Clique Cover Problem

Approximate Algorithms

Group No: 10

Student ID:
1505002
1505044
1505057
1505097
1505101

Polynomial-Time Approximation Scheme

If there is a polynomial-time $\boldsymbol{\delta}$-approximation algorithm with $\boldsymbol{\delta} = 1 + \varepsilon$, for any fixed value $\varepsilon > 0$ to solve a problem, then the problem is said to have a polynomial-time approximation scheme (PTAS)

Clique Cover does not admit a polynomial time approximation scheme unless P=NP [Lund and Yannakakis,1994]

- It follows from known reductions from the Graph Coloring Problem [Simon 1990]

Clique Cover admits a PTAS for Planar Graphs

[Munaro A. 2017]

Notations

G : a planar graph

$\mathbf{L_i}$: the *i*-th layer of *G*

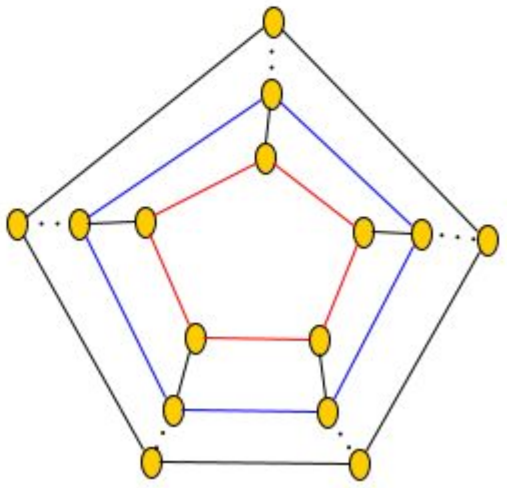
$\mathbf{N(v_i)}$: neighbors of the vertex v_i

V(G) : all vertices of *G*

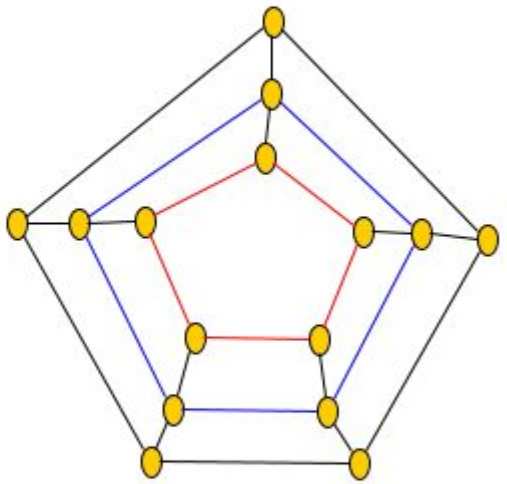
Q : minimum size clique cover of *G*

$\mathbf{G_{ij}}$: subgraph of *G* where layer indices are between

$(j-1)(k-1)+i$ and $j(k-1)+i$ inclusively



K-outerplanar graph : *k-outerplanar graph* is a *planar graph* that has a planar embedding in which the vertices belong to at most *k* concentric layers.



Clique Cover can be solved in polynomial time for k-outerplanar graphs. [M.Rao,2017]

Proof

Here, $v_i \epsilon L_i$ and $N(v_i) \epsilon L_{i-1} \cup L_i \cup L_{i+1}$

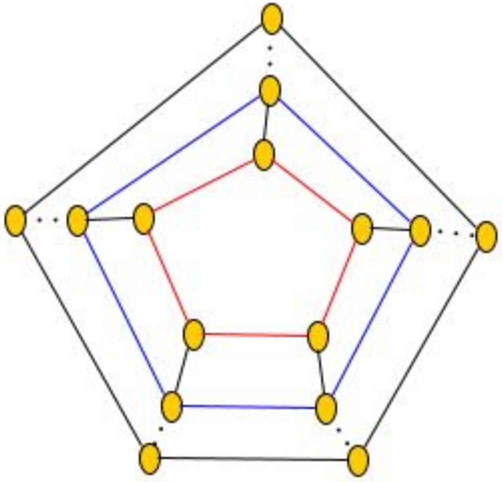
Suppose, $k=\lceil 2/\varepsilon \rceil$,for a given $\varepsilon>0$

