Maximum Cut in Hyperbolic Random Graphs

Janosch Ruff

June 3, 2022

1 Maximum Cut Optimization Problem

Definition 1 *Let $G = (V, E)$ be a Graph with $V = \{1, 2, .., n\}$ and*

$$E_{i,j} := \begin{cases} 1 & \text{if } i \sim j, \\ 0 & \text{otherwise.} \end{cases}$$

Let the number of edges be denoted by $|E(G)| = m$ and the number of vertices $|V(G)| = n$. Then the Maximum cut is the set $\mathcal{S}^* \subseteq V(G)$ such that

$$\mathcal{S}^* = \max_{\mathcal{S} \subseteq V(G)} \sum_{e_{i,j} \in E} \chi_{i,j} \tag{1}$$

where

$$\chi_{i,j} := \begin{cases} 1 & \text{if } i \in \mathcal{S} \wedge j \notin \mathcal{S}, \\ 0 & \text{otherwise.} \end{cases}$$

2 Probabilistic Method for Maximum Cut

Algorithm 1 randomized_maxcut(G)

```
//Initialize set S
S ← ∅
for v ∈ V do
    //Flip a fair random coin
    r ← coin_flip(1/2)
    //Include v in S if flipped head
    if r = 1 then
        S ← S ∪ v
    end if
end for
return S
```

where $\mathbb{P}[coin\_flip(p) = 1] = p$ and $\mathbb{P}[coin\_flip(p) = 0] = 1 - p$.
Hence, the algorithm flips a fair coin for each $v \in G(V)$ in order to decide if $v \in \mathcal{S}$.

Proposition 1 *Let X be the number of edges between $\mathcal{S}_r \subseteq V(G)$ and $\mathcal{T}_r = V(G)/\mathcal{S}_r$ where $\mathcal{S}_r$ = randomized_maxcut(G). Then $\mathbb{E}[X] \geq \frac{1}{2}|\mathcal{S}_e^*|$ where $|\mathcal{S}_e^*|$ denotes the number of edges included in connecting the two disjoint vertex sets $\mathcal{S}^* \subseteq V(G)$ and $\mathcal{T}^* = V(G)\backslash\mathcal{S}^*$.*

Proof. We define a random indicator variable $X_{i,j}$ indicating if an edge is included in our cut

$$X_{i,j} := \begin{cases} 1 & \text{if } (i \in \mathcal{S} \wedge j \notin \mathcal{S}) \vee (i \notin \mathcal{S} \wedge j \in \mathcal{S}), \\ 0 & \text{otherwise.} \end{cases}$$

Clearly $\mathbb{P}[X_{i,j} = 1] = \underbrace{\mathbb{P}[i \in \mathcal{S}]}_{\frac{1}{2}}\underbrace{\mathbb{P}[j \notin \mathcal{S}]}_{\frac{1}{2}} + \underbrace{\mathbb{P}[i \notin \mathcal{S}]}_{\frac{1}{2}}\underbrace{\mathbb{P}[j \in \mathcal{S}]}_{\frac{1}{2}} = \frac{1}{2}$.

Hence, for the expected size of X we get

$$\mathbb{E}[X] = \mathbb{E}[\sum_{e\{i,j\}\in E} X_{i,j}] = \sum_{e\{i,j\}\in E} \mathbb{E}[X_{i,j}] = m\mathbb{E}[X_{i,j}] = \frac{m}{2} \geq \frac{|\mathcal{S}_e^*|}{2}.$$

■

3 d-Wise Independent Values Derandomization

Definition 2 *Let $\mathcal{U}$ and $\mathcal{Z}$ be finite sets. Further, let $\mathcal{F}$ be an index set of a family of functions such that*

$$\{h_f : \mathcal{U} \to \mathcal{Z}\}_{f\in\mathcal{F}}.$$

$\{h_f\}_{f\in\mathcal{F}}$ is a universal family of hash functions if $\forall \alpha, \beta \in \mathcal{Z}, \forall x, y \in \mathcal{U}, x \neq y$,

$$\mathbb{P}_{f\in\mathcal{F}}[h_f(x) = \alpha \wedge h_f(y) = \beta] = \frac{1}{|\mathcal{Z}|^2}.$$

This is similar to the definition of pairwise independence.

