Bipartizing with a Matching*

Carlos V. G. C. Lima[a] Dieter Rautenbach[b]
Uéverton S. Souza[c] Jayme L. Szwarcfiter[d]

[a] Departamento de Ciência da Computação, UFMG, Brazil.

[b] Institute of Optimization and Operations Research, Ulm University, Germany.

[c] Instituto de Computação, UFF, Brazil.

[d] Programa de Engenharia de Sistemas e Computação, COPPE, UFRJ, Brazil.

ForWorC – 2019

Graph Modification Problems

- Let $G = (V, E)$ and Π be a graph and a graph property, respectively.

Graph Modification Problems

- Let $G = (V, E)$ and Π be a graph and a graph property, respectively.
- **Graph modification problems** are those in which some changes in $E(G)$ $(V(G))$ are required in order to obtain a new graph satisfying Π.

Graph Modification Problems

- Let $G = (V, E)$ and Π be a graph and a graph property, respectively.
- **Graph modification problems** are those in which some changes in $E(G)$ $(V(G))$ are required in order to obtain a new graph satisfying Π.
  - *Completion* – it only allows the addition of edges (vertices).
  - *Deletion* – it only allows the deletion of edges (vertices).
  - *Editing* – it allows additions and deletions of edges (vertices).

Graph Modification Problems

- Let $G = (V, E)$ and Π be a graph and a graph property, respectively.
- **Graph modification problems** are those in which some changes in $E(G)$ $(V(G))$ are required in order to obtain a new graph satisfying Π.
  - *Completion* – it only allows the addition of edges (vertices).
  - *Deletion* – it only allows the deletion of edges (vertices).
  - *Editing* – it allows additions and deletions of edges (vertices).
- For a set $M \subseteq E(G)$, if $G - M$ is bipartite, then M is said to be an **edge bipartizing set** of G.

Graph Modification Problems

- Let $G = (V, E)$ and Π be a graph and a graph property, respectively.
- **Graph modification problems** are those in which some changes in $E(G)$ $(V(G))$ are required in order to obtain a new graph satisfying Π.
  - *Completion* – it only allows the addition of edges (vertices).
  - *Deletion* – it only allows the deletion of edges (vertices).
  - *Editing* – it allows additions and deletions of edges (vertices).
- For a set $M \subseteq E(G)$, if $G - M$ is bipartite, then M is said to be an **edge bipartizing set** of G.
  - All graphs admit an edge bipartizing set.

Graph Modification Problems

- Let $G = (V, E)$ and Π be a graph and a graph property, respectively.
- **Graph modification problems** are those in which some changes in $E(G)$ $(V(G))$ are required in order to obtain a new graph satisfying Π.
  - *Completion* – it only allows the addition of edges (vertices).
  - *Deletion* – it only allows the deletion of edges (vertices).
  - *Editing* – it allows additions and deletions of edges (vertices).
- For a set $M \subseteq E(G)$, if $G - M$ is bipartite, then M is said to be an **edge bipartizing set** of G.
  - All graphs admit an edge bipartizing set.
  - Several works concern on minimization versions.

Graph Modification Problems

- Let $G = (V, E)$ and Π be a graph and a graph property, respectively.
- **Graph modification problems** are those in which some changes in $E(G)$ $(V(G))$ are required in order to obtain a new graph satisfying Π.
  - *Completion* – it only allows the addition of edges (vertices).
  - *Deletion* – it only allows the deletion of edges (vertices).
  - *Editing* – it allows additions and deletions of edges (vertices).
- For a set $M \subseteq E(G)$, if $G - M$ is bipartite, then M is said to be an **edge bipartizing set** of G.
  - All graphs admit an edge bipartizing set.
  - Several works concern on minimization versions.
  - **What happens with restricted edge bipartizing sets?**

Bipartizing Matching

- Given a finite, simple, and undirected graph G, if M is an edge bipartizing set of G that is a matching, then we call M as **bipartizing matching**.

Bipartizing Matching

- Given a finite, simple, and undirected graph G, if M is an edge bipartizing set of G that is a matching, then we call M as **bipartizing matching**.
- Let $\mathcal{BM}$ be the family of all graphs admitting a bipartizing matching.

Bipartizing Matching

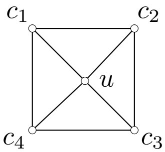
- Given a finite, simple, and undirected graph G, if M is an edge bipartizing set of G that is a matching, then we call M as **bipartizing matching**.
- Let $\boldsymbol{\mathcal{BM}}$ be the family of all graphs admitting a bipartizing matching.
- Our goal is to decide whether $G \in \mathcal{BM}$. Let us call it as BM problem.

