Spanning Trees with Few Branch Vertices in Graphs of Bounded Neighborhood Diversity

L.Gargano A.A.Rescigno

Department of Computer Science
University of Salerno

SIROCCO, June 5-10, 2023

Minimum Branch Vertices Problem

- Given an undirected graph $G = (V, E)$, where
- V represents the set of the members of the network
- E represents the relationships among them
- A *branch vertex* is a vertex having degree at least three

Minimum Branch Vertices Problem

- Given an undirected graph $G = (V, E)$, where
- V represents the set of the members of the network
- E represents the relationships among them
- A *branch vertex* is a vertex having degree at least three

The Minimum Branch Vertices Problem is defined as follows

Minimum Branch Vertices (MBV)

Instance: A connected graph $G = (V, E)$.
Goal: Find a spanning tree of G having the smallest number $b(G)$ of branch vertices among all the spanning trees of G.

Minimum Branch Vertices Problem

- Given an undirected graph $G = (V, E)$, where
- V represents the set of the members of the network
- E represents the relationships among them
- A *branch vertex* is a vertex having degree at least three

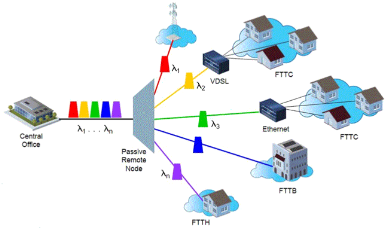
The Minimum Branch Vertices Problem is defined as follows

MINIMUM BRANCH VERTICES (MBV)

Instance: A connected graph $G = (V, E)$.
Goal: Find a spanning tree of G having the smallest number $b(G)$ of branch vertices among all the spanning trees of G.

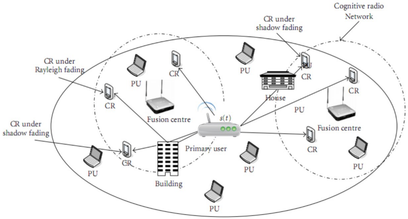
Since a spanning tree without branch vertices is a Hamiltonian path of G, we have $b(G) = 0$ if and only if G admits a Hamiltonian path.

- Wavelength-division multiplexing (WDM) technology
  - One wants to minimize the number of lightsplitting switches in a light-tree

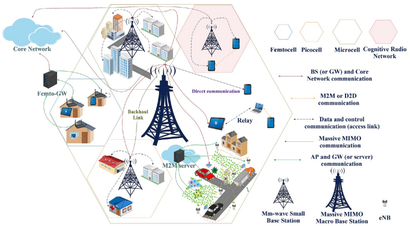


Networks operating with a wide range of frequencies

- Cognitive Radio Networks



- 5G technologies



Previous Works

- Gargano *et al.* [2002] proved it is NP-complete to decide whether, given a graph G and an integer k, G admits a spanning tree with at most k branch vertices, even in cubic graphs.
- Salomon [2005] proved
  - the existence of an algorithm that finds a spanning tree with $O(\log |V(G)|)$ branch vertices whenever the degree of each vertex of the input graph is $\Omega(|V(G)|)$;
  - an approximation factor better than $O(\log |V(G)|)$ would imply that $NP \subseteq DTIME(n^{O(\log \log n)})$
- Heuristics and approximation results for MBV have been presented from then on [Chimani et al. 2015, Marin 2015, Landete et al. 2017, etc.]

Parameterized Complexity

Fixed Parameter Tractable (FPT)

A problem with input size n and parameter p is called *fixed parameter tractable (FPT)* if it can be solved in time $f(p) \cdot n^c$, where f is a computable function only depending on p and c is a constant.

Parameterized Complexity

Fixed Parameter Tractable (FPT)

A problem with input size n and parameter p is called *fixed parameter tractable (FPT)* if it can be solved in time $f(p) \cdot n^c$, where f is a computable function only depending on p and c is a constant.

Parameters:

- treewidth, rankwidth, and vertex cover: computing them is NP-hard
- clique-width: computing it is still an open problem
- modular width, neighborhood diversity: computable in polynomial time

MBV and parameterized complexity

- MBV is FPT with respect to treewidth [Baste et al., 2022]
- Hamiltonian Path problem (MBV special case) is W[1]-hard when parametetized by cliquewidth [Fomin et al., 2010]
- Hamiltonian Path problem is FPT with respect to modular-width [Gajarský et al., 2013]

MBV and parameterized complexity

- MBV is FPT with respect to treewidth [Baste et al., 2022]
- Hamiltonian Path problem (MBV special case) is W[1]-hard when parametetized by cliquewidth [Fomin et al., 2010]
- Hamiltonian Path problem is FPT with respect to modular-width [Gajarský et al., 2013]