Definition 3 *A collection $\{X_i\}_{i=1}^n$ of (discrete) random variables over the same probability space is pairwise independent if $\forall a, b, \forall i \neq j$ holds*

$$\mathbb{P}[X_i = a \cap X_j = b] = \mathbb{P}[X_i = a]\mathbb{P}[X_j = b].$$

Similar the collection is d-wise independent, if $\forall \mathcal{I} \subseteq \{1, 2, .., n\}, |\mathcal{I}| \leq d, \forall a_{i\in\mathcal{I}}$ holds

$$\mathbb{P}[\bigcap_{i\in\mathcal{I}} X_i = a_i] = \prod_{i\in\mathcal{I}} \mathbb{P}[X_i = a_i].$$

Consider a prime q such that $2^m < q2^{m+1}$. Pick $a, b \in \mathbb{F} := \mathbb{Z}_q$ uniformly at random and define

$$h_{(a,b)}(x) \equiv a \cdot x + b \pmod{q}. \tag{2}$$

Proposition 2 *$\{h_{a,b}\}_{a,b\in\mathbb{Z}_q}$ is a universal family of hash functions.*

Proof. Since $\mathbb{Z}_q$ is a finite field we have $\forall i, j \in \mathbb{F}$

$$\mathbb{P}[h_{a,b}(i) = z] = \frac{1}{q}$$

which shows that all values are uniformly distributed.
For the independence consider $i \neq j$ and $f_{a,b}(i) = z_1, f_{a,b}(j) = z_2$. This is similar to the linear equation system

$$\begin{aligned} a \cdot i \quad &+b \quad = z_1 \\ a \cdot j \quad &+b \quad = z_2. \end{aligned}$$

Since $i \neq j$ the determinant of this matrix is not 0. Hence, there is a unique solution what yields

$$\mathbb{P}[f_{a,b}(i) = z_1 \cap f_{a,b}(j) = z_2] = \frac{1}{q^2}$$

which concludes proof for pairwise independence of $\{h_{a,b}\}_{a,b\in\mathbb{Z}_q}$. ∎

Remark. We can get d-wise independence by generalizing $f_{a,b}(x)$. This is achieved by the polynomial $f_{a_{d-1},..,a_1,a_0}(x) = a_{d-1}x^{d-1} + .. + a_1x + a_0$. We picked a_i uniformly at random from a field $\mathbb{Z}_q$. Since $\mathbb{F} = GF(2^m)$ is also a field we might also pick uniformly at random from $GF(2^m)$.

Next, we derandomize our randomized algorithm to find a maximum cut.

Algorithm 2 derandomized_maxcut(G)

```
1:  S ← ∅
2:  //pick m such that 2^m ≥ n > 2^(m-1)
3:  m ← ⌈log|V(G)|⌉
4:  //pick random a and b within our field for the universal hash function
5:  a, b ← random(0, 2^m)
6:  for i = 0,1,..,n-1 do
7:      //get the first bit of our hashed integer
8:      r ← a·i + b (mod 2)
9:      if r = 1 then
10:         S ← S ∪ v_i
11:     end if
12: end for
13: return S
```

where $random(\ell, k)$ in line 5 returns an integer j uniformly at random within the interval $j \in [\ell, k)$.