G_{ij} is a k-outerplanar graph, for $1 \leq i \leq k$ and $j \geq 1$

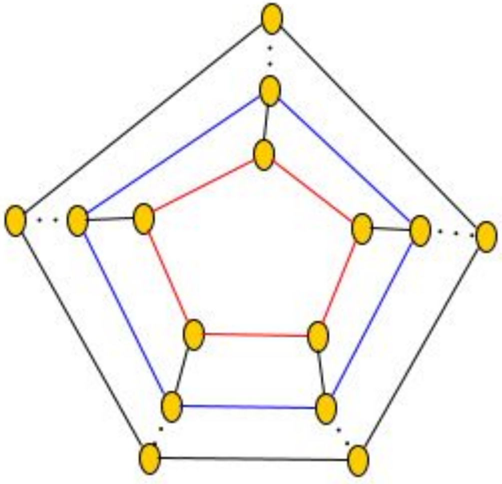
We can set, $C_i = \cup_{j>0} C_{ij}$

Again, $\cup_{j>0} V(G_{ij}) = V(G)$

We will take one with minimum size from all possible C_i



Let, $Q_i \in Q$ where Q_i contains at least one vertex in $\cup_{j=i \bmod k} L_j$



As, $\cup Q_i = Q$, it can be said,

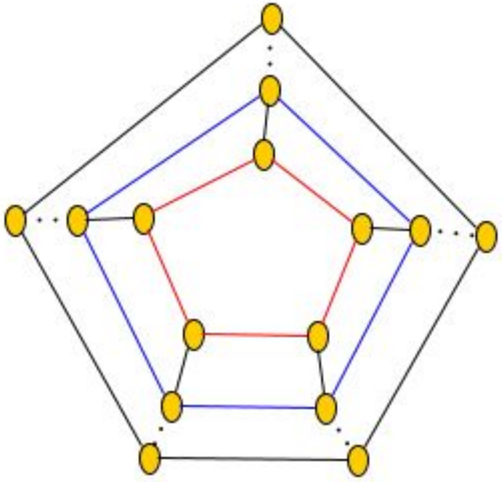
$| Q_l | \leq 2/k \; | Q | \leq \varepsilon \, | Q |$

Let, $Q_{lj} \in Q$ where Q_{lj} contains at least one vertex in $V(G_{ij})$

For each j>0, $| C_{lj} | \leq | Q_{lj} |$ and $\sum | Q_{lj} | = | Q | + | Q_l |$

Finally,

$$
\begin{aligned}
| C_l | &\leq \sum | C_{lj} | \leq \sum | Q_{lj} | \\
&= | Q | + | Q_l | \\
&= | Q | + \varepsilon | Q | \\
&= (1+\varepsilon) | Q |
\end{aligned}
$$



Approximation Algorithm

- In this section, we present approximation algorithms for the k-clique covering problem. We first examine the problem on graphs for the cases k 4 and then for arbitrary k.
- All vertices that are not isolated get covered when all edges are covered. Only isolated vertices may require additional cliques to be covered. For simplicity's sake, we are going to present the algorithms for covering edges only.

Notations

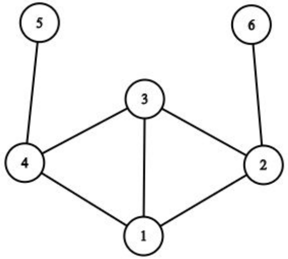
- We use the common notation for a graph G (V, E)
- IVI = n and IEI = m.
- In case G has isolated vertices, **m** denotes the number of edges and isolated vertices.

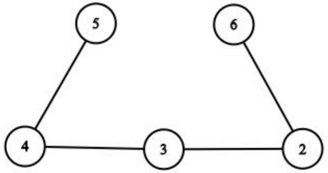
Approximation Algorithm for Cubic Graph

- **Input:** A K_4 free graph G(V,E) with maximum degree 3
- **Output:** A partition into cliques for V with size at most 5/4 of OPT(G).