Bipartizing Matching

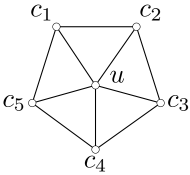
- Given a finite, simple, and undirected graph G, if M is an edge bipartizing set of G that is a matching, then we call M as **bipartizing matching**.
- Let $\boldsymbol{\mathcal{BM}}$ be the family of all graphs admitting a bipartizing matching.
- Our goal is to decide whether $G \in \mathcal{BM}$. Let us call it as BM problem.
- Observe that $\mathcal{BM}$ is closed under taking subgraphs.

Bipartizing Matching

- Given a finite, simple, and undirected graph G, if M is an edge bipartizing set of G that is a matching, then we call M as **bipartizing matching**.
- Let $\boldsymbol{\mathcal{BM}}$ be the family of all graphs admitting a bipartizing matching.
- Our goal is to decide whether $G \in \mathcal{BM}$. Let us call it as BM problem.
- Observe that $\mathcal{BM}$ is closed under taking subgraphs.
- Every $G \in \mathcal{BM}$ does not admit any **wheel** as subgraph.



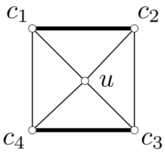
(a) W_5.



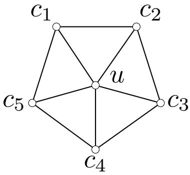
(b) W_6.

Bipartizing Matching

- Given a finite, simple, and undirected graph G, if M is an edge bipartizing set of G that is a matching, then we call M as **bipartizing matching**.
- Let $\boldsymbol{\mathcal{BM}}$ be the family of all graphs admitting a bipartizing matching.
- Our goal is to decide whether $G \in \mathcal{BM}$. Let us call it as BM problem.
- Observe that $\mathcal{BM}$ is closed under taking subgraphs.
- Every $G \in \mathcal{BM}$ does not admit any **wheel** as subgraph.



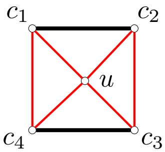
(a) W_5.



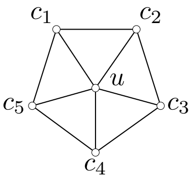
(b) W_6.

Bipartizing Matching

- Given a finite, simple, and undirected graph G, if M is an edge bipartizing set of G that is a matching, then we call M as **bipartizing matching**.
- Let $\boldsymbol{\mathcal{BM}}$ be the family of all graphs admitting a bipartizing matching.
- Our goal is to decide whether $G \in \mathcal{BM}$. Let us call it as BM problem.
- Observe that $\mathcal{BM}$ is closed under taking subgraphs.
- Every $G \in \mathcal{BM}$ does not admit any **wheel** as subgraph.



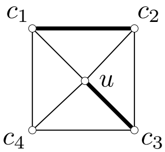
(a) W_5.



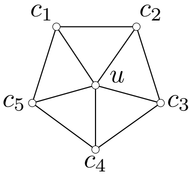
(b) W_6.

Bipartizing Matching

- Given a finite, simple, and undirected graph G, if M is an edge bipartizing set of G that is a matching, then we call M as **bipartizing matching**.
- Let $\boldsymbol{\mathcal{BM}}$ be the family of all graphs admitting a bipartizing matching.
- Our goal is to decide whether $G \in \mathcal{BM}$. Let us call it as BM problem.
- Observe that $\mathcal{BM}$ is closed under taking subgraphs.
- Every $G \in \mathcal{BM}$ does not admit any **wheel** as subgraph.



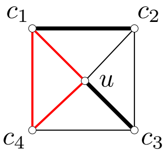
(a) W_5.



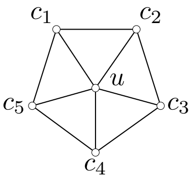
(b) W_6.

Bipartizing Matching

- Given a finite, simple, and undirected graph G, if M is an edge bipartizing set of G that is a matching, then we call M as **bipartizing matching**.
- Let $\boldsymbol{\mathcal{BM}}$ be the family of all graphs admitting a bipartizing matching.
- Our goal is to decide whether $G \in \mathcal{BM}$. Let us call it as BM problem.
- Observe that $\mathcal{BM}$ is closed under taking subgraphs.
- Every $G \in \mathcal{BM}$ does not admit any **wheel** as subgraph.