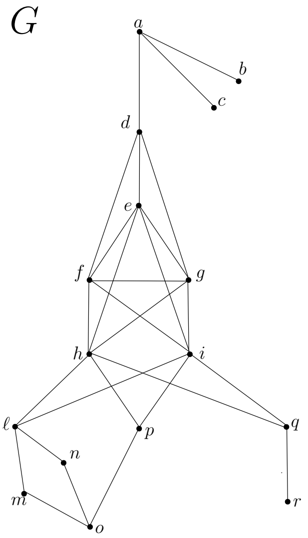
We study the MBV problem with respect to neighborhood diversity and prove

Theorem 1

The MINIMUM BRANCH VERTICES *spanning tree problem is FPT when parameterized by neighborhood diversity.*

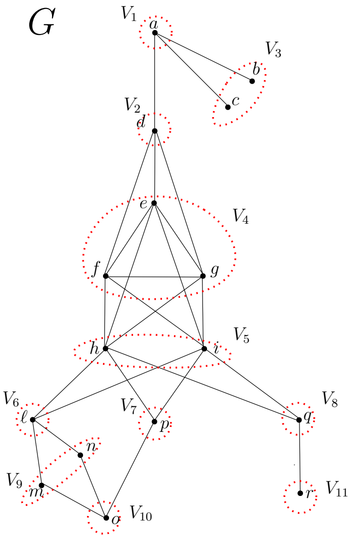
Neighborhood Diversity [Lampis 2010]

- Graph $G = (V, E)$
- $u, v \in V$ have the same *type* iff $N(v) - \{u\} = N(u) - \{v\}$



Neighborhood Diversity [Lampis 2010]

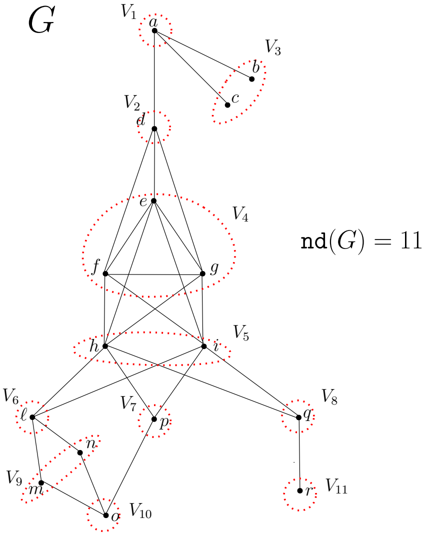
- Graph $G = (V, E)$
- $u, v \in V$ have the same *type* iff $N(v) - \{u\} = N(u) - \{v\}$
- A type partition of G is a partition $V_1, V_2, \ldots, V_t$, of the vertex set V, such that all the vertices in the type set V_i have the same type, for $i \in \{1, \ldots, t\}$
  - Note that by definition, each type set V_i induces either a *clique* or an *independent set* in G



Neighborhood Diversity [Lampis 2010]

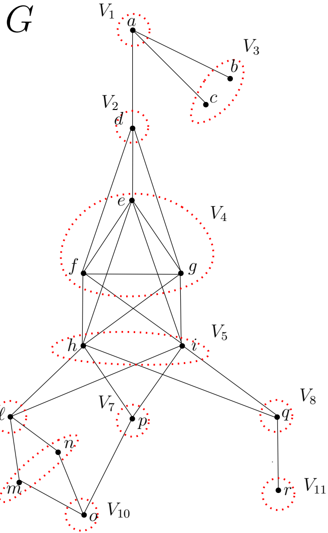
- Graph $G = (V, E)$
- $u, v \in V$ have the same *type* iff $N(v) - \{u\} = N(u) - \{v\}$
- A type partition of G is a partition $V_1, V_2, \ldots, V_t$, of the vertex set V, such that all the vertices in the type set V_i have the same type, for $i \in \{1, \ldots, t\}$
  - Note that by definition, each type set V_i induces either a *clique* or an *independent set* in G

The *neighborhood diversity* of G, $\mathtt{nd}(G)$, is the minimum number t of sets in a type partition $V_1, V_2, \ldots, V_t$ of G

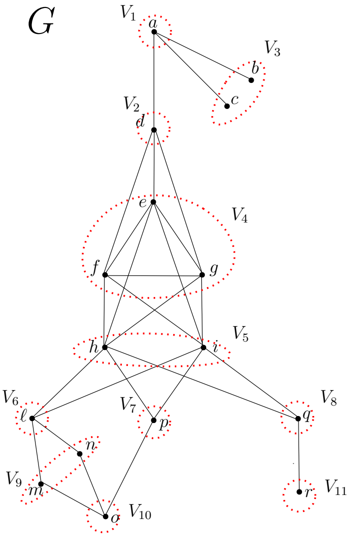


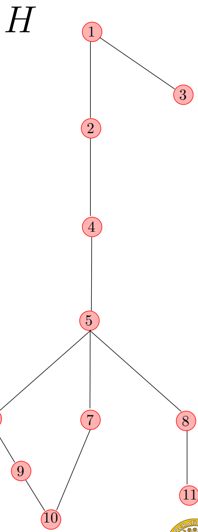
Neighborhood Diversity [Lampis 2010]

