given an edge set, we can verify if it is a matching cut in O(E^2) time. Thus we can conclude that **Matching Cut is NP Complete**.

Graph notations and terminology

- Let G(V, E) be a graph with vertex set V(G) = V and edge set E(G) = E
- | V | = n and | E | = m
- A stable set (Independent set) / clique is a set of pairwise non-adjacent / adjacent vertices
- The neighbourhood of a vertex v in G, Ng(v) or N(v), is the set of vertices in G adjacent to v
- Degree of the vertex, deg(v) = | N(v) |
- For a subset W ⊆ V, G[W] is the subgraph of G induced by W and G - W stands for G [V \ W]
- W-neighbours of v, Nw(v), are the neighbours of V in the induced subgraph
- A cluster graph is a graph induced by a disjoint union of cliques.



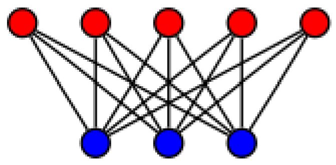
Example of a cluster graph
Ref: https://en.wikipedia.org/wiki/Cluster_graph

- Each such maximal clique(each such disjoint clique)

are referred to as clusters.
The example has 7 clusters

- A co-cluster graph is a graph such that its complement is a cluster graph(multipartite graph, having the number of clusters of the complement as the number of partitions)
- A single clique is both a cluster and a co-cluster graph

- A vertex cover of G is a subset C ⊆ V such that every edge of G has at least one endpoint in C. V \ C is a stable set. The smallest size of a vertex cover of G is called the vertex cover number of G, denoted by τ (G).
- For a graph property P, a distance to P set is a subset U ⊆ V such that G - U has the property P. The distance to P is the smallest size of a distance to P set.
    - Distance to cluster dc(G) is the smallest size of distance to cluster set, ie, by removing the distance to cluster set having size dc(G), the resultant graph is a cluster graph
    - Distance to co-cluster $d\bar{c}(G)$is the smallest size of distance to co-cluster set
    - Distance to clique dq(G) is the smallest size of distance to clique set(removal results in a clique)
- τ (G) ≥ max{ dc(G), $d\bar{c}(G)$}
    - If we remove the vertex cover, it trivially becomes a cluster and a co-cluster graph. Thus, dc(G), $d\bar{c}(G)$ must both be at most τ (G)
- dq(G) ≥ max{ dc(G), $d\bar{c}(G)$}
    - If we get a clique from removing distance to clique set, since a clique is both a cluster and a co-cluster graph, dc(G), $d\bar{c}(G)$ must both be at most dq(G)
- A subset U ⊆ V is called monochromatic in G if for every matching cut (A, B) if G, either U ⊆ A or U ⊆ B. Can also be referred to as G[U] is monochromatic in G.
    - Monochromatic property is hereditary: Every induced subgraph of G also carries the property for the subset U.
    - A complete subgraph Kn is monochromatic if n = 1 or n >= 3
    - a complete bipartite subgraph Kn,m is monochromatic if n ≥ 3 and m ≥ 2



The example has m = 5 and n = 3

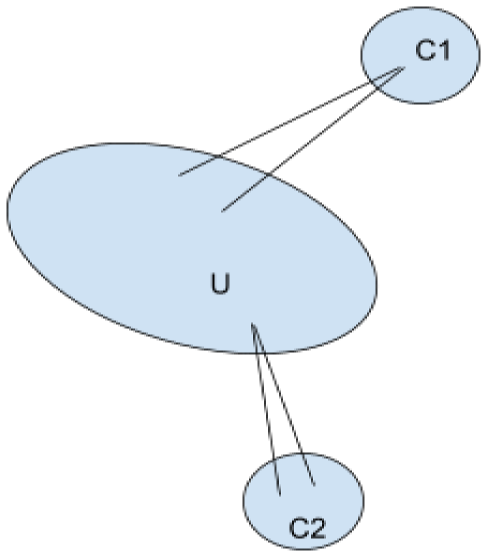
Kernelization for over Distance to Cluster

- Parameters available for Matching cut include tree-width of a graph, minimum size of the matching cut set, maximum degree of the input graph. All these parameters do not admit a polynomial kernel. (proof: https://arxiv.org/abs/1011.4224)
- Disconnected graphs and graphs having a vertex of degree at most one admit a matching cut. Thus we assume that graphs are considered to be connected and have a minimum degree of at least 2.
- We start with the vertex set U = {U1, U2, U3…. Up} where p <= 3*dc(G) can be calculated using a 3-approximation of Cluster Vertex Deletion in time O(dc(G)(n+m)) based on the observation that a graph is a cluster graph if and only if it does not contain an induced path on three vertices
- We maintain a partition of U1, U2 …. Ul such that Ui is monochromatic. Initial partition will be single vertex sets for each partition Ui = {ui}
- The kernelization focuses on merging sets Ui and Uj, removing Ui and Uj and adding Ui U Uj. Merging is safe if Ui U Uj is monochromatic in G.

