JASS 2008: Trees

Trees with many leaves

Nikolay Gravin

Saint-Petersburg State University
Faculty of Mathematics and Mechanics

March 2008

What this talk is about:

- Linial's conjecture (posed in 1988)
- Storer's algorithm of finding a spanning tree with many leaves for cubic graphs (1981)
- The maximum leaf spanning tree problem is NP-complete (P. Lemke 1988)

Problem's definition

We are given a graph and we want to find a spanning tree in this graph.

Problem definition

Moreover we want to pick out the most “nonsingular” tree.

A good criteria of such “nonsingularity” is the number of leaves in a tree.

Problem definition

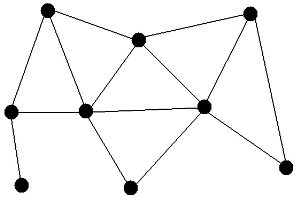
So we get the following problem:

Given a graph we need to find a spanning tree with maximal number of leaves.

Spanning tree definition

Definition (Spanning tree)

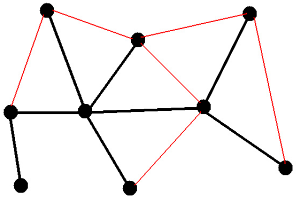
A tree which is a subgraph of some graph G and contains all its vertices called a **spanning tree** of G.



Spanning tree definition

Definition (Spanning tree)

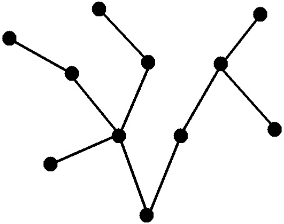
A tree which is a subgraph of some graph G and contains all its vertices called a **spanning tree** of G.



Leaf definition

Definition (Pendant vertex)

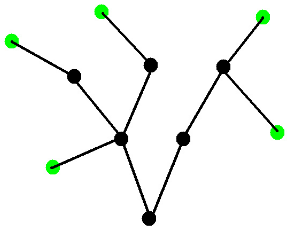
A vertex in a tree with degree one called a **pendant vertex** or a **leaf**.



Leaf definition

Definition (Pendant vertex)

A vertex in a tree with degree one called a **pendant vertex** or a **leaf**.



Notations

$V(G)$ — Set of all vertices of the graph G

$\delta(G) :=$ Minimum degree of the graph G

$L(T) :=$ Number of leaves in the tree T

$L(G) :=$ Maximal number of leaves over all spanning trees of the graph G.

Linial's Conjecture

Conjecture

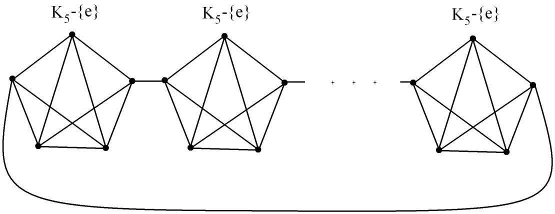
Let G be a graph on N vertices with $\delta(G) = k$. Then

$$L(G) \geq \frac{k-2}{k+1}N + c_k$$

where c_k depends only on k.
$\delta(G) :=$ Minimum degree of the graph G.
$L(G) :=$ Maximal number of leaves over all spanning trees of the graph G.

Tightness

The lower bound of $L(G)$ ($\frac{k-2}{k+1}N + c_k$), where k is minimum degree of G, is tight.



A series of examples for $k = 4$.
The same "necklace" example suits for another values of k.

