Vertices removal for feasibility of clustered spanning tree

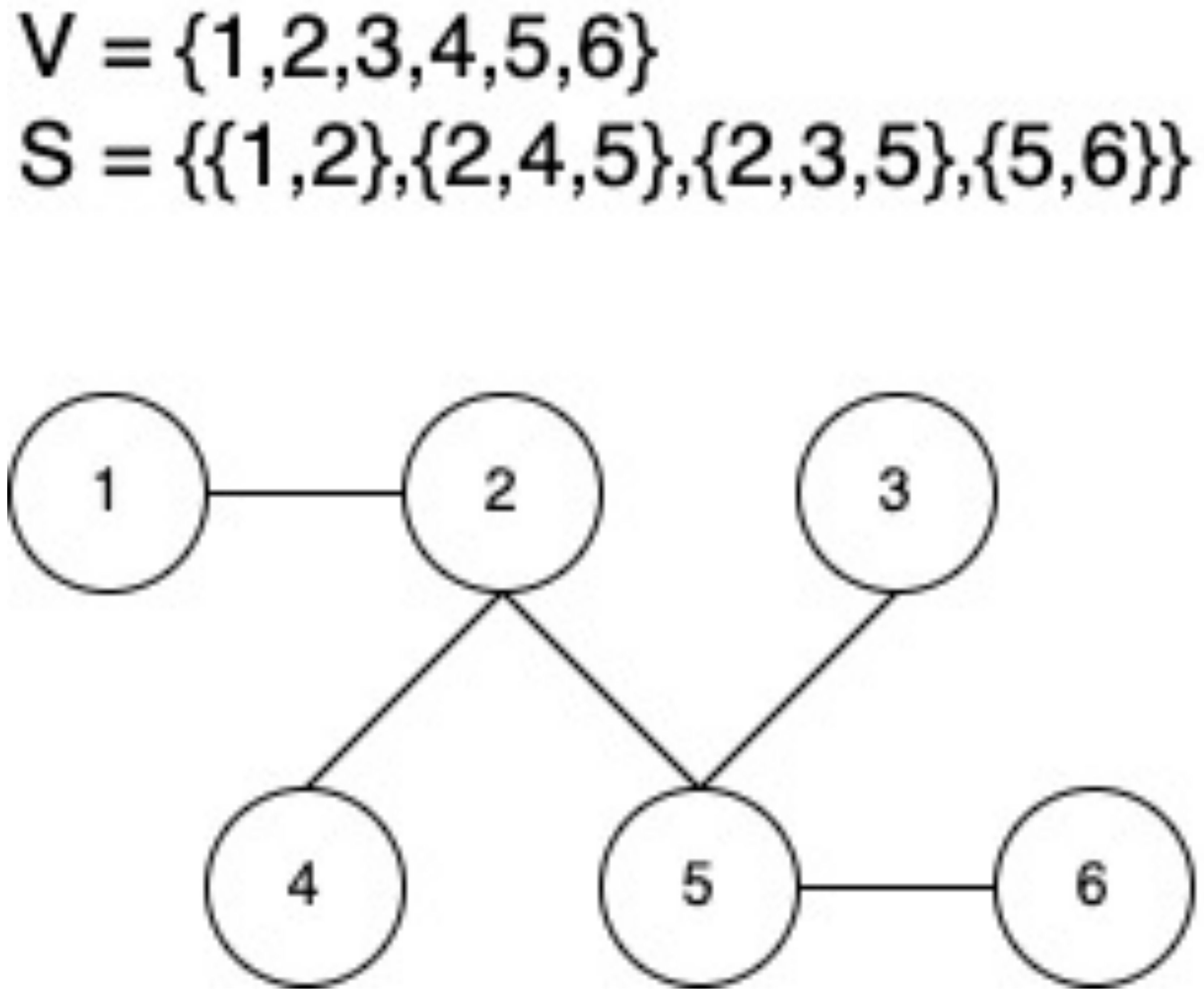
Nili Guttmann-Beck, Roni Rozen, Michal Stern: Vertices removal for feasibility of clustered spanning trees. Discret. Appl. Math. 296: 68-84 (2021)

Outline

- Cluster spanning tree
- Problem description
- Reduction graph
- Find removal list and solution tree
- Minimal removal list
- Non disappearing vertices and clusters
- Conclusion

Cluster spanning tree

- $H = < V, S >$ is a hypergraph
- V is a set of vertices, S is a set of clusters
- The Cluster Spanning Tree problem is to find a tree spanning all vertices in V which satisfies that each cluster S_i induces a spanning tree if it exists
- application
  - communication network
  - databases with synchronous replications
  - key management for secure group communications