Rule 1

If V \ U contains a degree 1 vertex or a cluster C such that (V \ C, C) is a matching cut, it is a YES instance

- For a degree 1 vertex, removing that single edge results in the partition {v} and V \ {v} which satisfies matching cut properties, hence it is a YES instance.
- In case of the cluster, the partition between the cluster and the rest of vertices can be considered a matching cut.
- After application of this rule, degree of vertices are at least 2 and **each cluster has at least 2 neighbours in U or at least there is a vertex in U that has 2 neighbours in C**.(C1 or C2)



Time to run - (n+m) to find clusters + n for degree + O(m) - each edge find if u and v in diff partition and

For each sets Ui, we define N2 (Ui) the set of vertices v ∈ V \ U which follows at least one of the 3 properties:

- v has 2 neighbours in Ui,
- v is in a cluster of size at least 3 in G - U that contains a vertex that has 2 neighbours in Ui,
- v is in a cluster C in G - U and some vertex in Ui has 2 neighbours in C.

Proposition 1: Ui U N2(Ui) is monochromatic

Case 1: Trivially monochromatic

Case 2: The clusters of size at least size 3 are monochromatic and therefore this vertex with Ui is also monochromatic. In case of size 2 clusters, this may not necessarily be monochromatic.

Case 3: 2 vertices form a triangle with a vertex in Ui, thus monochromatic

Time to run to calculate N2(Ui) for all Ui: Can be done in O(dc * (n + m)) : Case 1 takes O(dc * (m+n)) followed by Case 3 taking O(dc * n + m) after which Case 2 can be done in O(n) - We can assign each cluster to an N2(Ui) if there and each vertex refer to the cluster's status directly.

Rule 2

If there is a vertex v that is in N2(Ui) and N2(Uj) for i != j, merge Ui and Uj.

- By proposition 1, N2(Ui) U Ui and N2(Uj) U Uj are monochromatic in G. We can conclude that Ui U Uj U {v} is monochromatic. (can be inferred as v binding Ui and Uj together)

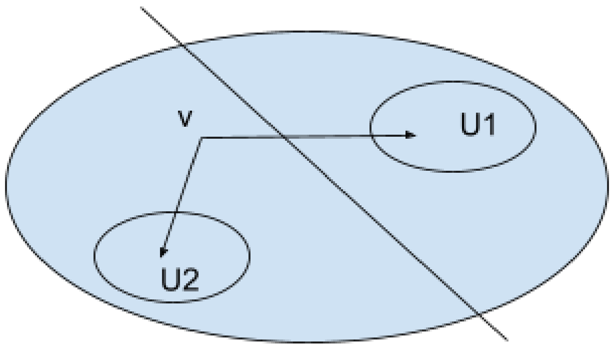


Fig 2: Negative example for Rule 2

Time to test: O(dc * n)
Time to run: O(n)

Rule 3

If there are 3 vertices v1, v2, v3 in V that has 2 common neighbours u ∈ Ui and u' ∈ Uj, i != j then Ui and Uj are monochromatic, merge them

- Based on the fact that K n,m is monochromatic for n >= 3 and m >= 2.
- Assume ui and u' are not in the same part of a matching cut (A, B). Then in {v1, v2, v3} at most 1 can be in A and at most 1 can be in B which contradicts that all of them should belong to one of the 2 partition sets.