Known results about Linial's conjecture

- $\delta(G) = 3$ Linial's conjecture holds (Storer 1981)
- $\delta(G) = 4$ Linial's conjecture holds (Jerrold, R. Griggs, Mingshen Wu 1992)
- $\delta(G) = 5$ Linial's conjecture holds (Jerrold, R. Griggs, Mingshen Wu 1992)
- $\delta(G) \geq 6$ open problem
- $\delta(G) \to \infty$ Linial's conjecture fails. (N. Alon 1990) There are series of graphs G_k with $\delta(G_k) = k$ such

$$L(G_k) \leq \left(1 - \frac{\log(k)}{k+1}\right) |V(G_k)| \left(1 + \frac{O(1)}{k+1}\right)$$

Known results about Linial's conjecture

- $\delta(G) = 3$ Linial's conjecture holds (Storer 1981)
- $\delta(G) = 4$ Linial's conjecture holds (Jerrold, R. Griggs, Mingshen Wu 1992)
- $\delta(G) = 5$ Linial's conjecture holds (Jerrold, R. Griggs, Mingshen Wu 1992)
- $\delta(G) \geq 6$ open problem
- $\delta(G) \to \infty$ Linial's conjecture fails. (N. Alon 1990) There are series of graphs G_k with $\delta(G_k) = k$ such

$$L(G_k) \leq \left(1 - \frac{\log(k)}{k+1}\right) |V(G_k)| \left(1 + \frac{O(1)}{k+1}\right)$$

Known results about Linial's conjecture

- $\delta(G) = 3$ Linial's conjecture holds (Storer 1981)
- $\delta(G) = 4$ Linial's conjecture holds (Jerrold, R. Griggs, Mingshen Wu 1992)
- $\delta(G) = 5$ Linial's conjecture holds (Jerrold, R. Griggs, Mingshen Wu 1992)
- $\delta(G) \geq 6$ open problem
- $\delta(G) \to \infty$ Linial's conjecture fails. (N. Alon 1990) There are series of graphs G_k with $\delta(G_k) = k$ such

$$L(G_k) \leq \left(1 - \frac{\log(k)}{k+1}\right) |V(G_k)| \left(1 + \frac{O(1)}{k+1}\right)$$

Known results about Linial's conjecture

- $\delta(G) = 3$ Linial's conjecture holds (Storer 1981)
- $\delta(G) = 4$ Linial's conjecture holds (Jerrold, R. Griggs, Mingshen Wu 1992)
- $\delta(G) = 5$ Linial's conjecture holds (Jerrold, R. Griggs, Mingshen Wu 1992)
- $\delta(G) \geq 6$ open problem
- $\delta(G) \to \infty$ Linial's conjecture fails. (N. Alon 1990) There are series of graphs G_k with $\delta(G_k) = k$ such

$$L(G_k) \leq \left(1 - \frac{\log(k)}{k+1}\right) |V(G_k)| \left(1 + \frac{O(1)}{k+1}\right)$$

Known results about Linial's conjecture

- $\delta(G) = 3$ Linial's conjecture holds (Storer 1981)
- $\delta(G) = 4$ Linial's conjecture holds (Jerrold, R. Griggs, Mingshen Wu 1992)
- $\delta(G) = 5$ Linial's conjecture holds (Jerrold, R. Griggs, Mingshen Wu 1992)
- $\delta(G) \geq 6$ open problem
- $\delta(G) \to \infty$ Linial's conjecture fails. (N. Alon 1990)
  There are series of graphs G_k with $\delta(G_k) = k$ such

$$L(G_k) \leq \left(1 - \frac{\log(k)}{k+1}\right) |V(G_k)| \left(1 + \frac{O(1)}{k+1}\right)$$

Known results about Linial's conjecture

- $\delta(G) = 3$ Linial's conjecture holds (Storer 1981)
- $\delta(G) = 4$ Linial's conjecture holds (Jerrold, R. Griggs, Mingshen Wu 1992)
- $\delta(G) = 5$ Linial's conjecture holds (Jerrold, R. Griggs, Mingshen Wu 1992)
- $\delta(G) \geq 6$ open problem
- $\delta(G) \to \infty$ Linial's conjecture fails. (N. Alon 1990) There are series of graphs G_k with $\delta(G_k) = k$ such

$$L(G_k) \leq \left(1 - \frac{\log(k)}{k+1}\right) |V(G_k)| \left(1 + \frac{O(1)}{k+1}\right)$$

Assumptions

- Cubic is a graph where all vertices have degree equal to three.
- Now and then G be a cubic graph on N vertices.
- We want to pick out a spanning tree with at least $\lfloor \frac{1}{4}N \rfloor + 2$ leaves.
- We will construct such tree consequently and at each step of algorithm we have a partial tree of G.