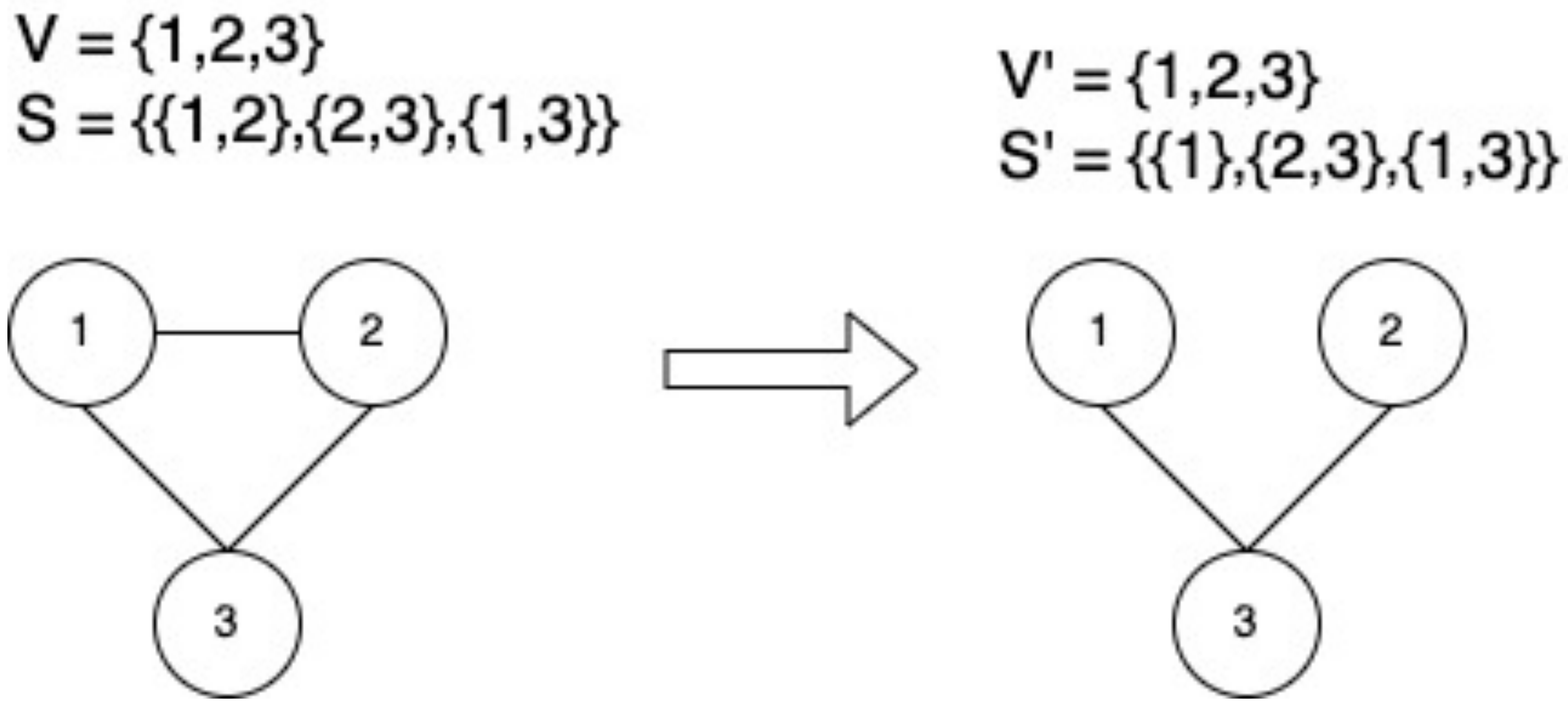


Problem description

- If the given hypergraph H does not have feasible solution tree, remove some vertices from some clusters to gain feasibility

- L is a removal list of H if L is a list of pairs :
  $L = \{(v_1, S_{i1}), \cdots, (v_k, s_{ik})\}$ with $v_j \in S_{ij}$

- Check for feasibility
  - Helly property
  - cordiality and acyclicity
  - ES algorithm