Definitions

- K_3 is called **triangle**, $K_4 - e$ is called a **diamond** where e is an edge.
- T is denoted as the collection of all triangles in $G(V,E)$, Z_T is $V \setminus V(T)$ and M_{ZT} is the maximum matching of subgraph Z_T
- **Opening a Diamond:**
  (Remove either 1 or 3)





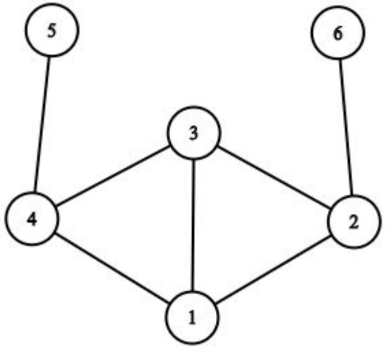
Lemma 1

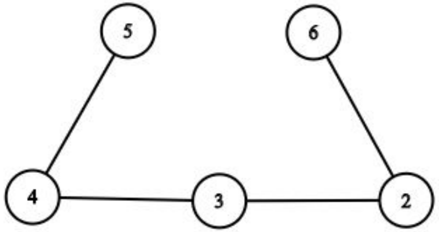
- Denote open(G) as a subgraph of G such that all diamond of G is opened.
- OPT(open(G)) = OPT(G)

Proof of lemma 1

- OPT(open(G)) <= OPT(G) because open(G) is a subgraph of G

- OPT(open(G)) >= OPT(G) because we can add vertex *1* to the clique containing vertex *2* in a partition into cliques for open*(G)* in order to obtain a partition into cliques with same size for *G*





- OPT(open(G)) = { {4,5}, {2,3}, {6} }
- Insert vertex 1 in the clique with vertex 2 thus getting the same size solution for G
- OPT(G) = { {4,5}, {1,2,3}, {6} }

Algorithm for Cubic graph

- Update G by opening all diamonds of G
- Set A_1 as the partition defined the collection T of triangles of G plus the edges of M_{TZ} plus the remaining $|Z_T| - 2|M_{TZ}|$ vertices. Note that $|A_1| = |T| + |Z_T| - |M_{TZ}|$.
- Set A_2 as the partition defined by the edges of M_G, plus the remaining $|V| - 2*|M_G|$ vertices. Note that, $|A_2| = |V| - |M_G|$

Algorithm for Cubic graph (Continue)

- Update A_1 and A_2 with the diamond conversion rule of open(G) to G
- Choose SOL(G) = A_i where $A_i = \min(|A_1|, |A_2|)$

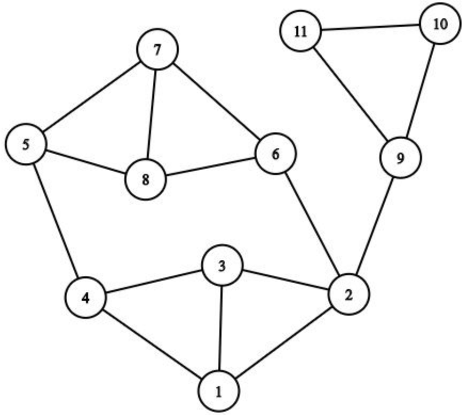


Fig: G

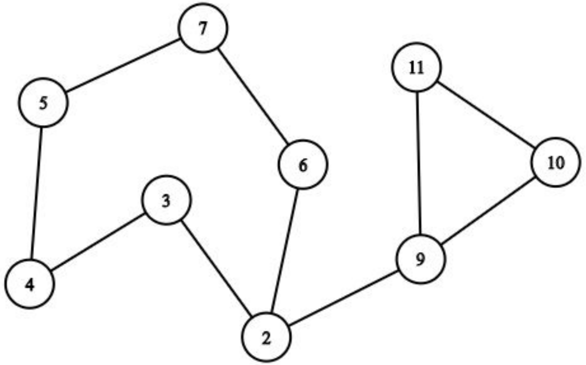


Fig: open(G)

- Now A1 = { {9,10,11}, {2,6}, {5,7}, {3,4} }. With diamond updated, A1 = { {9,10,11}, {2,6}, {5, 7, 8}, {1, 3, 4} }.