- Graph $G = (V, E)$
- The type partition $V_1, V_2, \ldots, V_{\texttt{nd}}$ of G



- Graph $G = (V, E)$
- The type partition $V_1, V_2, \ldots, V_{\texttt{nd}}$ of G
- The type graph H of G is defined as
    - $V(H) = \{1, \ldots, \texttt{nd}\}$
    - $E(H) = \{\{x, y\} \mid x \neq y$ and for each $u \in V_x$, $v \in V_y$ it holds $\{u, v\} \in E\}$





The minimum number of branch vertices

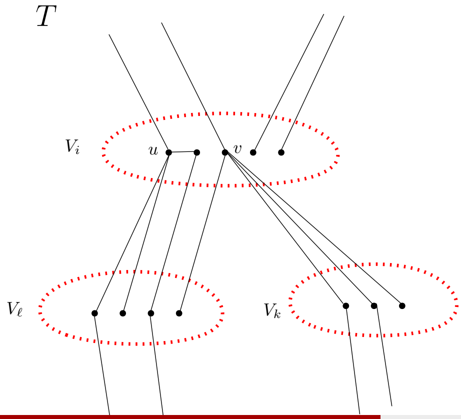
Lemma 2

Let $G = (V, E)$ *be a connected graph with type partition* $\mathcal{V} = \{V_1, V_2, \ldots, V_{\mathtt{nd}}\}$. *Any spanning tree of* G *with* $b(G)$ *branch vertices has at most one branch vertex belonging to any set of the type partition* $\mathcal{V}$. *Hence,* $b(G) \leq \mathtt{nd}$.

The minimum number of branch vertices

Lemma 2

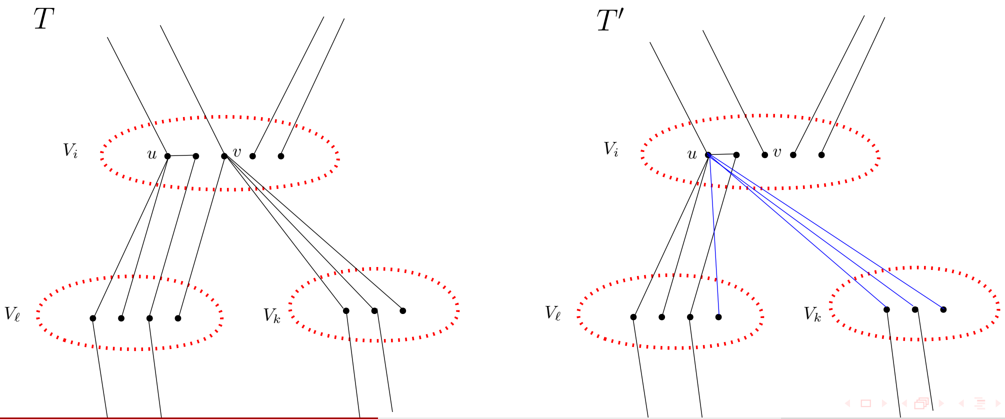
Let $G = (V, E)$ be a connected graph with type partition $\mathcal{V} = \{V_1, V_2, \ldots, V_{\mathtt{nd}}\}$. Any spanning tree of G with $b(G)$ branch vertices has at most one branch vertex belonging to any set of the type partition $\mathcal{V}$. Hence, $b(G) \leq \mathtt{nd}$.



The minimum number of branch vertices

Lemma 2

Let $G = (V, E)$ be a connected graph with type partition $\mathcal{V} = \{V_1, V_2, \ldots, V_{\mathtt{nd}}\}$. Any spanning tree of G with $b(G)$ branch vertices has at most one branch vertex belonging to any set of the type partition $\mathcal{V}$. Hence, $b(G) \leq \mathtt{nd}$.



The Algorithm

- Connected graph $G = (V, E)$
- The type partition $V_1, V_2, \dots, V_{\texttt{nd}}$ of G
- The type graph H of G

The algorithm

For each fixed set $B_H \subseteq \{1, \dots, \texttt{nd}\}$, ordered by size,

1. solve an Integer Linear Program for B_H (exploiting the properties of the type partition of G)
2. if a solution of ILP exists for the current set B_H, construct a spanning tree of G with $|B_H|$ branch vertices (where each branch vertex is from exactly one type set whose index is in B_H).