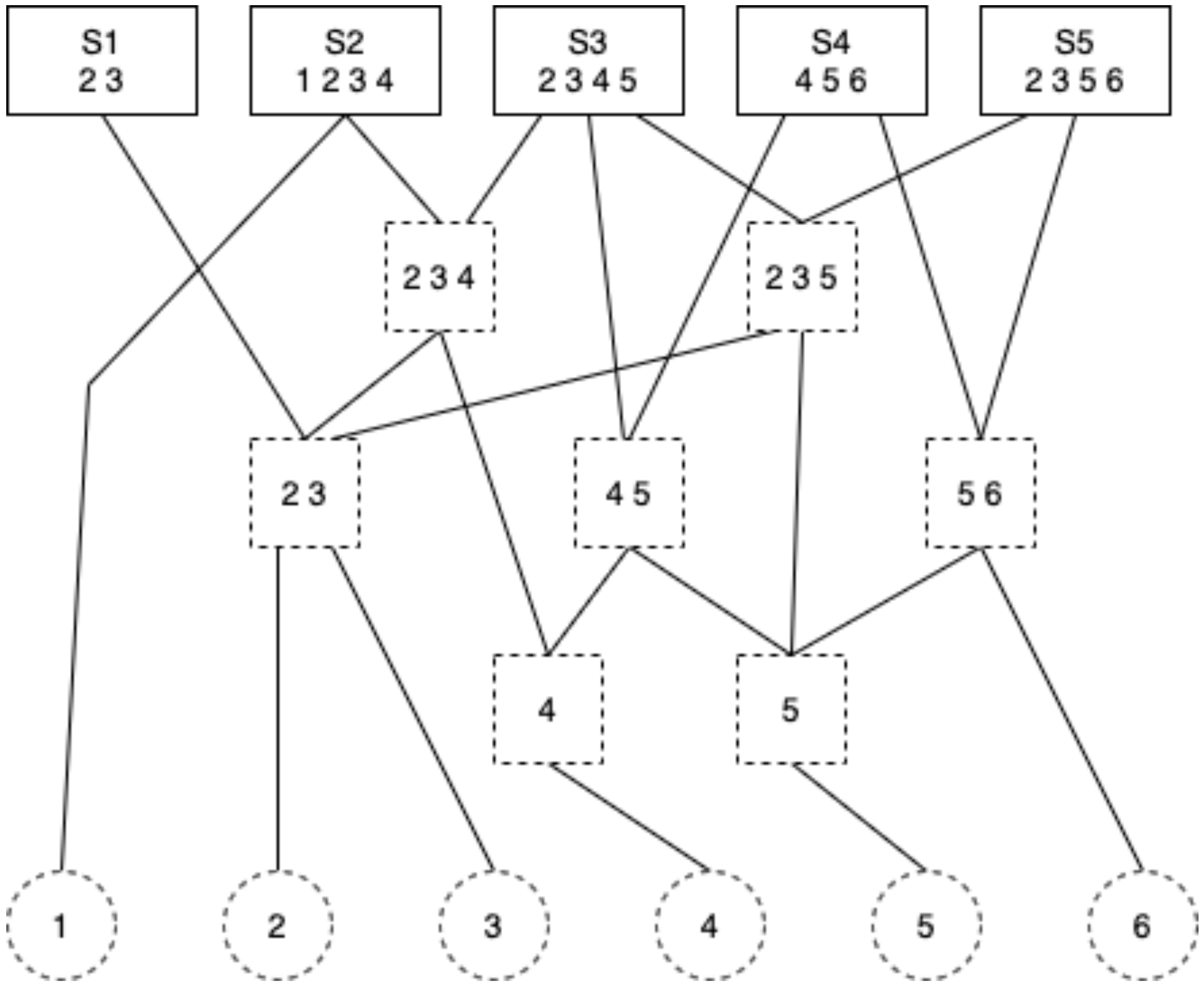


Reduction graph

- Given hypergraph $H = < V, S >$, its reduction graph $G^r = < V^r, E^r >$
- V^r contains
  - Cluster node s_i , each denoted corresponded cluster $S_i \in S$
  - Intersection node x, denoted corresponded intersection set $X = \cap_{S_i \in S'}, S' \subseteq S \ and \, |S'| > 1$
  - Vertex node v_i, each denoted corresponded vertex $V_i \in V$
- E^r contains
  - edge (x_1, x_2) if $X_1 \subseteq X_2$ and there is no intersection set X' that $X_1 \subseteq X' \subseteq X_2$
  - edge (v, x) if $v \in X$ and there is no intersection set X' that $v \in X' \subseteq X$

Reduction graph

- Example:
- $H = < V, S >$,$V = \{1,2,3,4,5,6\}$, $S = \{S_1, S_2, S_3, S_4, S_5\}$
- 右圖則為H的reduction graph $G^r = < V^r, E^r >$
- 方形實線 : cluster node
- 方形虛線 : intersection node
- 圓形虛線 : vertex node



Find removal list

- Accessible : given $G^r = < V^r, E^r >$ and its subgraph G'. $x, w \in V^r$, w is accessible to x in G' if the path between x and w in G' only use set nodes contained in x.

- For any set node x , $A(G', x)$ is a set of vertices accessible to x in G'