- A2 = { {9,10}, {2,6}, {5,7}, {3,4}, {11} }. With diamond updated, A2 = { {9,10}, {2,6}, {5,7, 8}, {1, 3,4}, {11} }

- Since |A1| < |A2|. We output A1 as the final partition.

Proof of Approximation Ratio

The performance ratio of the algorithm **A** is at most 5/4.

Proof(Continue)

Assuming, $Z = Z_T$

$|OPT_{PIC}(G)| > |T| + ⅔ |Z| - ⅓ |M_Z|$ **Theorem.9**

$|A(G)| < |T| + |Z| - |M_z|$

Considering two cases: $|T| \geq ⅔|Z|- ⅓|M_Z|$ or $|T| < ⅔|Z|- ⅓|M_Z|$

1) If $|T| \geq ⅔|Z|- ⅓|M_Z|$ then performance ration,

$R_A = |A(G)| / |OPT_{PIC}(G)|$

$\leq (|T| + |Z| - |M_z|) / (|T| + ⅔ |Z| - ⅓ |M_Z|)$

$\leq 5/4$

Iff $12|T| + 12|Z| - 12|M_z| \leq 15|T| + 10|Z| - 5|M_z|$

$\Leftrightarrow |T| \geq (2/3) |Z| - (7/3) |M_Z|$

But, $|T| \geq ⅔ |Z| - ⅓ |M_Z| \geq ⅔ |Z| - ⅓ |M_Z| - (6/3) |M_Z| = (2/3) |Z| - (7/3) |M_Z|$

Proof(Continue)

2) If $|T| < \frac{2}{3}|Z| - \frac{1}{3}|M_Z|$ then we have two additional cases, $|T^*| \leq |T|/2$ and $|T^*| > |T|/2$

In both case we prove that α is within $|T|/2$

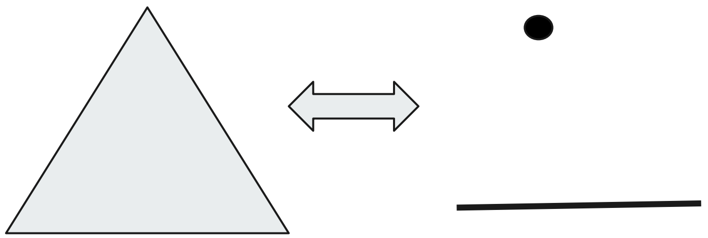
a) If $|T^*| \leq |T|/2$, new partition β ,

$|A_2| \leq |\beta|$

$= |\alpha| + |T^*|$

$= (|T^*| + |Z_{T*}| - |M_{ZT*}|) + |T^*|$

$\leq (|T^*| + |Z_{T*}| - |M_{ZT*}|) + |T|/2$



b) If $|T^*| > |T|/2$, β by adding to α all triangles of $T - T^*$

$|A_1| \leq |\beta|$ where, $|\beta| = (|T^*| + |Z_{T*}| - |M_{ZT*}|) + |T - T^*|$

$\leq (|T^*| + |Z_{T*}| - |M_{ZT*}|) + |T - T^*|$

$< (|T^*| + |Z_{T*}| - |M_{ZT*}|) + |T|/2$

Proof(Continue)

Hence, in either case, α is within $|T|/2$ of $A(G)$,

$$R_A = |A(G)| \,/\, |OPT_{PIC}(G)|$$
$$\le (|T|/2 + |OPT_{PIC}(G)|) \,/\, |OPT_{PIC}(G)|$$
$$< 5/4 \qquad \text{iff } |T| < |OPT_{PIC}(G)| \,/\, 2$$
$$\Leftrightarrow 2\,|T| < |OPT_{PIC}(G)|$$

Approximation Algorithm for k-Clique Cover

- We present a linear time $(\frac{1}{2} + \frac{1}{k-1})$k-approximation algorithm (H) for the k-clique covering problem.
- Algorithm (H) finds a cover with star-shaped components. Phase 1 of (H) finds a sequence of vertices and edges, such that each edge and each vertex occurs once, and such that all edges in the sequence between consecutive vertices V_i and V_{i+1} are incident with V. The isolated vertices are appended at the end of this sequence.
- In phase 2 the sequence is broken into batches of size at most k- 1. Such a batch of k- 1 edges and vertices induces a graph on at most k vertices, and forms a clique.