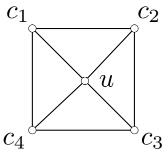
(a) W_5.



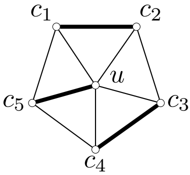
(b) W_6.

Bipartizing Matching

- Given a finite, simple, and undirected graph G, if M is an edge bipartizing set of G that is a matching, then we call M as **bipartizing matching**.
- Let $\boldsymbol{\mathcal{BM}}$ be the family of all graphs admitting a bipartizing matching.
- Our goal is to decide whether $G \in \mathcal{BM}$. Let us call it as BM problem.
- Observe that $\mathcal{BM}$ is closed under taking subgraphs.
- Every $G \in \mathcal{BM}$ does not admit any **wheel** as subgraph.



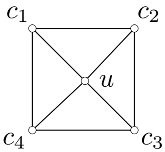
(a) W_5.



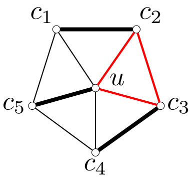
(b) W_6.

Bipartizing Matching

- Given a finite, simple, and undirected graph G, if M is an edge bipartizing set of G that is a matching, then we call M as **bipartizing matching**.
- Let $\boldsymbol{\mathcal{BM}}$ be the family of all graphs admitting a bipartizing matching.
- Our goal is to decide whether $G \in \mathcal{BM}$. Let us call it as BM problem.
- Observe that $\mathcal{BM}$ is closed under taking subgraphs.
- Every $G \in \mathcal{BM}$ does not admit any **wheel** as subgraph.



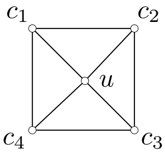
(a) W_5.



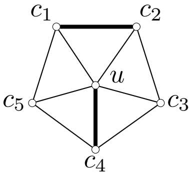
(b) W_6.

Bipartizing Matching

- Given a finite, simple, and undirected graph G, if M is an edge bipartizing set of G that is a matching, then we call M as **bipartizing matching**.
- Let $\boldsymbol{\mathcal{BM}}$ be the family of all graphs admitting a bipartizing matching.
- Our goal is to decide whether $G \in \mathcal{BM}$. Let us call it as BM problem.
- Observe that $\mathcal{BM}$ is closed under taking subgraphs.
- Every $G \in \mathcal{BM}$ does not admit any **wheel** as subgraph.



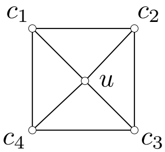
(a) W_5.



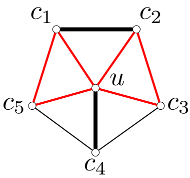
(b) W_6.

Bipartizing Matching

- Given a finite, simple, and undirected graph G, if M is an edge bipartizing set of G that is a matching, then we call M as **bipartizing matching**.
- Let $\boldsymbol{\mathcal{BM}}$ be the family of all graphs admitting a bipartizing matching.
- Our goal is to decide whether $G \in \mathcal{BM}$. Let us call it as BM problem.
- Observe that $\mathcal{BM}$ is closed under taking subgraphs.
- Every $G \in \mathcal{BM}$ does not admit any **wheel** as subgraph.



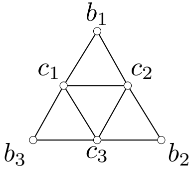
(a) W_5.



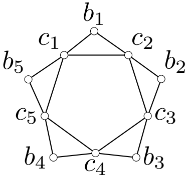
(b) W_6.

Bipartizing Matching

- Given a finite, simple, and undirected graph G, if M is an edge bipartizing set of G that is a matching, then we call M as **bipartizing matching**.
- Let $\boldsymbol{\mathcal{BM}}$ be the family of all graphs admitting a bipartizing matching.
- Our goal is to decide whether $G \in \mathcal{BM}$. Let us call it as BM problem.
- Observe that $\mathcal{BM}$ is closed under taking subgraphs.
- Every $G \in \mathcal{BM}$ does not admit any **k-pool** as subgraph, $k \geq 3$ and odd.



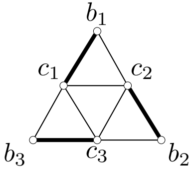
(a) 3-pool.



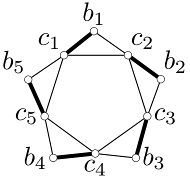
(b) 5-pool.