Assumptions

- Cubic is a graph where all vertices have degree equal to three.
- Now and then G be a cubic graph on N vertices.
- We want to pick out a spanning tree with at least $\lfloor \frac{1}{4}N \rfloor + 2$ leaves.
- We will construct such tree consequently and at each step of algorithm we have a partial tree of G.

Assumptions

- Cubic is a graph where all vertices have degree equal to three.
- Now and then G be a cubic graph on N vertices.
- We want to pick out a spanning tree with at least $\lfloor \frac{1}{4}N \rfloor + 2$ leaves.
- We will construct such tree consequently and at each step of algorithm we have a partial tree of G.

Assumptions

- Cubic is a graph where all vertices have degree equal to three.
- Now and then G be a cubic graph on N vertices.
- We want to pick out a spanning tree with at least $\lfloor \frac{1}{4}N \rfloor + 2$ leaves.
- We will construct such tree consequently and at each step of algorithm we have a partial tree of G.

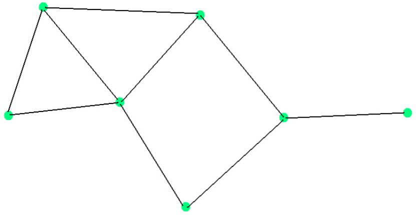
Assumptions

- Cubic is a graph where all vertices have degree equal to three.
- Now and then G be a cubic graph on N vertices.
- We want to pick out a spanning tree with at least $\lfloor \frac{1}{4}N \rfloor + 2$ leaves.
- We will construct such tree consequently and at each step of algorithm we have a partial tree of G.

Definitions of a dead leaf

Definition Dead vertex

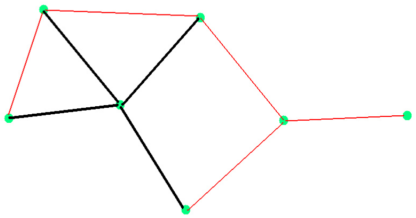
A leaf v of a partial tree T of G called **dead** iff v has no adjacent to it vertices outside T.



Definitions of a dead leaf

Definition Dead vertex

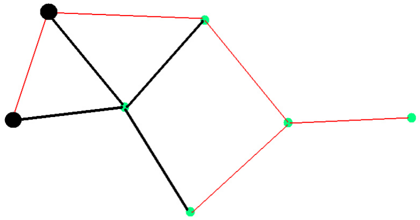
A leaf v of a partial tree T of G called **dead** iff v has no adjacent to it vertices outside T.



Definitions of a dead leaf

Definition `Dead vertex`

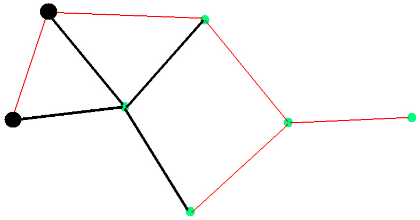
A leaf v of a partial tree T of G called **dead** iff v has no adjacent to it vertices outside T.



Definitions of a dead leaf

Definition Dead vertex

A leaf v of a partial tree T of G called **dead** iff v has no adjacent to it vertices outside T.



$D(T) :=$ number of dead leaves in the partial tree T.

Storer's Algorithm

Plan of algorithm

We would construct a spanning tree consequently.

- Consider cost function involving the number of leaves, dead leaves and vertices of T

$$f(T) := 3L(T) + D(T) - |V(T)|.$$

- At each step of algorithm we shall always seek to enlarge a constructed partial tree T of G while not decreasing $f(T)$.

$L(T)$:= Number of leaves in T.
$D(T)$:= Number of dead leaves in T

Storer's Algorithm

Plan of algorithm

We would construct a spanning tree consequently.