Approximation Algorithm for k-Clique Cover

ALGORITHM (H).

Input: Graph $G = (V, E)$.

Phase 1: Set $SEQ = \varnothing$, $\overline{V} = V$, $\overline{E} = E$

while $\overline{V} \neq \varnothing$ **do begin**

Select a vertex $v \in \overline{V}$

Set $SEQ = (SEQ, \{v\})$

repeat

Find a vertex $u \in V \setminus \overline{V}$ such that $(u, v) \in \overline{E}$

Set SEQ = (SEQ ,{u, v});

$\overline{E} \leftarrow \overline{E} \setminus \{(u,v)\}$;

until no such u can be found;

end

Approximation Algorithm for k-Clique Cover

Let $SEQ = S1, S2, \ldots, S_N$, where $N = |E| + |V|$

Phase 2: Set $C^H = \varnothing$, $\bar{k} = 0$

while $\bar{k} < N$ **do begin**

Let C denote the component with vertex set $V(C) = S_{\bar{k}+1} \cup \ldots S_{\bar{k}+k-1}$ and edge set $E(C) = \{S_{\bar{k}+1}, \ldots, S_{\bar{k}+k-1}\} \cap E$;

$C^H \leftarrow C^H \cup C$

$\bar{k} = \bar{k} + k - 1$

End

Output $z_H = |C^H|$ as the number of cliques used by the algorithm

End of (H).

Complexity

THEOREM : (H) is a linear time $(\frac{1}{2} + \frac{1}{k-1})$-approximation algorithm.

Proof. Phase 1 can be realized in linear time by a simple breadth-first search. Phase 2 is obviously linear in the length of the sequence, O(m).

Let z^H be the number of cliques used for covering edges by heuristic (H), and let z^* be the number used by an optimal clique covering. Let w denote the number of isolated vertices, V the set of non-isolated vertices, and E the set of edges. Then,

$$z^H = \left\lceil \frac{|E| + |V| + w}{k-1} \right\rceil$$

Complexity

THEOREM : (H) is a linear time $(\frac{1}{2} + \frac{1}{k-1})$-approximation algorithm.

and,

$$kz^* \geq w + \sum_{v \in V} \left\lceil \frac{\delta(v)}{k-1} \right\rceil,$$

as each vertex ***v*** of degree ***δ(v)*** occurs in at least

$$\left\lceil \frac{\delta(v)}{k-1} \right\rceil$$

cliques.

Complexity

THEOREM : (H) is a linear time $(\frac{1}{2} + \frac{1}{k-1})$-approximation algorithm.

The latter number is bounded from below by $w + \frac{2|E|}{k-1}$, and also by *w + IVI* follows that

$$z^H \leq \frac{|E| + |V| + w}{k-1} + 1$$
$$\leq \frac{1}{2}kz^* + \frac{1}{k-1}kz^* + 1.$$

Complexity

THEOREM : (H) is a linear time $(\frac{1}{2} + \frac{1}{k-1})$-approximation algorithm.

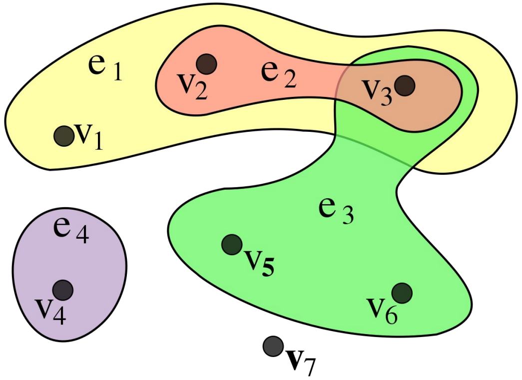
The bound is tight, as can be seen by taking as input a graph consisting of *N* k-cliques. Each clique has *k/(k-1)* edges and k vertices. The algorithm partitions these edges and vertices in

$$\left\lceil N(\frac{k}{2} + \frac{k}{k-1}) \right\rceil$$

batches of size *k- 1*, whereas the optimal cover uses N cliques.

k-Clique Covering of Hyperedges of Size k-1

- First let us learn a bit about hypergraphs.
- A hypergraph is a generalization of a graph in which an edge can join any number of vertices. In contrast, in an ordinary graph, an edge connects exactly two vertices.