The Algorithm

- Connected graph $G = (V, E)$
- The type partition $V_1, V_2, \ldots, V_{\mathtt{nd}}$ of G
- The type graph H of G

The algorithm

For each fixed set $B_H \subseteq \{1, \ldots, \mathtt{nd}\}$, ordered by size,

1. solve an Integer Linear Program for B_H (exploiting the properties of the type partition of G)
2. if a solution of ILP exists for the current set B_H, construct a spanning tree of G with $|B_H|$ branch vertices (where each branch vertex is from exactly one type set whose index is in B_H).

Since modular-width generalizes neighborhood diversity, the FPT algorithm for Hamiltonian Path parameterized by modular width in [Gajarský et al., 2013] can be used for the case $B_H = \emptyset$.

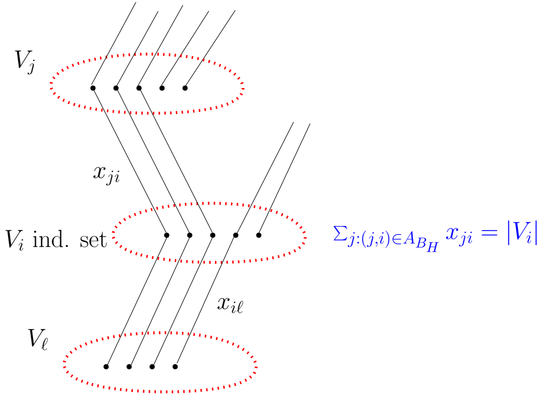
In the following we assume $|B_H| \geq 1$.

Integer Linear Program for $B_H \subseteq \{1, \ldots, \mathtt{nd}\}$

- Select any arbitrary $r \in B_H$
- $H_{B_H} = (\{1, \ldots, \mathtt{nd}\} \cup \{s\}, A_{B_H})$, with $A_{B_H} = \{(s,r)\} \cup \{(i,j),(j,i) \mid \{i,j\} \in E(H)\}$
- For each $(j,i) \in A_{B_H}$, ILP uses a variable x_{ji}
  $x_{ji} =$ the number of vertices in the type set V_i whose parent in the spanning tree, is a vertex in the type set V_j

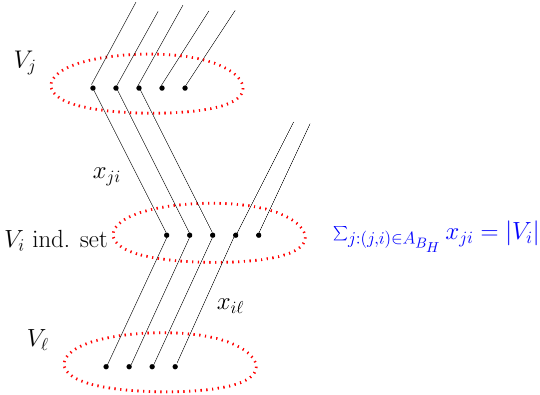
Integer Linear Program for $B_H \subseteq \{1, \ldots, \mathtt{nd}\}$

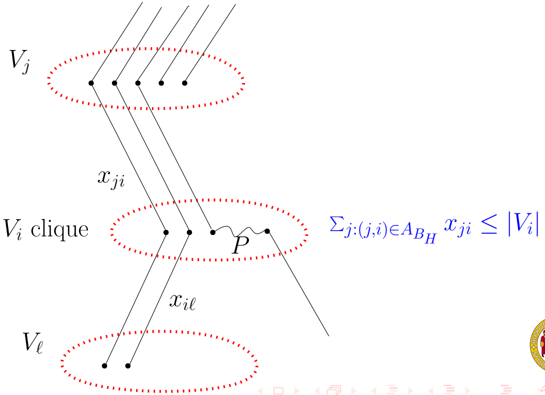
- Select any arbitrary $r \in B_H$
- $H_{B_H} = (\{1, \ldots, \mathtt{nd}\} \cup \{s\}, A_{B_H})$, with $A_{B_H} = \{(s,r)\} \cup \{(i,j),(j,i) \mid \{i,j\} \in E(H)\}$
- For each $(j,i) \in A_{B_H}$, ILP uses a variable x_{ji}
  $x_{ji} =$ the number of vertices in the type set V_i whose parent in the spanning tree, is a vertex in the type set V_j