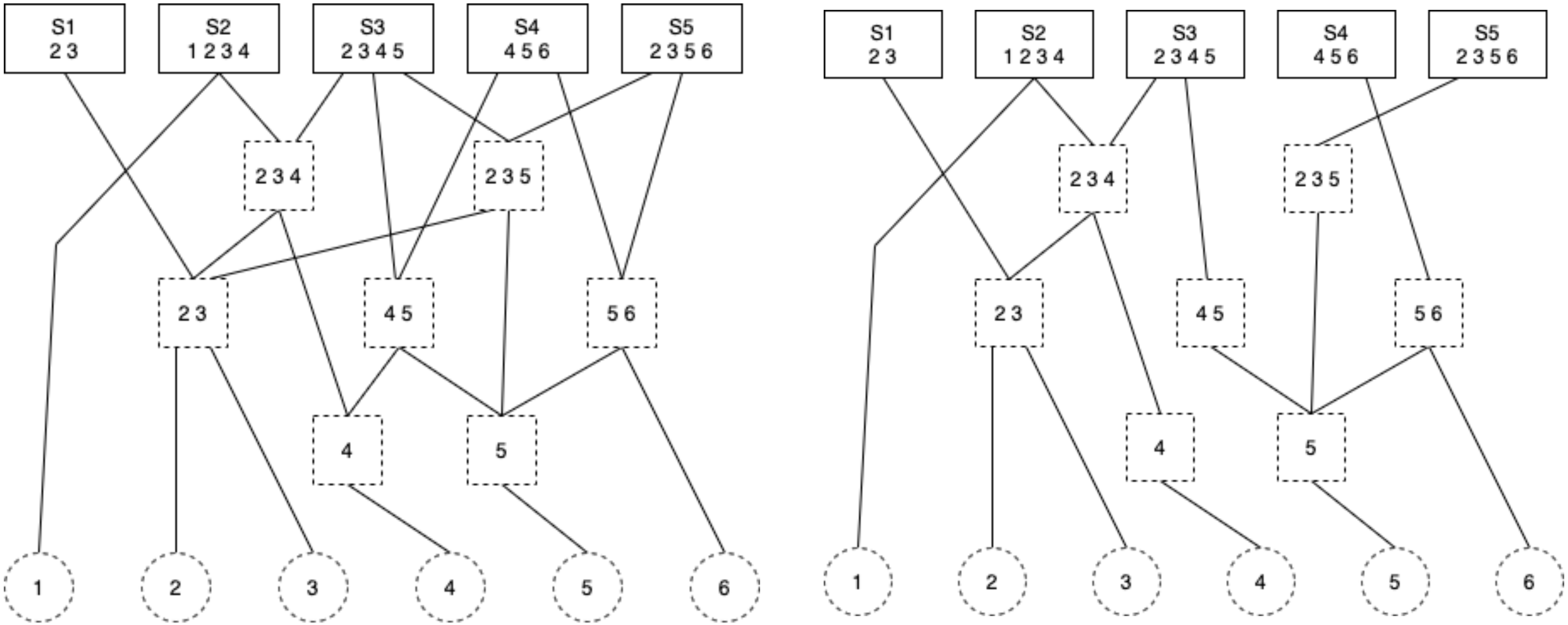
- To get a feasible removal list

  1. Find a spanning tree T^r on G^r

  2. For every cluster S_i , if vertex $v_i \in S_i$ and $v_i \notin A(T^r, S_i)$, add (v_i, S_i) into removal list L

Example

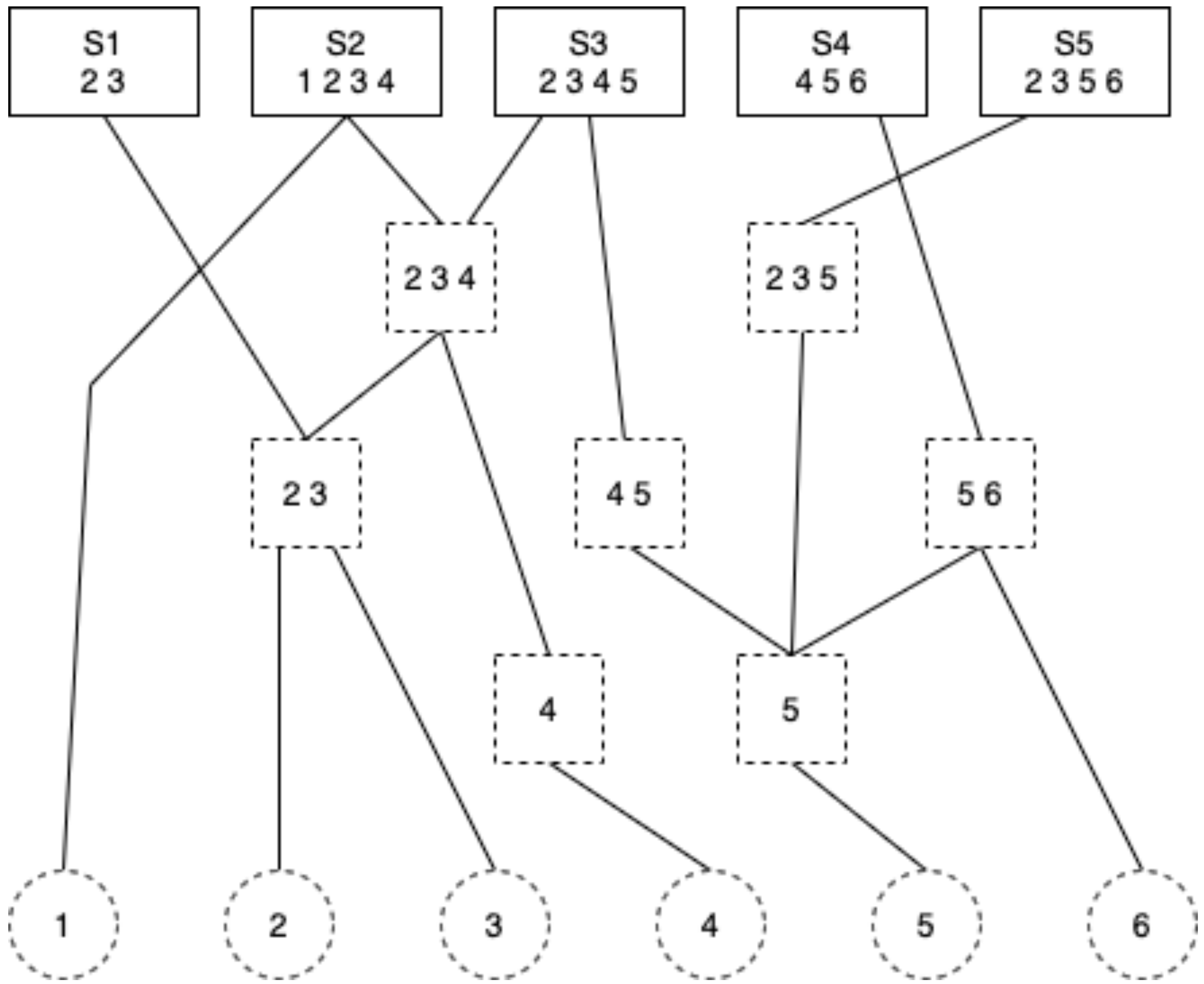
- 右圖為G^r中的一個spanning tree T^r，feasible removal list $L = \{(4,S_4), (2,S_5), (3,S_5)\}$