- Consider cost function involving the number of leaves, dead leaves and vertices of T

$$f(T) := 3L(T) + D(T) - |V(T)|.$$

- At each step of algorithm we shall always seek to enlarge a constructed partial tree T of G while not decreasing $f(T)$.

$L(T)$:= Number of leaves in T.
$D(T)$:= Number of dead leaves in T

Storer's Algorithm

Plan of algorithm

We would construct a spanning tree consequently.

- Consider cost function involving the number of leaves, dead leaves and vertices of T

$$f(T) := 3L(T) + D(T) - |V(T)|.$$

- At each step of algorithm we shall always seek to enlarge a constructed partial tree T of G while not decreasing $f(T)$.

$L(T)$:= Number of leaves in T.
$D(T)$:= Number of dead leaves in T

Storer's Algorithm

Plan of algorithm

We would construct a spanning tree consequently.

- Starting tree T would be any vertex with its neighborhood, so $f(T) \geq 3 * 3 + 0 - 4 \geq 5$
- At the end of algorithm T would be some spanning tree of G. So $D(T) = L(T)$ as all leaves would be dead.
- $3L(T) + L(T) - |V(T)| \geq 5$, so $4L(T) \geq N + 5$

$L(T) :=$ Number of leaves in T.
$D(T) :=$ Number of dead leaves in T

Storer's Algorithm

Plan of algorithm

We would construct a spanning tree consequently.

- Starting tree T would be any vertex with its neighborhood, so $f(T) \geq 3 * 3 + 0 - 4 \geq 5$
- At the end of algorithm T would be some spanning tree of G. So $D(T) = L(T)$ as all leaves would be dead.
- $3L(T) + L(T) - |V(T)| \geq 5$, so $4L(T) \geq N + 5$

$L(T)$:= Number of leaves in T.
$D(T)$:= Number of dead leaves in T

Storer's Algorithm

Plan of algorithm

We would construct a spanning tree consequently.

- Starting tree T would be any vertex with its neighborhood, so $f(T) \geq 3 * 3 + 0 - 4 \geq 5$
- At the end of algorithm T would be some spanning tree of G. So $D(T) = L(T)$ as all leaves would be dead.
- $3L(T) + L(T) - |V(T)| \geq 5$, so $4L(T) \geq N + 5$

$L(T) :=$ Number of leaves in T.
$D(T) :=$ Number of dead leaves in T

Storer's Algorithm

Plan of algorithm

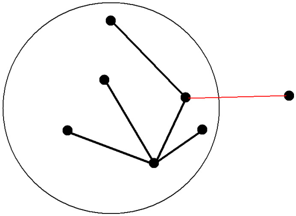
We would construct a spanning tree consequently.

- Starting tree T would be any vertex with its neighborhood, so $f(T) \geq 3 * 3 + 0 - 4 \geq 5$
- At the end of algorithm T would be some spanning tree of G. So $D(T) = L(T)$ as all leaves would be dead.
- $3L(T) + L(T) - |V(T)| \geq 5$, so $4L(T) \geq N + 5$

$L(T) :=$ Number of leaves in T.
$D(T) :=$ Number of dead leaves in T

Why we could enlarge T and do not decrease $f(T)$

Non leaf vertex of T is adjacent to vertex outside T.

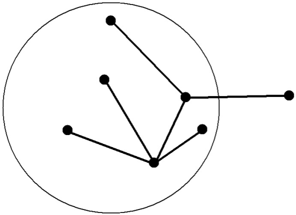


$L(T)$	$D(T)$	$\lvert V(T)\rvert$
$+1$	$+\geq 0$	$+1$

$$f(T) := 3L(T) + D(T) - |V(T)|$$

Why we could enlarge T and do not decrease $f(T)$

Non leaf vertex of T is adjacent to vertex outside T.