Integer Linear Program for $B_H \subseteq \{1, \ldots, \mathtt{nd}\}$

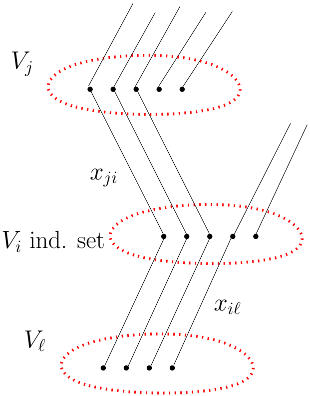
- Select any arbitrary $r \in B_H$
- $H_{B_H} = (\{1, \ldots, \mathtt{nd}\} \cup \{s\}, A_{B_H})$, with $A_{B_H} = \{(s, r)\} \cup \{(i, j), (j, i) \mid \{i, j\} \in E(H)\}$
- For each $(j, i) \in A_{B_H}$, ILP uses a variable x_{ji}
  $x_{ji} =$ the number of vertices in the type set V_i whose parent in the spanning tree, is a vertex in the type set V_j





Integer Linear Program for $B_H \subseteq \{1, \ldots, \mathtt{nd}\}$

- Select any arbitrary $r \in B_H$
- $H_{B_H} = (\{1, \ldots, \mathtt{nd}\} \cup \{s\}, A_{B_H})$, with $A_{B_H} = \{(s, r)\} \cup \{(i, j), (j, i) \mid \{i, j\} \in E(H)\}$
- For each $(j, i) \in A_{B_H}$, ILP uses a variable x_{ji}
  $x_{ji} =$ the number of vertices in the type set V_i whose parent in the spanning tree, is a vertex in the type set V_j



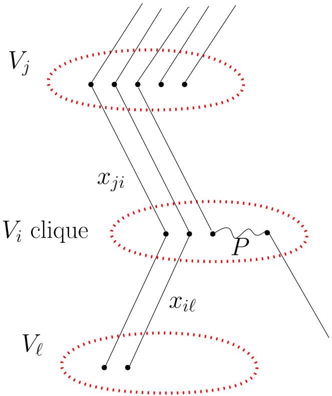
$i \notin B_H$

$\Sigma_{\ell:(i,\ell)\in A_{B_H}} x_{i\ell} - \Sigma_{j:(j,i)\in A_{B_H}} x_{ji} \leq 0$

Integer Linear Program for $B_H \subseteq \{1, \ldots, \mathtt{nd}\}$

- Select any arbitrary $r \in B_H$
- $H_{B_H} = (\{1, \ldots, \mathtt{nd}\} \cup \{s\}, A_{B_H})$, with $A_{B_H} = \{(s, r)\} \cup \{(i, j), (j, i) \mid \{i, j\} \in E(H)\}$
- For each $(j, i) \in A_{B_H}$, ILP uses a variable x_{ji}

  $x_{ji} =$ the number of vertices in the type set V_i whose parent in the spanning tree, is a vertex in the type set V_j



$i \notin B_H$

$\Sigma_{\ell:(i,\ell) \in A_{B_H}} x_{i\ell} - \Sigma_{j:(j,i) \in A_{B_H}} x_{ji} \leq 0$

Integer Linear Program for $B_H \subseteq \{1, \ldots, \texttt{nd}\}$

$$x_{sr} = 1$$

$$\sum_{j:(j,i)\in A_{B_H}} x_{ji} \leq |V_i| \qquad \forall i \in \{1, \ldots, \texttt{nd}\} \text{ s.t. } V_i \text{ is a clique}$$

$$\sum_{j:(j,i)\in A_{B_H}} x_{ji} = |V_i| \qquad \forall i \in \{1, \ldots, \texttt{nd}\} \text{ s.t. } V_i \text{ is an ind. set}$$

$$\sum_{\ell:(i,\ell)\in A_{B_H}} x_{i\ell} - \sum_{j:(j,i)\in A_{B_H}} x_{ji} \leq 0 \qquad \forall i \in \{1, \ldots, \texttt{nd}\} - B_H$$

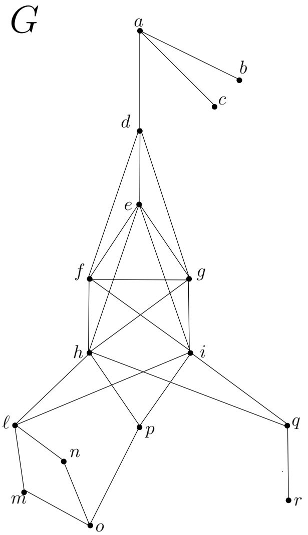
$$y_{sr} = \texttt{nd}$$

$$\sum_{j:(j,i)\in A_{B_H}} y_{ji} - \sum_{\ell:(i,\ell)\in A_{B_H}} y_{i\ell} = 1 \qquad \forall i \in \{1, \ldots, \texttt{nd}\}$$