An example of a hypergraph, with
V = $\{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$ and
E = $\{e_1, e_2, e_3, e_4\} = \{\{v_1, v_2, v_3\}, \{v_2, v_3\}, \{v_3, v_5, v_6\}, \{v_4\}\}$.

k-Clique Covering of Hyperedges of Size k-1

Algorithm:

Input: $V, E = \{E_1, \dots, E_m\}$.

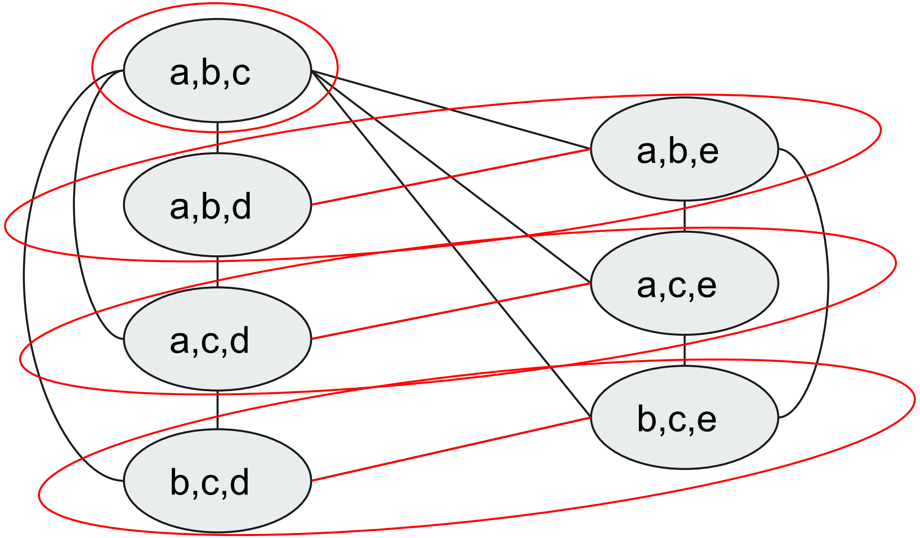
Step 1: Construct the following undirected graph $\mathbf{G(\bar{V},\bar{E})}$. Each vertex represents a hyperedge. Two vertices i and j are connected by an edge if and only if the corresponding hyperedges **i** and **j** form a ***spanning pair***(Two hyperedges (elements) $\mathbf{E_i}$ and $\mathbf{E_j}$ are called a spanning pair if $\mathbf{|E_i \cup E_j| = k}$).

Step 2: Find a ***maximum cardinality matching*** in **G**.

Step 3: Put each pair of hyperedges corresponding to the end vertices of a matching edge obtained in step 2 in a single clique. Put the remaining unmatched vertices(hyperedges) in separate cliques. Output the collection of cliques $\mathbf{C_H}$.

k-Clique Covering of Hyperedges of Size k-1

$V = \{a, b, c, d, e\}$
$E = \{\{a, b, c\}, \{a, b, d\}, \{a, c, d\}, \{b, c, d\}, \{a, b, e\}, \{a, c, e\}, \{b, c, e\}\}$