Corollary 1 *The derandomized_maxcut(G) algorithm uses $2\log n$ random bits whereas the randomized_maxcut(G) algorithm uses n random bits achieving the same amount of randomness.*

Proof. To implement $random(\ell, k)$ consider the following procedure.
Use $r_i = coin\_flip(1/2)$ for $i = 0, 1, .., m$ and we set $a = a_0 a_1 .. a_m$ where $a_i = r_i$ giving us our random number $a \in [0, 2^m)$ using m bits where $m = \log n$. In similar fashion we get a random integer $a \in [0, 2^m)$ with $\log n$ bits hence, we require $2 \log n$ bits.
The second statement of the corollary about the amount randomness follows immediately from Proposition 2 while the amount of random bits for the other randomized MAXCUT is a consequence of flipping a random coin for each vertex and $|V(G)| = n$. ■

Observation 1 *We get $|\{\{a, b\} : a, b \in GF(2^m)\}| < 4n^2$ or in other words there are less than $4n^2$ different choices for $\{a, b\}$.*

Using observation 1 we can examine all cuts given by the pairs $\{a, b\}$ in polynomial time. One of this cuts has the property $|\mathcal{S}_e^{a,b}| \geq \frac{|\mathcal{S}_e^*|}{2}$. This gives us a deterministic 2-Approximation.

4 Greedy Heuristic for Maximum Cut (Conditional Probabilities Derandomization)

We develop a greedy heuristic by based our randomized max cut approach.
$N(v)$ in line 6 denotes the neighbors of v hence, $N(v) \coloneqq \{u \in V(G) : u \sim v\}$.

Algorithm 3 greedy_maxcut(G)

```
S ← ∅
//Iterate (in any order) through all vertices in G
for v ∈ V do
    k ← 0
    //Iterate (in any order) through all neighbors of v
    for u ∈ N(v) do
        //Count the number of neighbors that are included in S
        if u ∈ S then
            k ← k + 1
        end if
    end for
    //Include v in S if the number of edges between v and S\V(G) is small
    if k/|N(v)| ≤ 1/2 then
        S ← S ∪ v
    end if
end for
return S
```

Proposition 3 *Let $S_g \subseteq V(G)$ be the subset of vertices returned by greedy_maxcut(G). Further, let K be the number of edges between S_g and $V(G)\backslash S_g$. Then $K \leq 2|\mathcal{S}_e^*|$ holds for any Graph G.*

Proof. We analyze the greedy algorithm based on the randomized version introduced before. The following observation expresses the result for the *randomized_maxcut*.

Observation 2 *Before starting the randomized procedure we have $\mathbb{E}[X] \geq \frac{1}{2}|\mathcal{S}_e^*|$.*

Next, we would like to express the expectation of X conditioned on the result after the first iteration step.

Observation 3 *By total law of probability we have for the expected value of X*

$$\mathbb{E}[X] = \frac{1}{2}(\mathbb{E}[X|v_1 \in S] + \mathbb{E}[X|v_1 \notin S]).$$

Based on observation 3 we get by induction for each iteration step $j < n$

$$\mathbb{E}[X|\phi(v_1, v_2, .., v_{j-1})] = \frac{1}{2}(\mathbb{E}[X|\phi(v_1, v_2, .., v_{j-1}), v_j \in S] + \mathbb{E}[X|\phi(v_1, v_2, .., v_{j-1}), v_j \notin S]) \tag{3}$$

where $\phi(\cdot)$ denotes the assignment already picked for the vertices with index smaller j. This results in a tree like structure the algorithm can branch into:

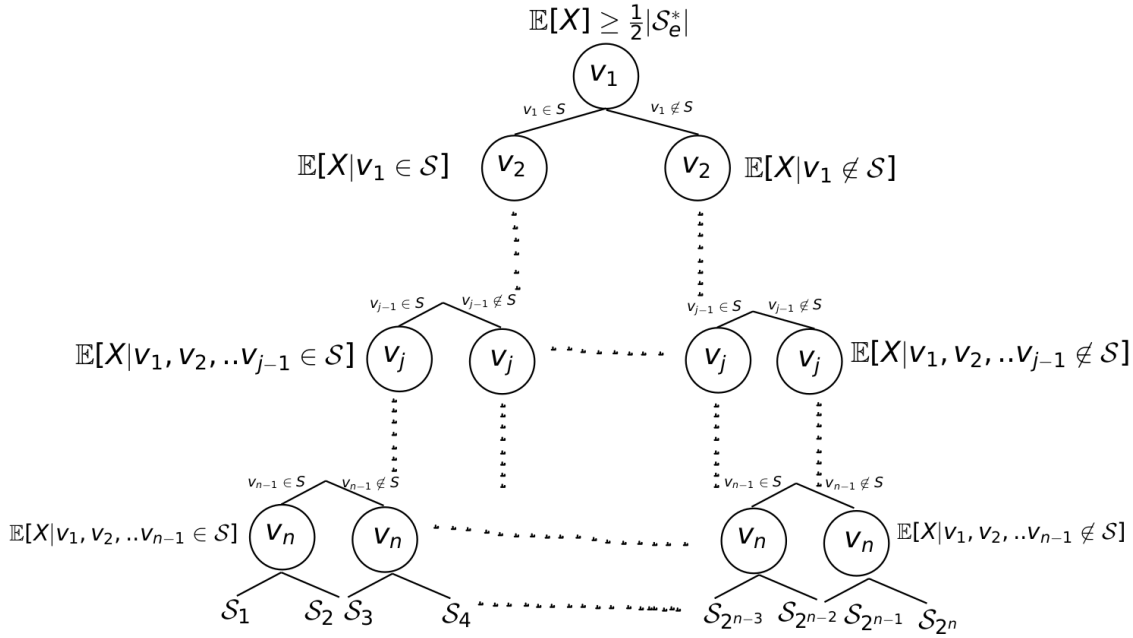

Figure 1: Tree representing branches and the expected size of Maximum cut.

The tree has depth n and there are 2^n leaves (for each possible subset $\mathcal{S} \subseteq V(G)$ we have a leaf).

Thus, the leaves represent the complete solution space for a Maximum Cut. Starting to traverse from top to bottom starting at the root, we can always branch into a child of $j-1$ such that

$$\mathbb{E}[X|\phi(v_1, v_2, .., v_j)] \geq \mathbb{E}[X|\phi(v_1, v_2, ..v_{j-1})] \tag{4}$$

by equation 3. Hence, we always pick the assignment $\phi(v_j)$ satisfying 3 and recursively repeat this process until we arrive to a leaf $\mathcal{S} \subseteq V(G)$ indicating our solution what concludes our greedy approach.
It is left to show that the greedy approach indicated by the branching of conditional probabilities is indeed the same as the greedy approach of our $greedy\_maxcut(G)$ algorithm. For this observe that the condition

$$\frac{k}{|N(v)|} \leq \frac{1}{2}$$

in line 13 holds exactly if inequality 4 is met for $v_j \in S$. ■

Remark. The performance of the greedy algorithm depends of the ordering of the vertices. There is a greedy path that is optimal.

5 SDP Approximation for Maximum Cut

We express our maximum value function 1 slightly different by

$$\mathcal{S}^* = \max_{\mathcal{S}} \sum_{1 \leq i < j \leq n} \frac{1 - y_i y_j}{2} \tag{5}$$

where $y_i \in \{\pm 1\} \forall i \in V(G)$ and the sign is given by

$$y_i := \begin{cases} +1 & \text{if } i \in \mathcal{S}, \\ -1 & \text{otherwise.} \end{cases}$$

Thus, $y_i \forall i$ is a one dimensional vector (or rather scalar) of unit norm and

$$1 - y_i y_j = \begin{cases} 1 & \text{if } i \in \mathcal{S} \wedge j \notin \mathcal{S}, \\ 0 & \text{otherwise} \end{cases}$$

works as XOR counting the number of edges between $V(\mathcal{S})$ and $V(G) \backslash V(\mathcal{S})$.
Next, we relax 5 to

$$\hat{\mathcal{S}} = \max_{\mathcal{S}} \sum_{1 \leq i < j \leq n} \frac{1 - y_i^T y_j}{2} \tag{6}$$

for $1 \leq i \leq n$ where $y_i \in s_m$ such that $s_m := \{x \in \mathbb{R}^m : \|x\|_2 = 1\}$ and $m \in \mathbb{N}$ denotes the dimension (check the figure below for an example).