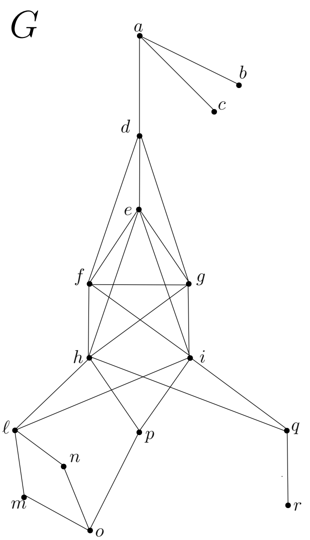
$$y_{ij} \leq \texttt{nd}\; x_{ij} \qquad \forall (i,j) \in A_{B_H}$$

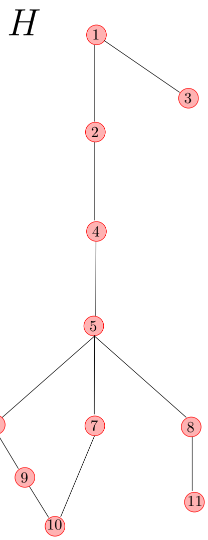
$$y_{ij}, x_{ij} \in \mathbb{N} \qquad \forall (i,j) \in A_{B_H}$$

Integer Linear Program

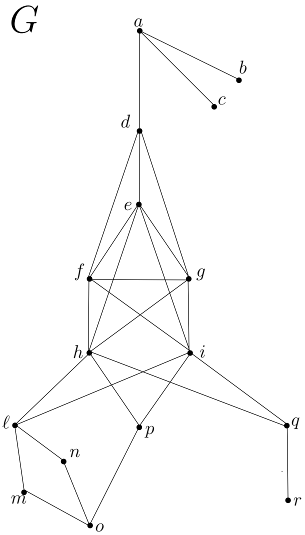


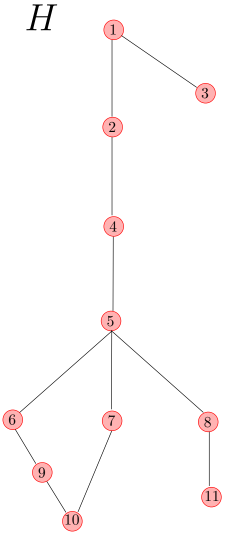
Integer Linear Program





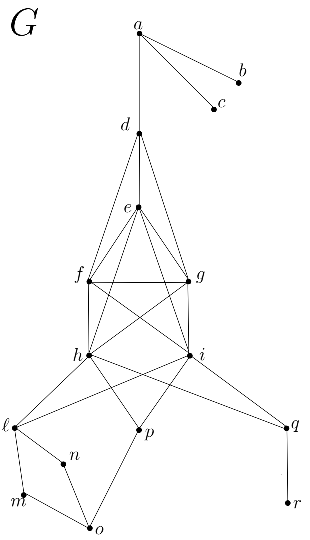
Integer Linear Program

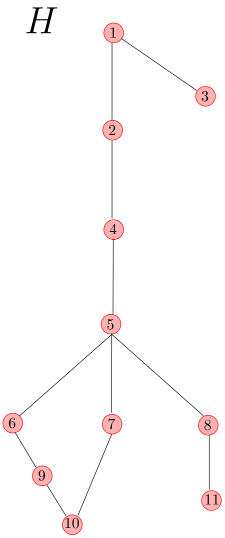




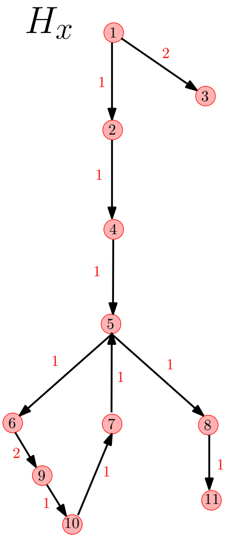
The ILP has no solution for each $B_H \subseteq \{1, 2, \ldots, 11\}$ with $|B_H| = 1$

Integer Linear Program





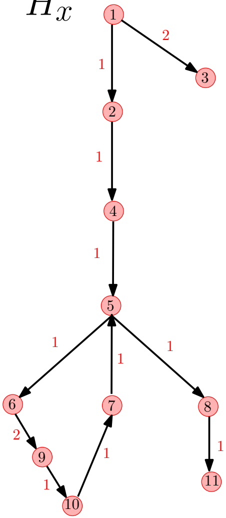
The ILP has a solution (x, y) for $B_H = \{1, 6\}$