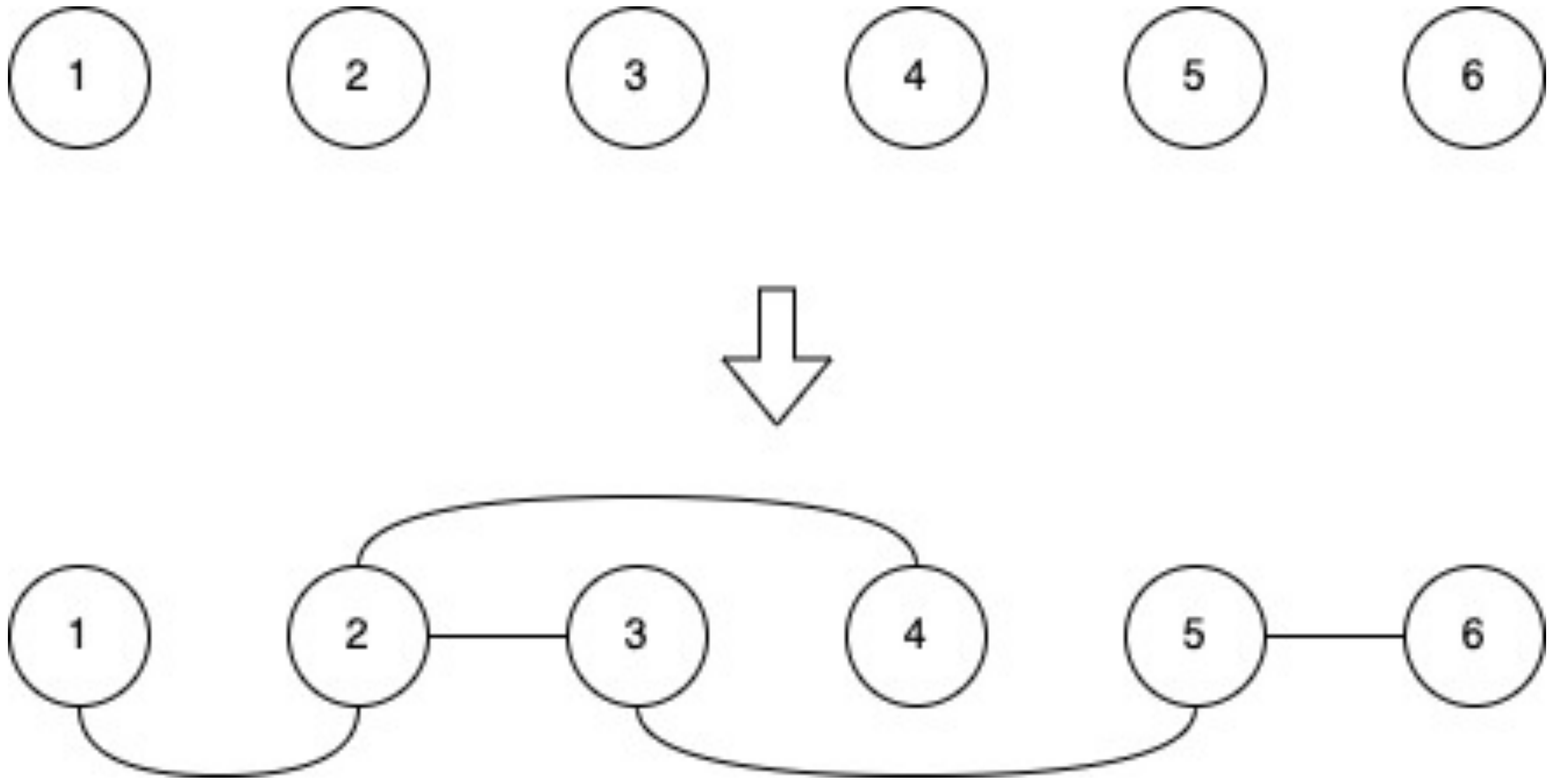


Find feasible solution tree

- Initialize solution tree F contained V without edges
- Scan T^r from bottom to top, according to the size of node
- For each set node X
    1. Add edge to create subtree spanning isolated vertices of $A(T^r, X)$ in F
    2. Connected all connected component of $A(T^r, X)$, using edge whose both endpoint are in $A(T^r, X)$, without creating cycle
- If V still not connected after scanning all set node, add arbitrarily edge to connect them without creating cycle

Example


S1
2 3
S2
1 2 3 4
S3
2 3 4 5
S4
4 5 6
S5
2 3 5 6
2 3 4
2 3 5
2 3
4 5
5 6
4
5
1
2
3
4
5
6


1
2
3
4
5
6

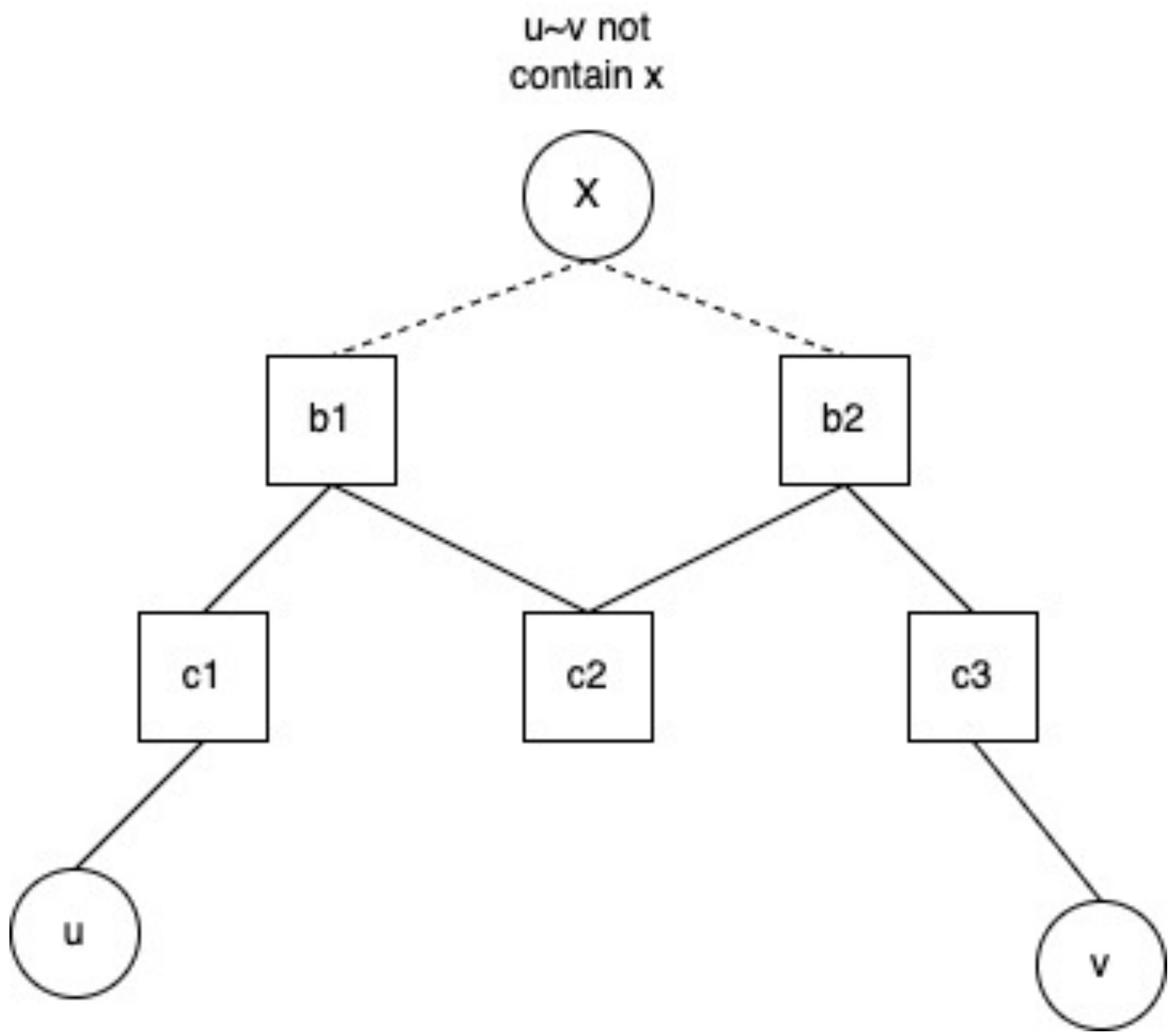
Proof

- 當我們 scan 到 set node X 的時候，如果 F 中$A(T^r, X)$已經形成好幾個 subtree，則對 X 中所有點 u,v ，兩者之間的path只會使用$A(T^r, X)$中的點。

- 假設F中存在一條u-v的路徑使用$A(T^r, X)$以外的vertices，在T^r上 u~X 和 v~X 的 path 的聯集會包含一條 u~v 的path