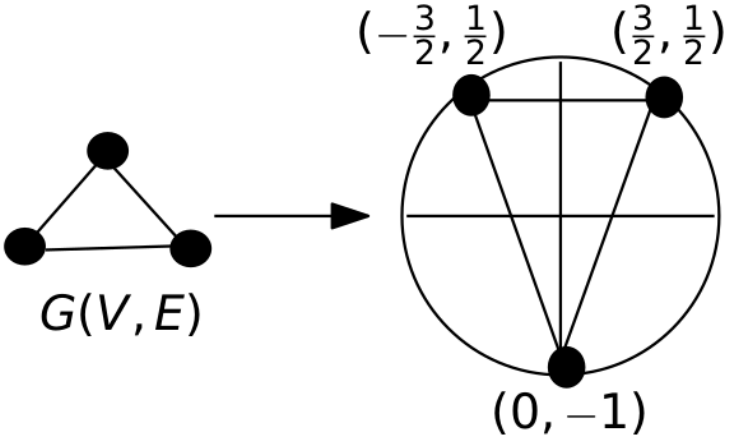

Figure 2: Example of the embedding for m = 2 and a triangle graph.

We have for the vevtor $y_i = \{\begin{bmatrix}1\\0\\\vdots\\0\end{bmatrix}, \begin{bmatrix}-1\\0\\\vdots\\0\end{bmatrix}\}$ just like for the scalar $y_i = \{\pm 1\}$. The advantage of 6 over 5 is, that we can solve it in polynomial time. For this we consider the matrix

$$A^{n\times n} := (y_i^T y_j)_{i,j=1,2,..,n}.$$

A is symmetric positiv semidefinite $\iff \exists B$ such that $A = B^T B$.
With this in hand we can express 6 as

$$\hat{\mathcal{S}} = \max_{\mathcal{S}} \sum_{1\le i<j\le n} \frac{1 - a_{i,j}}{2} \tag{7}$$

where we have for the diagonal $a_{i,i} = 1$ and A symmetric positive semidefinite. The advantage of 7 is obvious by the following lemmata:

Lemma 1 *The optimization problem in 7 can be resolved in polynomial time.*

Lemma 2 *If A is symmetric positive semidefinite we can retrieve $B^T B = A$ in polynomial time using Cholesky Decomposition.*

The lemmata imply that we can retrieve $\hat{y}_i$ from $\hat{\mathcal{S}}$ in polynomial time. Since the each retrieved vector $\hat{y}_i \in s^n$ lies on the unit sphere it is not necessarily a vector satisfying $y_i = \begin{bmatrix}\pm 1\\0\\\vdots\\0\end{bmatrix}$ thus we have to round the values of our vector $\hat{y}_i$ to decide whether we include vertex y_i in $\mathcal{S}$.

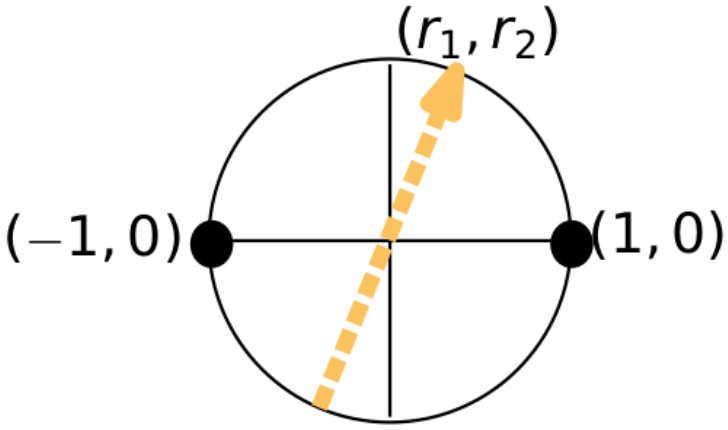

Figure 3: Unit circle with all vertices $(1,0) \in \mathcal{S}$ and $(-1,0) \notin \mathcal{S}$ seperated by the random vector $r = (r_1, r_2)$.