k-Clique Covering of Hyperedges of Size k-1

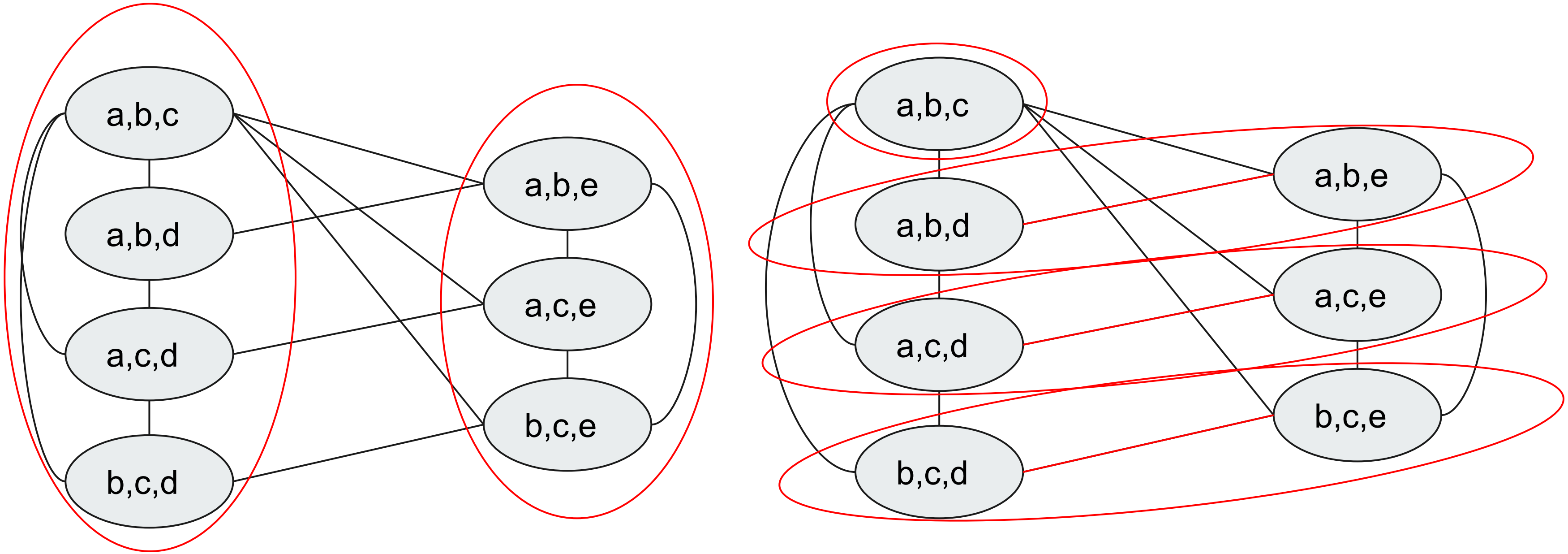


Fig: *Ideal Case* vs *Approximation*

k-Clique Covering of Hyperedges of Size k-1

Approximation Ratio:

- Since for two hyperedges to be packed together, their corresponding vertices must be connected by an edge in **G**. It suffices to prove the bound for any connected component of **G**, let us assume that **G** is connected.
- Let $\mathbf{t_i}$ be the number of hyperedges packed in the *i*th clique of the optimal solution. These $\mathbf{t_i}$ hyperedges must form a $\mathbf{t_i}$-clique in **G**. There exists a feasible matching with $\lfloor \mathbf{t_i}/2 \rfloor$ edges in each of these $\mathbf{t_i}$-cliques. If $\mathbf{t_i}$ is odd, then the number of sets required to cover these nodes is less than or equal to $\lceil \mathbf{t_i}/2 \rceil$. Hence step 3 results in a collection of $\sum^{z^*}_{i=1}$ cliques. Let $z^H = |C^H|$ and $\mathbf{z^*}$ be the optimal number of cliques.
- Since $\mathbf{t_i} <= \mathbf{k}$, $\quad \mathbf{z^H} <= \sum^{z^*}_{i=1} \lceil \mathbf{t_i}/2 \rceil <= \mathbf{z^*} \lceil \mathbf{k}/2 \rceil$.
- So the algorithm can be considered an $\lceil \mathbf{k}/2 \rceil$-approximation.

k-Clique Covering of Hyperedges of Size k-1

Complexity:

- The complexity of the algorithm is dominated by the step 2 of the algorithm that is the maximum cardinality matching.
- The maximum cardinality matching can be done in **$O(\sqrt{V}E)$**.
- So the complexity of our algorithm can be considered as such.

That’s all. Thank You. Any Questions?