Bipartizing Matching

- Given a finite, simple, and undirected graph G, if M is an edge bipartizing set of G that is a matching, then we call M as **bipartizing matching**.
- Let $\boldsymbol{\mathcal{BM}}$ be the family of all graphs admitting a bipartizing matching.
- Our goal is to decide whether $G \in \mathcal{BM}$. Let us call it as BM problem.
- Observe that $\mathcal{BM}$ is closed under taking subgraphs.
- Every $G \in \mathcal{BM}$ does not admit any $\mathbf{k}$-**pool** as subgraph, $k \geq 3$ and odd.



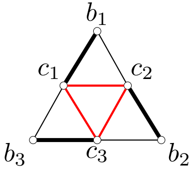
(a) 3-pool.



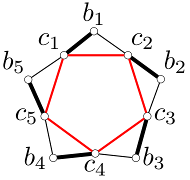
(b) 5-pool.

Bipartizing Matching

- Given a finite, simple, and undirected graph G, if M is an edge bipartizing set of G that is a matching, then we call M as **bipartizing matching**.
- Let $\boldsymbol{\mathcal{BM}}$ be the family of all graphs admitting a bipartizing matching.
- Our goal is to decide whether $G \in \mathcal{BM}$. Let us call it as BM problem.
- Observe that $\mathcal{BM}$ is closed under taking subgraphs.
- Every $G \in \mathcal{BM}$ does not admit any $\mathbf{k}$**-pool** as subgraph, $k \geq 3$ and odd.



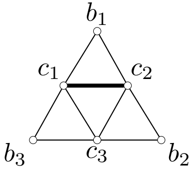
(a) 3-pool.



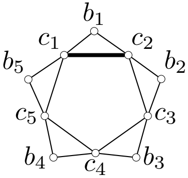
(b) 5-pool.

Bipartizing Matching

- Given a finite, simple, and undirected graph G, if M is an edge bipartizing set of G that is a matching, then we call M as **bipartizing matching**.
- Let $\boldsymbol{\mathcal{BM}}$ be the family of all graphs admitting a bipartizing matching.
- Our goal is to decide whether $G \in \mathcal{BM}$. Let us call it as BM problem.
- Observe that $\mathcal{BM}$ is closed under taking subgraphs.
- Every $G \in \mathcal{BM}$ does not admit any $\mathbf{k}$**-pool** as subgraph, $k \geq 3$ and odd.



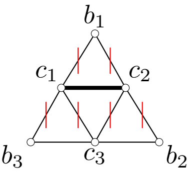
(a) 3-pool.



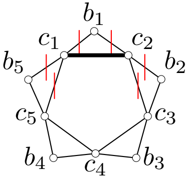
(b) 5-pool.

Bipartizing Matching

- Given a finite, simple, and undirected graph G, if M is an edge bipartizing set of G that is a matching, then we call M as **bipartizing matching**.
- Let $\boldsymbol{\mathcal{BM}}$ be the family of all graphs admitting a bipartizing matching.
- Our goal is to decide whether $G \in \mathcal{BM}$. Let us call it as BM problem.
- Observe that $\mathcal{BM}$ is closed under taking subgraphs.
- Every $G \in \mathcal{BM}$ does not admit any $\mathbf{k}$**-pool** as subgraph, $k \geq 3$ and odd.



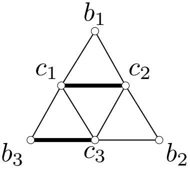
(a) 3-pool.



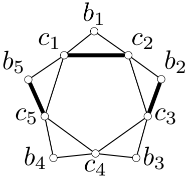
(b) 5-pool.

Bipartizing Matching

- Given a finite, simple, and undirected graph G, if M is an edge bipartizing set of G that is a matching, then we call M as **bipartizing matching**.
- Let $\boldsymbol{\mathcal{BM}}$ be the family of all graphs admitting a bipartizing matching.
- Our goal is to decide whether $G \in \mathcal{BM}$. Let us call it as BM problem.
- Observe that $\mathcal{BM}$ is closed under taking subgraphs.
- Every $G \in \mathcal{BM}$ does not admit any $\mathbf{k}$-**pool** as subgraph, $k \geq 3$ and odd.



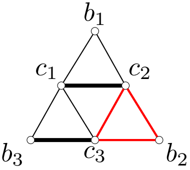
(a) 3-pool.



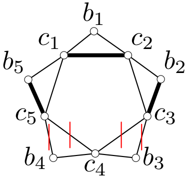
(b) 5-pool.