The error of this process is summed up in the following theorem.

Theorem 1 *$\hat{\mathcal{S}}$ approximates $\mathcal{S}^*$ such that*

$$|\hat{\mathcal{S}}_e| \geq 0.878 |\mathcal{S}_e^*|. \tag{8}$$

where $\hat{\mathcal{S}}_e$ denotes the edges included in the cut of $\hat{\mathcal{S}}$.

Proof. We define a random vector $r = (r_1, r_2, .., r_n)$ where $r_i \sim \mathcal{N}(0,1)$ i.i.d for all $1 \leq i \leq n$. Then assign $y_i \in \{\pm 1\}$ to $\hat{y_i}$ by

$$y_i = \begin{cases} 1 & \text{if } \hat{y_i} \cdot r \geq 0, \\ -1 & \text{otherwise.} \end{cases}$$

The expected number of edges cut by this procedure is given by

$$\mathbb{E}[X] = \sum_{\{i,j\} \in E} \mathbb{P}[y_i \neq y_j].$$

So we need to compute the probability of the event that $\{i,j\} \in E$ is cut by r. Let $\theta_{i,j}$ be the angle between $\hat{y_i}$ and $\hat{y_j}$. Then we get by the projection of r onto the plane $\overrightarrow{\hat{y_i}\hat{y_j}}$

Claim 1

$$\mathbb{P}[\hat{y_i} \sim \hat{y_j} \textit{is separated by } r] = \frac{\theta(i,j)}{\pi}.$$

On the other hand we have to divide by $|\mathcal{S}_e^*|$. An edge $\{i,j\} \in \mathcal{S}_e^*$ has to satisfy that the angle is π. Hence we can express $|\mathcal{S}_e^*|$ by

$$|\mathcal{S}_e^*| = \sum_{\{i,j\} \in E} \frac{1 - \cos(\theta(i,j)}{2}.$$

Putting everything together yields:

$$\frac{\mathbb{E}[|\hat{\mathcal{S}}_e|]}{|\mathcal{S}_e^*|} = \frac{\sum_{\{i,j\}\in E} \frac{\theta(i,j)}{\pi}}{\sum_{\{i,j\}\in E} \frac{1-\cos(\theta(i,j)}{2}} \geq \inf_{\theta\in[0,\pi]} \frac{\frac{\theta(i,j)}{\pi}}{\frac{1-\cos(\theta(i,j)}{2}} \geq 0.878. \tag{9}$$

■

Remark. There is no 0.941 factor approximation algorithm unless $P = NP$. Assuming the Unique Games Conjecture there is no better approximation scheme that achieves a better approximation factor than 0.878 in polynomial time.

6 Reduction rules for Maximum Cut

We denote some safe reduction rules for Maxium Cut.

Reduction Rule 1 *Let $H \subset G$ be a connected component of $G\backslash\{v\}$ for some vertex $v \in G(V)$ such that $G[H(V) \cup \{v\}]$ is a clique K_k where $|H(V) \cup \{v\}| = k$. Then we can set $F(V) = G(V)\backslash H(V)$ and $S^* \subseteq V(G)$ is the same as $S^* \subseteq F(V)$*