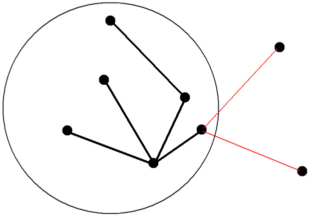


$L(T)$	$D(T)$	$\|V(T)\|$
$+1$	$+ \geq 0$	$+1$

$$f(T) := 3L(T) + D(T) - |V(T)|$$

Why we could enlarge T and do not decrease $f(T)$

Some leaf of T is adjacent to two vertices outside T.

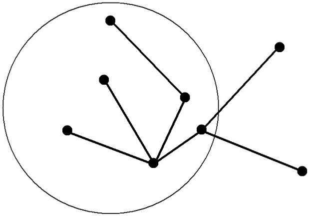


$L(T)$	$D(T)$	$\|V(T)\|$
$+1$	$+\geq 0$	$+2$

$$f(T) := 3L(T) + D(T) - |V(T)|$$

Why we could enlarge T and do not decrease $f(T)$

Some leaf of T is adjacent to two vertices outside T.

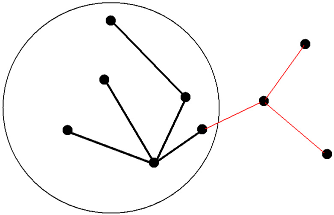


$L(T)$	$D(T)$	$\lvert V(T)\rvert$
$+1$	$+\geq 0$	$+2$

$$f(T) := 3L(T) + D(T) - |V(T)|$$

Why we could enlarge T and do not decrease $f(T)$

Some leaf of T is adjacent to an outside vertex with two neighbors outside T.

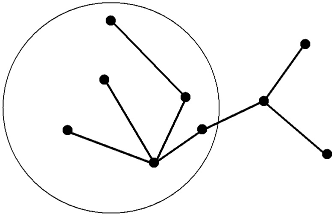


$L(T)$	$D(T)$	$\lvert V(T)\rvert$
$+1$	$+\geq 0$	$+3$

$$f(T) := 3L(T) + D(T) - |V(T)|$$

Why we could enlarge T and do not decrease $f(T)$

Some leaf of T is adjacent to an outside vertex with two neighbors outside T.

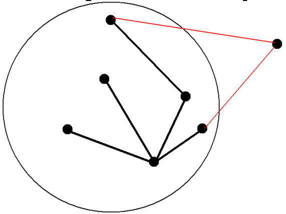


$L(T)$	$D(T)$	$\lvert V(T)\rvert$
$+1$	$+\geq 0$	$+3$

$$f(T) := 3L(T) + D(T) - |V(T)|$$

Why we could enlarge T and do not decrease $f(T)$

Outside T there is a vertex v adjacent to at least two leaves of T and this two leaves are adjacent to only v outside T.

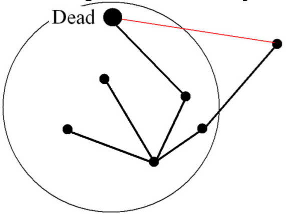


$L(T)$	$D(T)$	$\|V(T)\|$
$+0$	$+\geq 1$	$+1$

$$f(T) := 3L(T) + D(T) - |V(T)|$$

Why we could enlarge T and do not decrease $f(T)$

Outside T there is a vertex v adjacent to at least two leaves of T and this two leaves are adjacent to only v outside T.



$L(T)$	$D(T)$	$\lvert V(T)\rvert$
$+0$	$+\geq 1$	$+1$

$$f(T) := 3L(T) + D(T) - |V(T)|$$

NP-completeness

Theorem (Lemke) 1988

A maximum leaf spanning tree problem for cubic graphs is NP-complete.

INSTANCE: A cubic graph G and an integer number k

QUESTION: Does G posses a spanning tree with at least k leaves?

NP-completeness

Theorem (Lemke) 1988

INSTANCE: A cubic graph G

QUESTION: Does G posses a spanning tree with at least $\frac{|V(G)|}{2} + 1$ leaves?