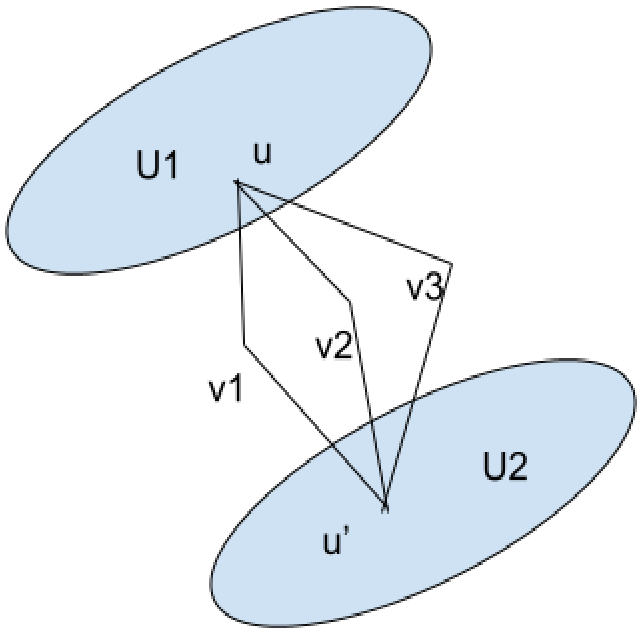


Fig 3 - Rule 3 Visualization

Time to test: O(dc^2*n)

Time to run: O(n)

In the remaining clusters, each cluster having 2 vertices is termed edge cluster and the others are termed non-edge clusters.
A vertex is ambiguous if it has neighbours in Ui and Uj where i != j. A cluster having at least 1 ambiguous vertex is ambiguous
A cluster is fixed if it is contained in N2(Ui) for some Ui

Proposition 2: After applying Rule 1, every non-edge cluster in G is ambiguous or fixed(can be both)

- Every cluster has at least one vertex v that has 2 neighbours in U, in which case it may be to the same U, making it fixed else ambiguous, or a vertex u in Ui has 2 neighbours in C, which makes it fixed.

Rule 4

If 2 clusters are contained in the same N2(Ui), add edges between them.

- This does not affect our results as all these vertices are in the same side of the partition and therefore do not interfere with the matching cut.
- This rule also reduces the number of fixed clusters

At this point, the number of fixed clusters is O(|U|).

Time to test: O(n)
Time to run: O(n^2) - can be done in O(n) by implicitly adding edges

Rule 5

If there is a cluster C with more than 3 vertices that contains a vertex v with no neighbours in U, remove v

- As there are no neighbours in U, this will not affect the resultant matching cut since C remains monochromatic even after removing v.

Time to test: O(dc * n)
Time to run: O(n)

Rule 6

If there is a cluster C with at least 3 vertices, and a monochromatic set Ui where C ⊆ N2 (Ui), remove all edges between them and choose an arbitrary vertex u ∈ Ui and 2 vertices v1, v2 ∈ C, add edges {u, v1} and {u, v2}.

If |Ui| = 2, add an edge between u’ ∈ Ui \ {u} and v3 ∈ C \ {v1, v2}.

If |Ui| > 2, make Ui a clique

- Since we are only adding edges to monochromatic vertices, it does not affect the matching cut that is being obtained.

Time to test : O(dc * n)
Time to run: O(n)

After applying rules 1 to 6, G now is reduced to having

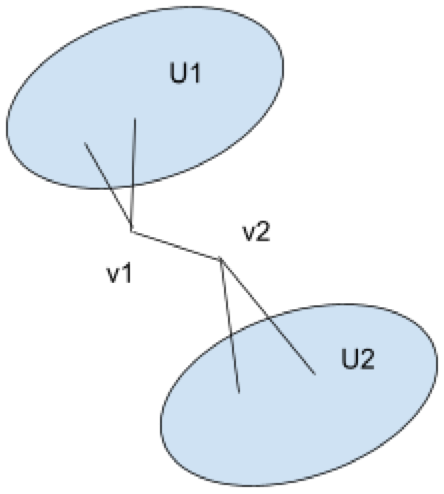
- O(|U|^2) ambiguous vertices - since there can be at most 2 vertices for 2 choices of U - and
- O(|U|^2) non-edge clusters - since there are at most O(|U|^2) ambiguous and O(|U|) fixed clusters - , each containing O(|U|) vertices - a monochromatic set Ui can have at most 2 vertices in each cluster and thus at most 2 * |U| vertices can have neighbours in U.

For edge clusters, we define them simple if for an edge cluster {u, v}, u only has neighbours in Ui and v only has neighbours in Uj.

Rule 7

If there is a simple edge cluster {u, v}, remove u and v from G

- Upon removing, if the matching cut includes the edge {u, v}, since this edge doesn't exist at this point, a matching cut can be found. If it does not, we can safely remove it since it only involves deleting vertices and edges in a monochromatic cluster.