The spanning tree construction

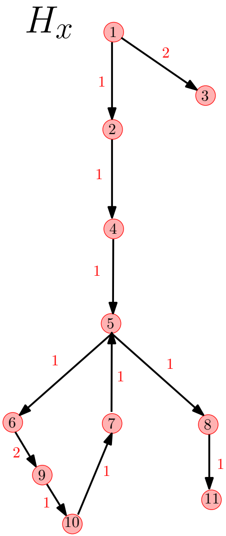
$B_H = \{1, 6\}$

H_x

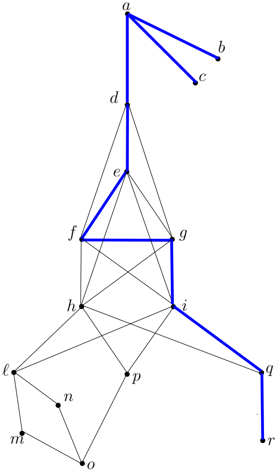


The spanning tree construction

$B_H = \{1, 6\}$

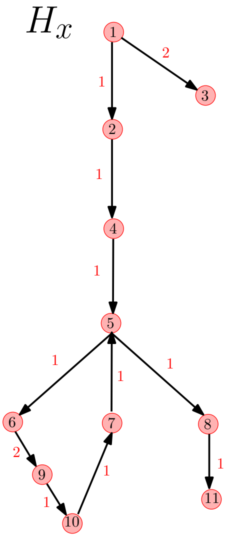


Choose a vertex in $V_1 = \{a\}$ as root of T and start to explore G, according to the values of x, until it is possible

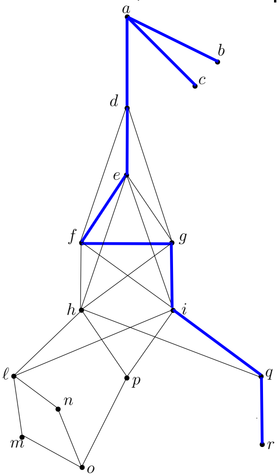


The spanning tree construction

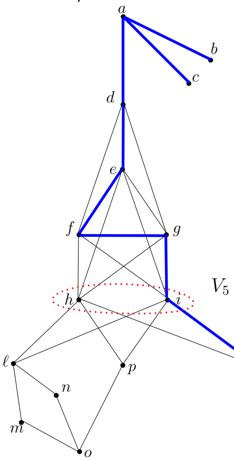
$B_H = \{1, 6\}$



Choose a vertex in $V_1 = \{a\}$ as root of T and start to explore G, according to the values of x, until it is possible



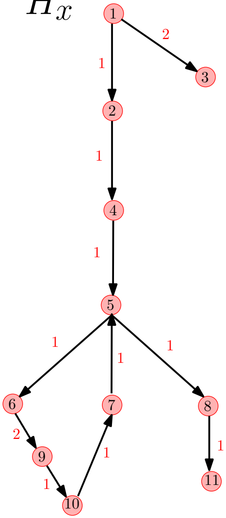
There exists always a type set having an explored vertex and an unexplored vertex; this holds for V_5



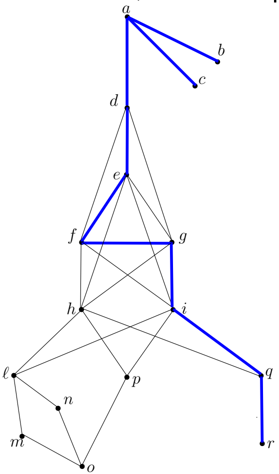
The spanning tree construction

$B_H = \{1, 6\}$

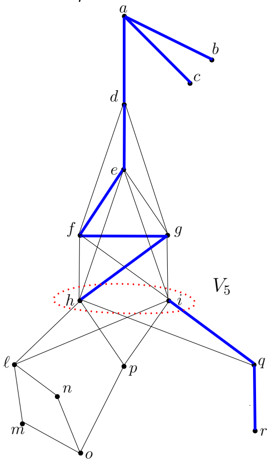
H_x



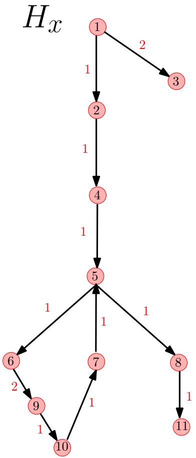
Choose a vertex in $V_1 = \{a\}$ as root of T and start to explore G, according to the values of x, until it is possible



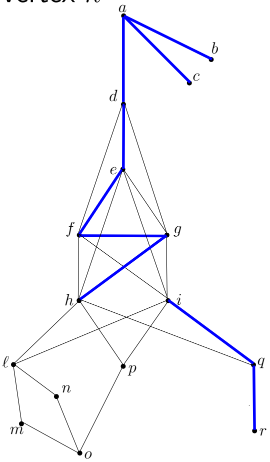
There exists always a type set having an explored vertex and an unexplored vertex; this holds for V_5