EQUIVALENT QUESTION: Does G posses a spanning tree with no vertices of degree two?

Why equivalent question

$$
\begin{aligned}
a_1 &:= \text{number of vertices in spanning tree with degree 1.} \\
a_2 &:= \text{number of vertices in spanning tree with degree 2.} \\
a_3 &:= \text{number of vertices in spanning tree with degree 3.} \\
N &:= \text{number of vertices in } T.
\end{aligned}
$$

Why equivalent question

Then

$$a_1 + a_2 + a_3 = N$$

$$a_1 + 2a_2 + 3a_3 = 2 * (N - 1)$$

$$a_1 - a_3 = 2$$

$$a_1 + a_3 \leq N$$

We want

$$a_2 = 0 \Leftrightarrow a_1 \geq \frac{N}{2} + 1.$$

Why equivalent question

Then

$$a_1 + a_2 + a_3 = N$$

$$a_1 + 2a_2 + 3a_3 = 2 * (N - 1)$$

$$a_1 - a_3 = 2$$

$$a_1 + a_3 \leq N$$

We want

$$a_2 = 0 \Leftrightarrow a_1 \geq \frac{N}{2} + 1.$$

reduction

The proof is by reduction of known NP-complete problem EXACT COVER BY 3-SETS to ours.

INSTANCE: Positive integers n and m, subsets $S_1, S_2, ..., S_m$ of $\{1, 2, ..., n\}$, with $|S_i| = 3$ for all $i \in \{1, 2, ..., m\}$.

QUESTION: Is there a subset $Q \subseteq \{1, 2, ..., m\}$ such that $\bigcup\limits_{i \in Q} S_i = \{1, 2, ..., n\}$ and $\forall i_1, i_2 \in Q, i_1 \neq i_2 \Rightarrow S_{i_1} \cap S_{i_2} = \emptyset$?

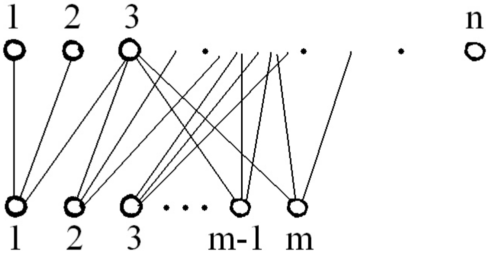
reduction

The proof is by reduction of known NP-complete problem EXACT COVER BY 3-SETS to ours.

INSTANCE: Positive integers n and m, subsets $S_1, S_2, ..., S_m$ of $\{1, 2, ..., n\}$, with $|S_i| = 3$ for all $i \in \{1, 2, ..., m\}$.

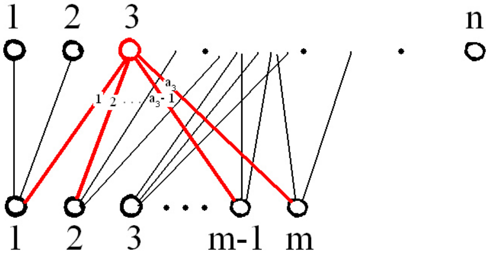
QUESTION: Is there a subset $Q \subseteq \{1, 2, ..., m\}$ such that $\bigcup_{i \in Q} S_i = \{1, 2, ..., n\}$ and $\forall i_1, i_2 \in Q, i_1 \neq i_2 \Rightarrow S_{i_1} \cap S_{i_2} = \emptyset$?

construction



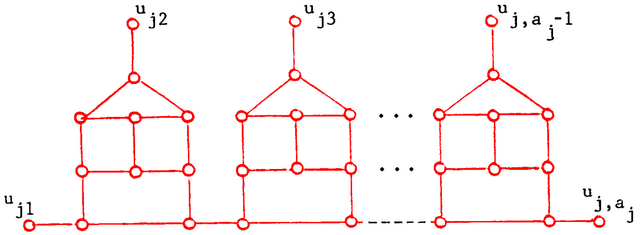
EXACT COVER BY 3-SETS instance representation as a bipartite graph

construction



In this representation we take every vertex from the set $\{1, 2, ..., n\}$ and all adjacent to it edges.