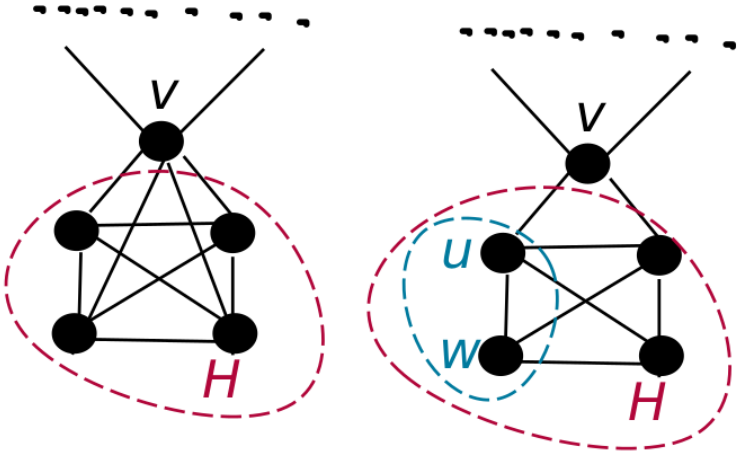

Figure 2: Reduction Rule 1 (left) where $G[H(V) \cup v]$ forms a clique K_5. Since $\forall u \in H(V)$ and also $v \in K_5$ we can remove the rest of the clique H.
Reduction Rule 2 (right) where H forms a clique K_4 but $v \notin H$. If there exists a combination of vertices $u, w \in H$ they can be included in S^*. Note that $v \not\sim w$.

Reduction Rule 2 *Let $H \subset G$ such that $H = G\backslash\{v\}$ is a Clique K_k. Let $\{u, w\} \in H(v)$ such that $G\backslash\{u, w\}$ is a connected component and $u \sim v$ but $u \not\sim v$. Then $\{u, v\} \in S^*$.*

Reduction Rule 3 *Let $u, v, w \in V(G)$ form a path such that $\{u, v\} \in E(G)$ and $\{v, w\} \in E(G)$ but $\{u, w\} \notin E(G)$. Additionally let $H = G\backslash\{u, v, w\}$ be a single component. Then $\{u, v, w\} \in S^*$.*

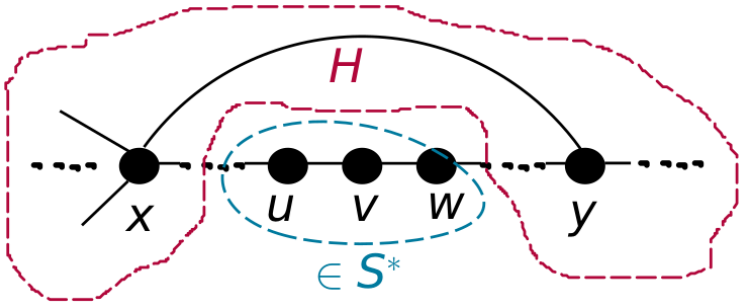

Figure 3: u, v, w form a path in G such that after deleting the path the prior component is still connected with each other via $x \sim y$.

Reduction Rule 4 *Let $u, v \in V(G)$ such that $u \not\sim v$ and $G \backslash \{u, v\}$ collapses in two connected components H and F. If $G[V(H) \cup \{v\}]$ and $G[V(H) \cup \{u\}]$ form cliques, then $u, v \in S^*$ and reduce G to $G \backslash H$.*

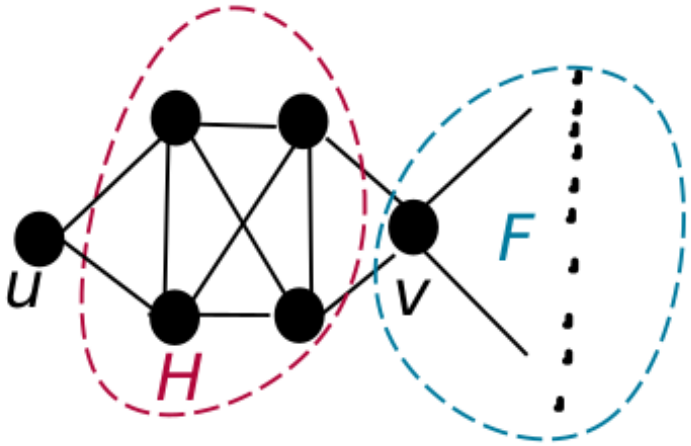

Figure 4: u and v form a two cliques with $H(V)$ such that $u, v \in S^*$ and we can reduce the problem to $F \subset G$.