Rule 7 Visualization

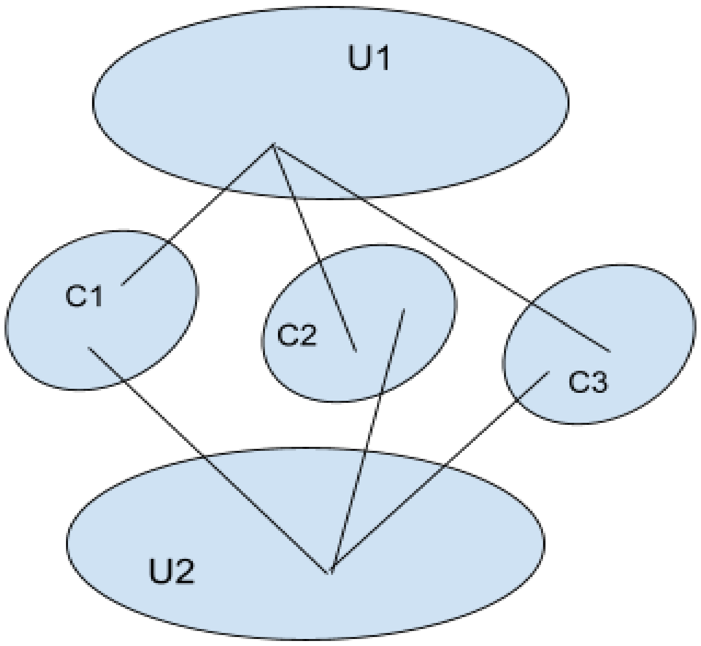
Time to test: O(n+m)
Time to run: O(1)

After applying rules 1 to 7, we obtain a kernel with O(dc(G)^3) vertices, O(dc(G)^2) clusters, each containing O(dc(G)) vertices.
Each edge cluster has at least 1 ambiguous vertex and therefore number of edge clusters is O(dc(G)^2)

Rule 8

If there are 2 vertices u ∈ Ui and u’ ∈ Uj, i != j, and 3 non edge clusters C1, C2 and C3 such that u and u’ has at least 1 neighbour in each of these clusters, merge Ui and Uj.

- For a partition (A, B), we can surely say that C1 and C2 will be in A. Since u has neighbours in C1 and C2, we can say that u has 2 neighbours in A and thus Ui will be in A. Similarly Uj will also be in A. Thus we can conclude that Ui and Uj are on the same part of the cut and monochromatic, thus merging Ui and Uj is safe.



Rule 8 Visualization

Time to test: O(dc^2 * (n + m))
Time to run: O(n)

After applying all 8 Rules, we have O(|U^2|) vertices in V \ U that are in non edge clusters and only have 1 neighbour in U(non ambiguous). - If there are more than 2 neighbours for a pair u, u’ such that those neighbours have only 1 neighbour which is u or u’, it will be reduced by Rule 8.
Therefore this fills the non ambiguous vertices and therefore the number of vertices overall is **O(dc(G)^2) -** O(|U|^2) + O(|U|^2) + O(|U|^2) (Fixed cluster vertices + ambiguous vertices + non-ambiguous vertices)

Running Time Analysis

All the rules can be tested in $O(dc^2 * (n + m))$ time and applied in $O(n)$ time.
Furthermore, Rules can be applied

1. Only once, at the beginning
2. $O(dc)$ times
3. $O(dc)$ times
4. $O(n)$ times
5. $O(n)$ times
6. At most the number of times the other rules are applied
7. $O(n)$ times
8. $O(dc)$ times

And N2(Ui) is calculated after each rule application, taking a running time of $O(dc * (n+m))$

Thus the total running time complexity is $O(dc^3 * n * (n + m))$

Correctness of the Algorithm

As each rule is proved to be safe under each definition, upon rigorous application of the rules, the resulting graph will provide the same result as the original graph. The problem kernel has also been proven to be reduced to **$O(dc(G)^2)$**.
Thus we conclude that this algorithm to reduce the problem kernel is correct.

Links to related documents

Presentation:
https://docs.google.com/presentation/d/1LeWtzYx7r2kEpusAUwd3cTmAoMTyJp0BP380M251Eog/edit?usp=sharing

The document:
https://docs.google.com/document/d/1ii7yvMGJRXF-N5eaUHIymW9HXF5UZ-WLjlIYIOkz4bI/edit?usp=sharing

Implementation:
https://colab.research.google.com/drive/1IO1xTIjbIMKiTQv4jPYI0JSheZuCUyDJ?usp=sharing