construction

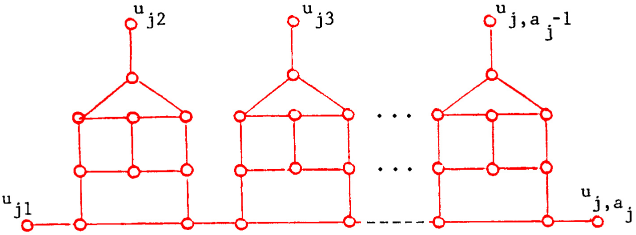


Draw the construction above for every vertex $j \in \{1, 2, ..., n\}$. In the figure u_{ji} corresponds to the i-th edge coming out from the vertex j.

Call graph constructed above U_j.

Draw the same construction U_0 with $2m$ vertices $u_{0,1}, u_{0,2}, ..., u_{0,2m}$ of degree one.

construction

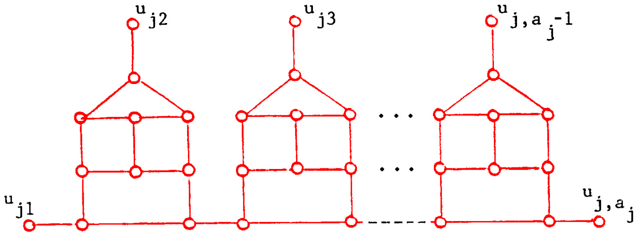


Draw the construction above for every vertex $j \in \{1, 2, ..., n\}$. In the figure u_{ji} corresponds to the i-th edge coming out from the vertex j.

Call graph constructed above U_j.

Draw the same construction U_0 with $2m$ vertices $u_{0,1}, u_{0,2}, ..., u_{0,2m}$ of degree one.

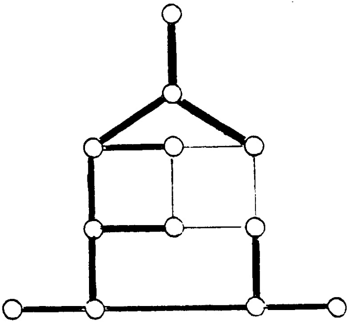
construction

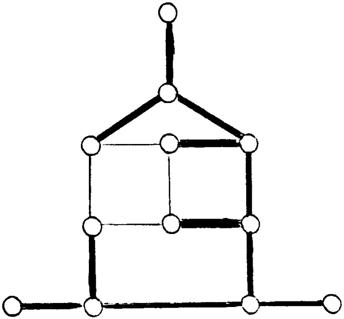


Draw the construction above for every vertex $j \in \{1, 2, ..., n\}$. In the figure u_{ji} corresponds to the i-th edge coming out from the vertex j.
Call graph constructed above U_j.
Draw the same construction U_0 with $2m$ vertices $u_{0,1}, u_{0,2}, ..., u_{0,2m}$ of degree one.

construction

The only two ways to assign the edges of a repeating sub-unit so that its nine interior vertices have odd degree in a spanning tree.





construction

The final cubic graph G will consists of $\bigcup_{j=0}^{n} U_j$ and some other vertices and edges.

For every set S_i consider three edges coming out from corresponding vertex of the bipartite graph. This three edges corresponds to some three vertices $u_{i_1 e_1}$, $u_{i_2 e_2}$, $u_{i_3 e_3}$.

Now let us describe the H_i graph corresponding to S_i.

construction

The final cubic graph G will consists of $\bigcup_{j=0}^{n} U_j$ and some other vertices and edges.

For every set S_i consider three edges coming out from corresponding vertex of the bipartite graph. This three edges corresponds to some three vertices $u_{i_1 e_1}$, $u_{i_2 e_2}$, $u_{i_3 e_3}$.