Bipartizing Matching

- Given a finite, simple, and undirected graph G, if M is an edge bipartizing set of G that is a matching, then we call M as **bipartizing matching**.
- Let $\boldsymbol{\mathcal{BM}}$ be the family of all graphs admitting a bipartizing matching.
- Our goal is to decide whether $G \in \mathcal{BM}$. Let us call it as BM problem.
- Observe that $\mathcal{BM}$ is closed under taking subgraphs.
- Every $G \in \mathcal{BM}$ does not admit any **k-pool** as subgraph, $k \geq 3$ and odd.



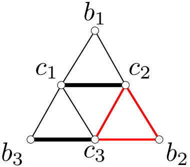
(a) 3-pool.



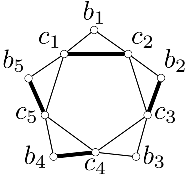
(b) 5-pool.

Bipartizing Matching

- Given a finite, simple, and undirected graph G, if M is an edge bipartizing set of G that is a matching, then we call M as **bipartizing matching**.
- Let $\boldsymbol{\mathcal{BM}}$ be the family of all graphs admitting a bipartizing matching.
- Our goal is to decide whether $G \in \mathcal{BM}$. Let us call it as BM problem.
- Observe that $\mathcal{BM}$ is closed under taking subgraphs.
- Every $G \in \mathcal{BM}$ does not admit any **k-pool** as subgraph, $k \geq 3$ and odd.



(a) 3-pool.



(b) 5-pool.

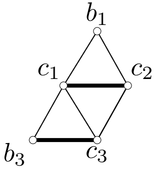
Bipartizing Matching

- Given a finite, simple, and undirected graph G, if M is an edge bipartizing set of G that is a matching, then we call M as **bipartizing matching**.
- Let $\boldsymbol{\mathcal{BM}}$ be the family of all graphs admitting a bipartizing matching.
- Our goal is to decide whether $G \in \mathcal{BM}$. Let us call it as BM problem.
- Observe that $\mathcal{BM}$ is closed under taking subgraphs.
- Every $G \in \mathcal{BM}$ does not admit any $\mathbf{k}$**-pool** as subgraph, $k \geq 3$ and odd.

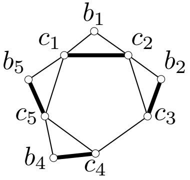
(a) 3-pool. (b) 5-pool.

Bipartizing Matching

- Given a finite, simple, and undirected graph G, if M is an edge bipartizing set of G that is a matching, then we call M as **bipartizing matching**.
- Let $\boldsymbol{\mathcal{BM}}$ be the family of all graphs admitting a bipartizing matching.
- Our goal is to decide whether $G \in \mathcal{BM}$. Let us call it as BM problem.
- Observe that $\mathcal{BM}$ is closed under taking subgraphs.
- Note that removing a **border** from a k-pool, we obtain a graph in $\mathcal{BM}$.



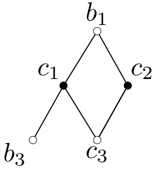
(a) 3-pool.



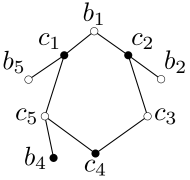
(b) 5-pool.

Bipartizing Matching

- Given a finite, simple, and undirected graph G, if M is an edge bipartizing set of G that is a matching, then we call M as **bipartizing matching**.
- Let $\boldsymbol{\mathcal{BM}}$ be the family of all graphs admitting a bipartizing matching.
- Our goal is to decide whether $G \in \mathcal{BM}$. Let us call it as BM problem.
- Observe that $\mathcal{BM}$ is closed under taking subgraphs.
- Note that removing a **border** from a k-pool, we obtain a graph in $\mathcal{BM}$.



(a) 3-pool.



(b) 5-pool.

Related Works I

- **Schaefer (1978):** proved the NP-completeness of deciding whether a given graph G admits a removal of a perfect matching in order to obtain a bipartite graph, even for planar cubic graphs.
- **Furmańczyk, Kubale, and Radziszowski (2016):** considered vertex bipartization of cubic graphs by removing an independent set.
- **Bonamy et all. (2018):** considered the Independent Feedback Vertex Set problem on P_5-free Graphs.

Related Works II

A $(\boldsymbol{k}, \boldsymbol{d})$-coloring of a graph G is a k-vertex coloring such that each vertex has at most d neighbors with same color as itself.

- Hence $G \in \mathcal{BM}$ if and only if G admits a $(2, 1)$-coloring.
- Is is also known as **defective coloring**.