  1. u~v 的path中不包含X

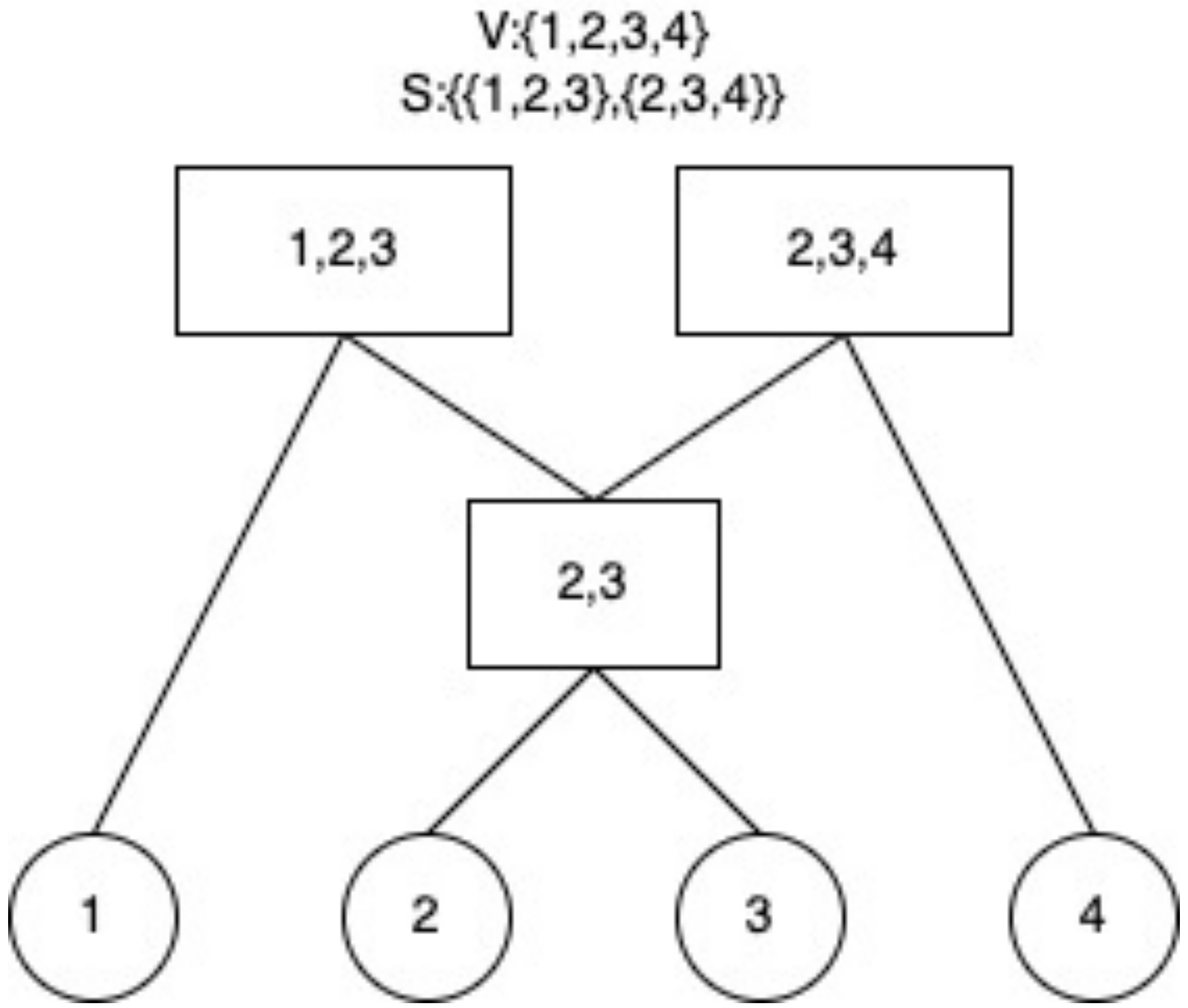
  2. u~v 的path中包含X



Minimal removal list

- Consistency : for $H = < V, S >$ and removal list L，L is consistent with respect to intersection set X if for each vertices $u, v \in X$，if $(u, S_i) \in L$ then $(v, S_i) \in L$