Now let us describe the H_i graph corresponding to S_i.

construction

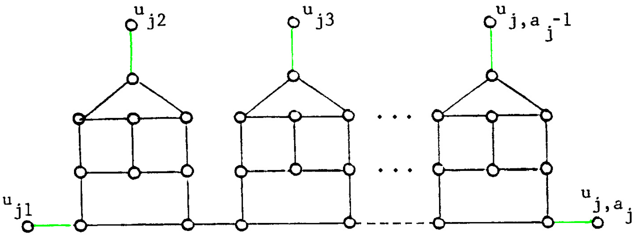
The final cubic graph G will consists of $\bigcup_{j=0}^{n} U_j$ and some other vertices and edges.

For every set S_i consider three edges coming out from corresponding vertex of the bipartite graph. This three edges corresponds to some three vertices $u_{i_1 e_1}$, $u_{i_2 e_2}$, $u_{i_3 e_3}$.

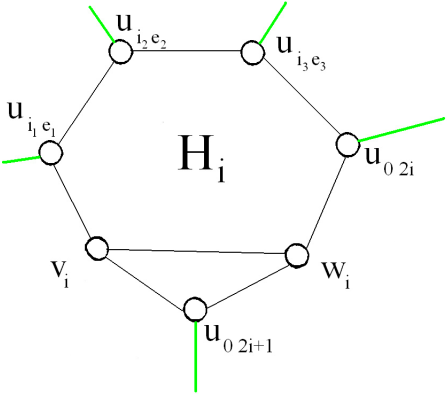
Now let us describe the H_i graph corresponding to S_i.

construction

U_j:

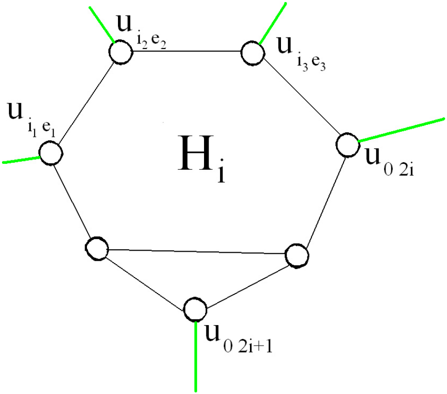


construction



So now the union of all U_j and H_i is a cubic graph.

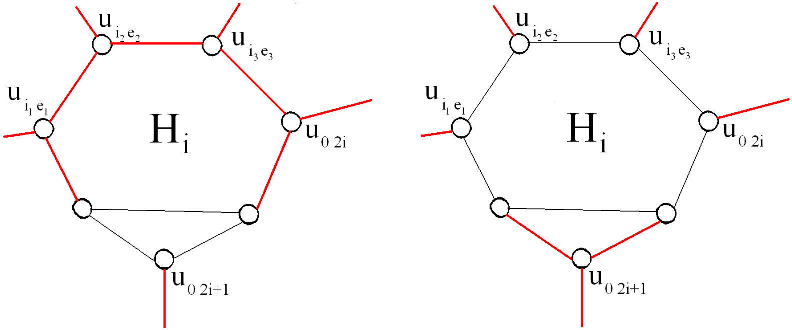
construction



We know that if a spanning tree has no vertices of degree two then all exterior edges (belong to some U_j) should belong to spanning tree.

construction

The only two ways to assign the edges of H_i so that its seven vertices have odd degree in a spanning tree.



construction

The first way corresponds to the situation when we take S_i in the covering.
And the second when we do not take S_i.

construction

The part of spanning tree corresponding to U_j is a connected subgraph.

In the first situation we connect $U_{i_1}, U_{i_2}, U_{i_3}$ with U_0.

In the second we do no connections between $U_{i_1}, U_{i_2}, U_{i_3}, U_0$

construction

The part of spanning tree corresponding to U_j is a connected subgraph.

In the first situation we connect $U_{i_1}, U_{i_2}, U_{i_3}$ with U_0.

In the second we do no connections between $U_{i_1}, U_{i_2}, U_{i_3}, U_0$

THANK YOU!