- **Eaton and Hull (1999):** proved that all triangle-free outerplanar graphs are $(2, 1)$-colorable.
- **Borodin, Kostochka, and Yancey (2013):** studied $(2, 1)$-colorable graphs with respect to the maximum average degree and its relation with the girth.
- **Angelini et al. (2017):** present a linear-time algorithm which determines that partial 2-trees, a subclass of planar graphs, are $(2, 1)$-colorable.

Related Works III

- **Lima et al. (2017):** considered the problem of deciding whether a given graph G admits a removal of a matching in order to obtain a forest.
- **Protti and Souza (2017):** consider characterizations of some graph classes admitting the removal of a matching in order to obtain a forest.

A Linear-Time Algorithm for Subcubic Graphs I

- A subcubic graph G is one that the maximum degree is at most 3.

A Linear-Time Algorithm for Subcubic Graphs I

- A subcubic graph G is one that the maximum degree is at most 3.
- We show that every subcubic graph belongs to $\mathcal{BM}$.

A Linear-Time Algorithm for Subcubic Graphs I

- A subcubic graph G is one that the maximum degree is at most 3.
- We show that every subcubic graph belongs to $\mathcal{BM}$.
- This result can be obtained by results from Erdős (1965), Lovász (1966), and Bondy and Locke (1986) obtained in different contexts.

A Linear-Time Algorithm for Subcubic Graphs I

- A subcubic graph G is one that the maximum degree is at most 3.
- We show that every subcubic graph belongs to $\mathcal{BM}$.
- This result can be obtained by results from Erdős (1965), Lovász (1966), and Bondy and Locke (1986) obtained in different contexts.
- Our algorithm is based on the fact that, for any bipartition (A, B) of $V(G)$, if the edges from A to B define a maximal edge cut of G, then the remaining edges define a matching.

A Linear-Time Algorithm for Subcubic Graphs I

- A subcubic graph G is one that the maximum degree is at most 3.
- We show that every subcubic graph belongs to $\mathcal{BM}$.
- This result can be obtained by results from Erdős (1965), Lovász (1966), and Bondy and Locke (1986) obtained in different contexts.
- Our algorithm is based on the fact that, for any bipartition (A, B) of $V(G)$, if the edges from A to B define a maximal edge cut of G, then the remaining edges define a matching.
- Hence we swap vertices between the parts A and B in order to obtain a maximal edge cut.

A Linear-Time Algorithm for Subcubic Graphs II

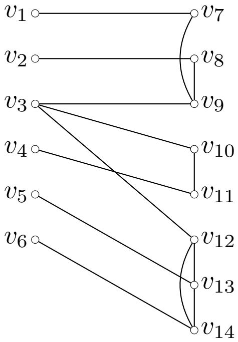
- We start by setting A as a maximal independent set and $B = V(G) \setminus A$.
- Moreover, we guarantee that the maximum degree in $G[A]$ is at most 1 for all changes.

Algorithm 1: Subcubic graphs.

$A \leftarrow$ A maximal independent set of G
$B \leftarrow V(G) \setminus A$

while there exists a vertex $v \in B$ of type $(1, 2)$ **do**
  $u \leftarrow N_{G[A]}(v)$
  if u is of type $(2, 0)$ or $(3, 0)$ **then**
    $B \leftarrow B \setminus \{v\}$
    $A \leftarrow A \cup \{v\}$
  else
    $B \leftarrow \{B \setminus \{v\}\} \cup \{u\}$
    $A \leftarrow \{A \setminus \{u\}\} \cup \{v\}$
    if $z \in N_{G[B]}(v)$ is of type $(0, 2)$ or $(0, 3)$ **then**
      $B \leftarrow B \setminus \{z\}$
      $A \leftarrow A \cup \{z\}$

return $E\,(G[A] \cup G[B])$

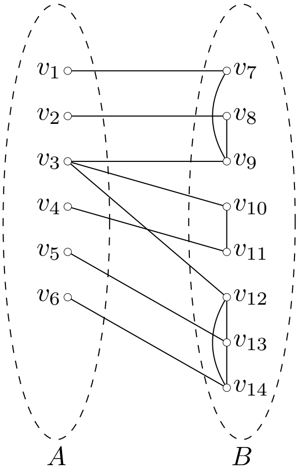


A Linear-Time Algorithm for Subcubic Graphs II

- We start by setting A as a maximal independent set and $B = V(G) \setminus A$.
- Moreover, we guarantee that the maximum degree in $G[A]$ is at most 1 for all changes.