$B_H = \{1, 6\}$



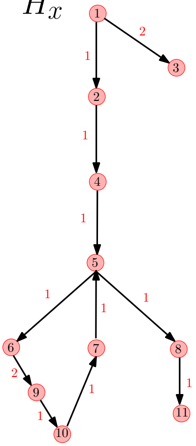
Start a new exploration from vertex h



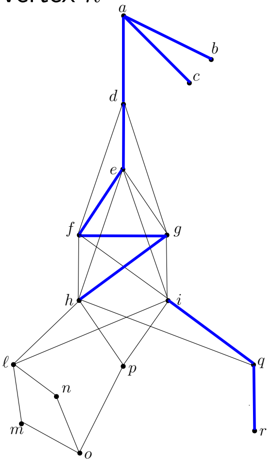
The spanning tree construction

$B_H = \{1, 6\}$

H_x



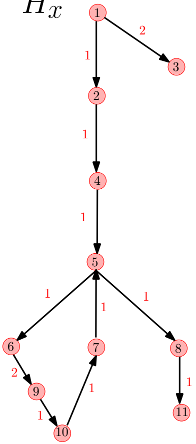
Start a new exploration from vertex h



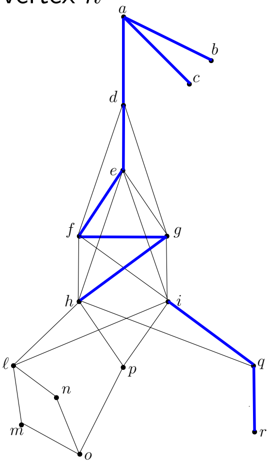
There is a cycle in H_x containing 5. The subtree rooted at i is reconnected to the main tree. All the vertices have been explored

$B_H = \{1, 6\}$

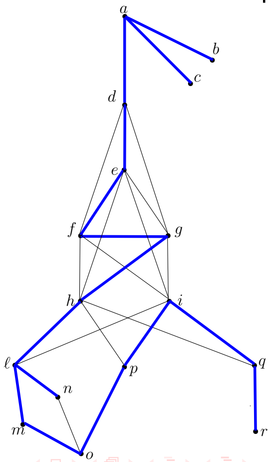
H_x



Start a new exploration from vertex h



There is a cycle in H_x containing 5. The subtree rooted at i is reconnected to the main tree. All the vertices have been explored



Optimality

Lemma 3

If there exists a spanning tree in G with $k \geq 1$ branch vertices then there exists a set $B_H \subseteq \{1, \ldots, \mathtt{nd}\}$ with $|B_H| = k$, and a solution (x, y) of the corresponding ILP.

The algorithm complexity

The algorithm

For each fixed set $B_H \subseteq \{1, \ldots, \mathtt{nd}\}$, ordered by size,

1. solve an Integer Linear Program for B_H
2. if a solution of ILP exists for the current set B_H, construct a spanning tree of G with $|B_H|$ branch vertices.

The algorithm complexity

The algorithm

For each fixed set $B_H \subseteq \{1, \ldots, \mathtt{nd}\}$, ordered by size,

1. solve an Integer Linear Program for B_H
2. if a solution of ILP exists for the current set B_H, construct a spanning tree of G with $|B_H|$ branch vertices.

For each fixed $B_H \subseteq \{1, \ldots, \mathtt{nd}\}$,

1. using [Jansen et al., 2023] that gives an efficient algorithm to find a feasible solution of an ILP with small number of constraints, we have that our ILP can be solved in time

$$O(\mathtt{nd}^2)^{(1+o(1))(\mathtt{nd}^2+3\mathtt{nd}+2)} + O(\mathtt{nd}^4)$$

2. the time to construct the spanning tree is $O(n^2)$

The algorithm complexity

The algorithm

For each fixed set $B_H \subseteq \{1, \ldots, \mathtt{nd}\}$, ordered by size,

1. solve an Integer Linear Program for B_H
2. if a solution of ILP exists for the current set B_H, construct a spanning tree of G with $|B_H|$ branch vertices.

For each fixed $B_H \subseteq \{1, \ldots, \mathtt{nd}\}$,

1. using [Jansen et al., 2023] that gives an efficient algorithm to find a feasible solution of an ILP with small number of constraints, we have that our ILP can be solved in time

$$O(\mathtt{nd}^2)^{(1+o(1))(\mathtt{nd}^2+3\mathtt{nd}+2)} + O(\mathtt{nd}^4)$$

2. the time to construct the spanning tree is $O(n^2)$

Overall the time of our algorithm is

$$2^{\mathtt{nd}}(O(\mathtt{nd}^2)^{(1+o(1))(\mathtt{nd}^2+3\mathtt{nd}+2)} + O(\mathtt{nd}^4)) + O(n^2)$$

which is only additive in the input size.

thank you