- 圖中$L_1 = \{(2,S_1),(3,S_1)\}$對intersection set {2,3}是consistency

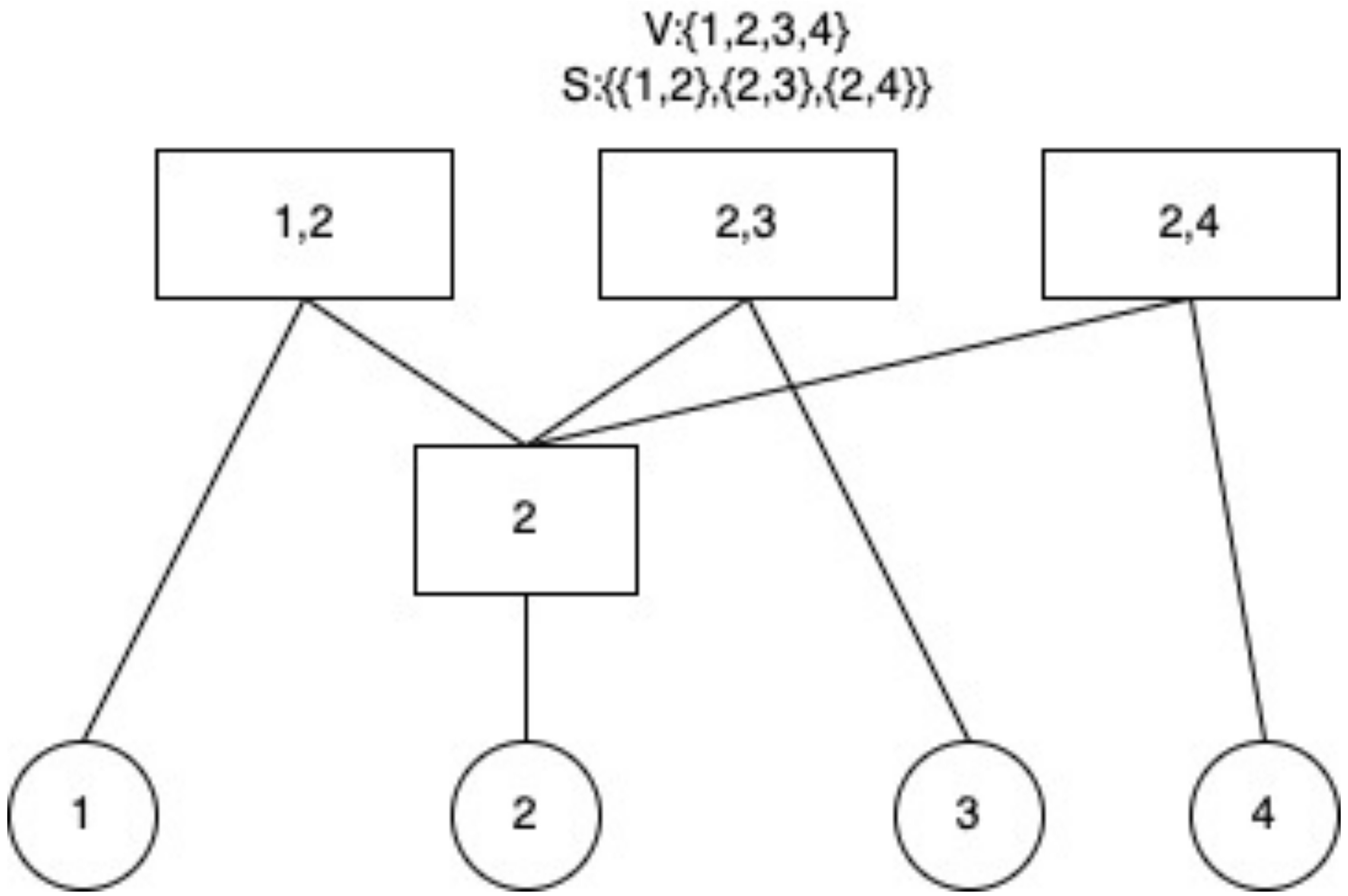


Minimal removal list

- If L is feasible removal list, and vertex $v \in$ intersection set $X \in \cap_{i=1}^{l} S_i$, and v is exist in H\L, then $L' = L \setminus \{(u, S_{i_j}) \,|\, u \in X/\{v\}, j = \{1, \cdots, l\}\}$ is also feasible removal list
- Therefore, if L is a minimal removal list, then for any intersection set $X \in \cap_{i=1}^{l} S_i$, $L' = L$, otherwise L in not minimal.
- This means that if we don't remove vertex $v \in X$ from cluster S_i , then we don't remove other vertices $u \in X$ of S_i
- If a feasible removal list is minimal, it is AllXconsistent

Minimal removal list

- Slender : given G^r，each intersection nodes X_1,X_2 do not contain others
- Check whether a hypergraph H has a slender reduction graph : for each cluster S_1,S_2,S_3 ，$S_1 \cap S_2 \cap S_3 = \phi$ or $S_1 \cap S_2 \cap S_3 = S_1 \cap S_2 = S_2 \cap S_3 = S_1 \cap S_3$
- If the reduction graph G^r is slender, then we can find a minimal removal list