Algorithm 1: Subcubic graphs.

$A \leftarrow$ A maximal independent set of G
$B \leftarrow V(G) \setminus A \Longleftarrow$
while there exists a vertex $v \in B$ of type $(1,2)$ **do**
  $u \leftarrow N_{G[A]}(v)$
  if u is of type $(2,0)$ or $(3,0)$ **then**
    $B \leftarrow B \setminus \{v\}$
    $A \leftarrow A \cup \{v\}$
  else
    $B \leftarrow \{B \setminus \{v\}\} \cup \{u\}$
    $A \leftarrow \{A \setminus \{u\}\} \cup \{v\}$
    if $z \in N_{G[B]}(v)$ is of type $(0,2)$ or $(0,3)$ **then**
      $B \leftarrow B \setminus \{z\}$
      $A \leftarrow A \cup \{z\}$
return $E\,(G[A] \cup G[B])$



A Linear-Time Algorithm for Subcubic Graphs II

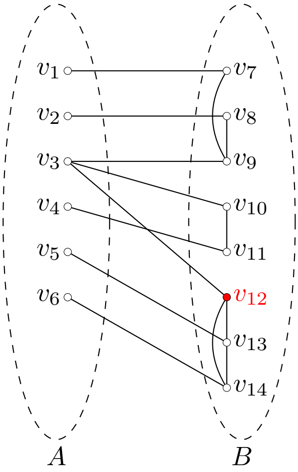
- We start by setting A as a maximal independent set and $B = V(G) \setminus A$.
- Moreover, we guarantee that the maximum degree in $G[A]$ is at most 1 for all changes.

Algorithm 1: Subcubic graphs.

$A \leftarrow$ A maximal independent set of G
$B \leftarrow V(G) \setminus A$

while there exists a vertex $v \in B$ of type $(1,2)$ **do**
    $u \leftarrow N_{G[A]}(v)$
    if u is of type $(2,0)$ or $(3,0)$ **then**
        $B \leftarrow B \setminus \{v\}$
        $A \leftarrow A \cup \{v\}$
    else
        $B \leftarrow \{B \setminus \{v\}\} \cup \{u\}$
        $A \leftarrow \{A \setminus \{u\}\} \cup \{v\}$
        if $z \in N_{G[B]}(v)$ is of type $(0,2)$ or $(0,3)$ **then**
            $B \leftarrow B \setminus \{z\}$
            $A \leftarrow A \cup \{z\}$

return $E\left(G[A] \cup G[B]\right)$

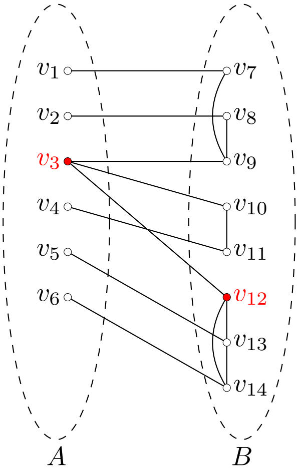


A Linear-Time Algorithm for Subcubic Graphs II

- We start by setting A as a maximal independent set and $B = V(G) \setminus A$.
- Moreover, we guarantee that the maximum degree in $G[A]$ is at most 1 for all changes.

Algorithm 1: Subcubic graphs.

$A \leftarrow$ A maximal independent set of G
$B \leftarrow V(G) \setminus A$
while there exists a vertex $v \in B$ of type $(1, 2)$ **do**
    $u \leftarrow N_{G[A]}(v) \Longleftarrow$
    if u is of type $(2, 0)$ or $(3, 0)$ **then**
        $B \leftarrow B \setminus \{v\}$
        $A \leftarrow A \cup \{v\}$
    else
        $B \leftarrow \{B \setminus \{v\}\} \cup \{u\}$
        $A \leftarrow \{A \setminus \{u\}\} \cup \{v\}$
        if $z \in N_{G[B]}(v)$ is of type $(0, 2)$ or $(0, 3)$ **then**
            $B \leftarrow B \setminus \{z\}$
            $A \leftarrow A \cup \{z\}$
return $E(G[A] \cup G[B])$



A Linear-Time Algorithm for Subcubic Graphs II

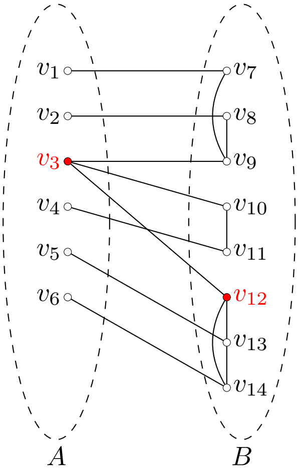
- We start by setting A as a maximal independent set and $B = V(G) \setminus A$.
- Moreover, we guarantee that the maximum degree in $G[A]$ is at most 1 for all changes.