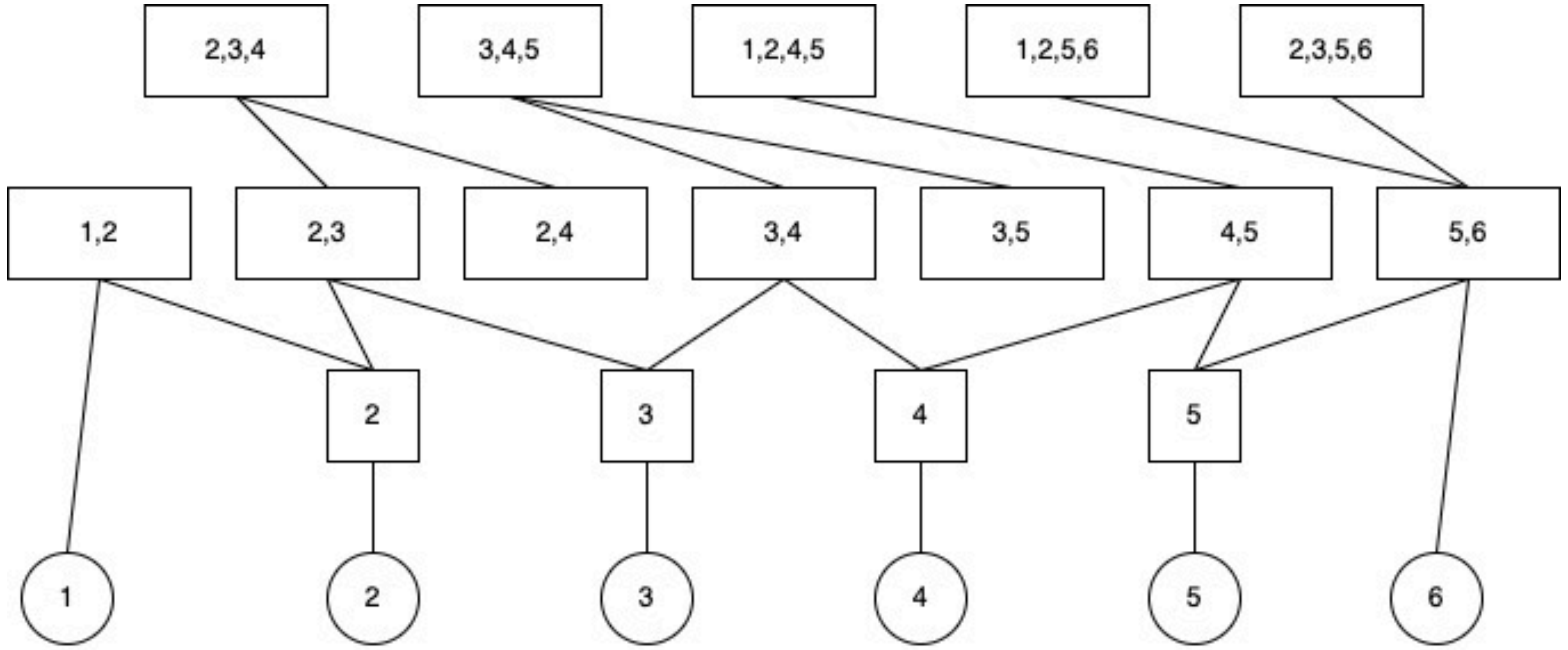


Minimal removal list

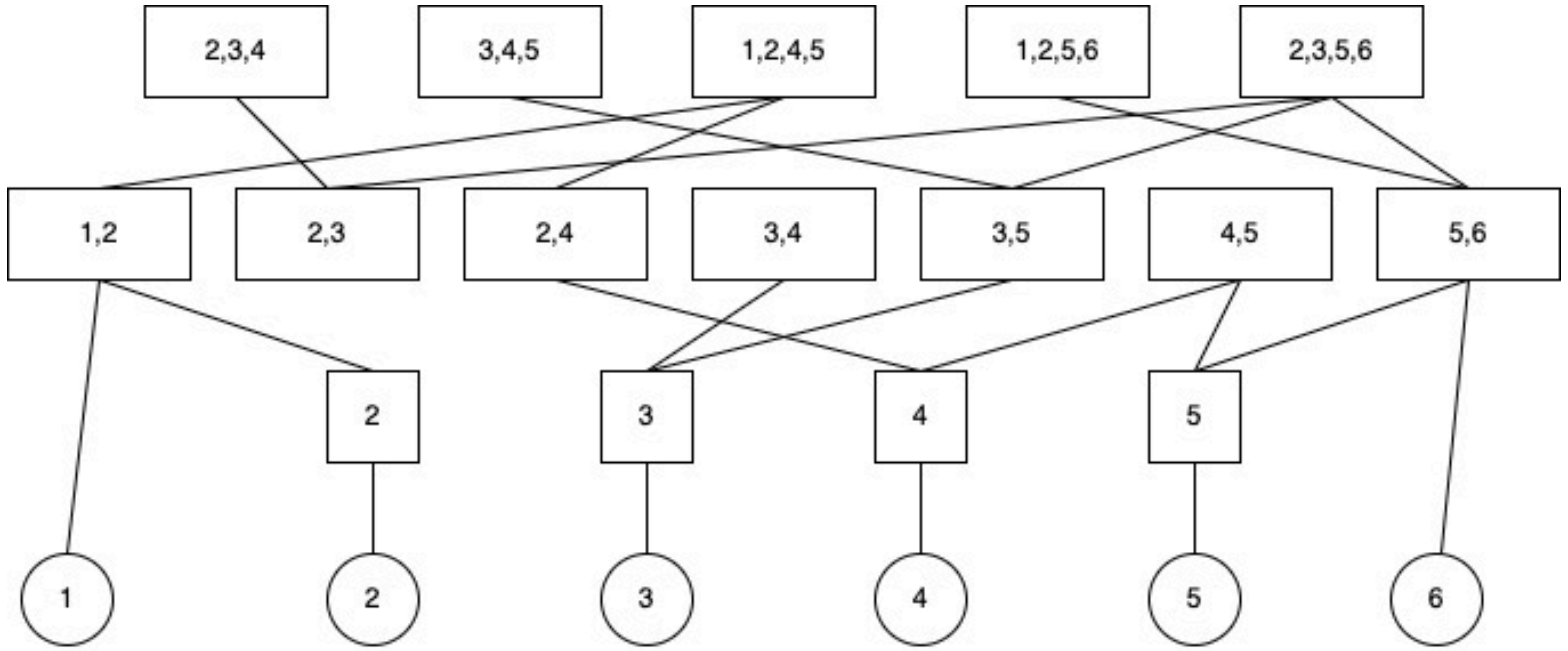
1. Given $H = < V, S >$, $G^r = < V^r, E^r >$. Define edges in E^r with weight
   - w(v, y) = 0 for every vertex node v and every set node y that $(v, y) \in E^r$.
   - w(x, s_i) = |X| for every intersection-node x and every cluster-node s_i that $(v, y) \in E^r$, where X is the intersection set which corresponds to x.
2. Find maximum spanning tree T^r on G^r
3. The size of removal list L = W(G^r) - W(T^r)
   - If edge $(X, S_i) \in E(G^r)/E(T^r)$, all of vertices in X are not accessible to S_i , otherwise G^r is not slender. So it will create |X| vertices removal.

Non disappearing vertices and clusters

- The result solution tree is depend on the spanning tree we found in reduction graph
- Sometimes it would cause disappearance of vertices or clusters



L = {(1, S3), (2, S3), (1, S4),(2,S4),(2,S5),(3,S5)}



L ={(2,S1),(3,S1),(4,S1),(4,S2),(5,S2),(1,S4),(2,S4),(2,S5)}.

Non disappearing vertices and clusters

- Non vertex disappear :

  1. Initial V(T^r) and E(T^r) with empty

  2. For each vertex node or intersection node u, find another node w, which has larger size than u , in V(T^r) that (u, w) exist in E^r, add (u, w) into E(T^r) and add $\{u, w\}$ into V(T^r)

  3. For each cluster node u, if u not in V(T^r), find an intersection node w in V(T^r) that (u, w) exist in E^r, add (u, w) into E(T^r) and add $\{u, w\}$ into V(T^r)

- Because for each node u, it has at least one edge to a bigger set node that contain u, so each vertex node has at least one path to a cluster, which means it would appear at least once.

Non disappearing vertices and clusters

- Non cluster disappear :

  1. Initial V(T^r) and E(T^r) with empty

  2. For each set node u, find another node w, which has smaller size than u, in V(T^r) that (u, w) exist in E^r, add (u, w) into E(T^r) and add $\{u, w\}$ into V(T^r)

  3. For each vertex node u, if u not in V(T^r), find an intersection node w in V(T^r) that (u, w) exist in E^r, add (u, w) into E(T^r) and add $\{u, w\}$ into V(T^r)

- Because for each node u, it has at least one edge to a smaller set node that contain u, so each cluster node has at least one path to a vertex, which means each cluster would contain at least one vertex.

Conclustion

- If a hypergraph cannot find a cluster spanning tree，we can remove some vertices from some cluster

- To find a removal list，we need the reduction graph G^r and a spanning tree T^r on G^r

- Different T^r may produce different removal list and different solution tree

- If we want minimal removal list -> use maximum spanning tree on G^r

- If we want removal list that don’t make vertices or clusters disappear

  ->use correspond spanning tree