Algorithm 1: Subcubic graphs.

$A \leftarrow$ A maximal independent set of G
$B \leftarrow V(G) \setminus A$

while there exists a vertex $v \in B$ of type $(1,2)$ **do**
  $u \leftarrow N_{G[A]}(v)$
  if u is of type $(2,0)$ or $(3,0)$ **then**
    $B \leftarrow B \setminus \{v\}$
    $A \leftarrow A \cup \{v\}$
  else
    $B \leftarrow \{B \setminus \{v\}\} \cup \{u\}$
    $A \leftarrow \{A \setminus \{u\}\} \cup \{v\}$
    if $z \in N_{G[B]}(v)$ is of type $(0,2)$ or $(0,3)$ **then**
      $B \leftarrow B \setminus \{z\}$
      $A \leftarrow A \cup \{z\}$

return $E\,(G[A] \cup G[B])$



A Linear-Time Algorithm for Subcubic Graphs II

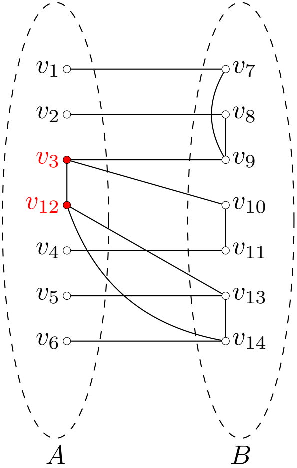
- We start by setting A as a maximal independent set and $B = V(G) \setminus A$.
- Moreover, we guarantee that the maximum degree in $G[A]$ is at most 1 for all changes.

Algorithm 1: Subcubic graphs.

$A \leftarrow$ A maximal independent set of G
$B \leftarrow V(G) \setminus A$

while there exists a vertex $v \in B$ of type $(1, 2)$ **do**
    $u \leftarrow N_{G[A]}(v)$
    if u is of type $(2, 0)$ or $(3, 0)$ **then**
        $B \leftarrow B \setminus \{v\}$
        $A \leftarrow A \cup \{v\} \Longleftarrow$
    else
        $B \leftarrow \{B \setminus \{v\}\} \cup \{u\}$
        $A \leftarrow \{A \setminus \{u\}\} \cup \{v\}$
        if $z \in N_{G[B]}(v)$ is of type $(0, 2)$ or $(0, 3)$ **then**
            $B \leftarrow B \setminus \{z\}$
            $A \leftarrow A \cup \{z\}$

return $E\,(G[A] \cup G[B])$

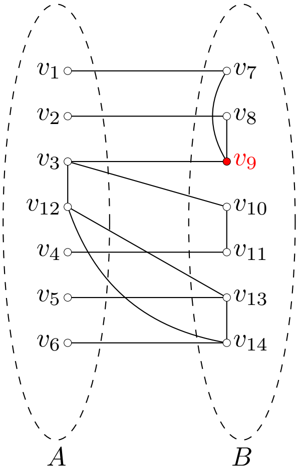


A Linear-Time Algorithm for Subcubic Graphs II

- We start by setting A as a maximal independent set and $B = V(G) \setminus A$.
- Moreover, we guarantee that the maximum degree in $G[A]$ is at most 1 for all changes.

Algorithm 1: Subcubic graphs.

$A \leftarrow$ A maximal independent set of G
$B \leftarrow V(G) \setminus A$
while there exists a vertex $v \in B$ of type $(1,2)$ **do**
  $u \leftarrow N_{G[A]}(v)$
  if u is of type $(2,0)$ or $(3,0)$ **then**
    $B \leftarrow B \setminus \{v\}$
    $A \leftarrow A \cup \{v\}$
  else
    $B \leftarrow \{B \setminus \{v\}\} \cup \{u\}$
    $A \leftarrow \{A \setminus \{u\}\} \cup \{v\}$
    if $z \in N_{G[B]}(v)$ is of type $(0,2)$ or $(0,3)$ **then**
      $B \leftarrow B \setminus \{z\}$
      $A \leftarrow A \cup \{z\}$
return $E(G[A] \cup G[B])$

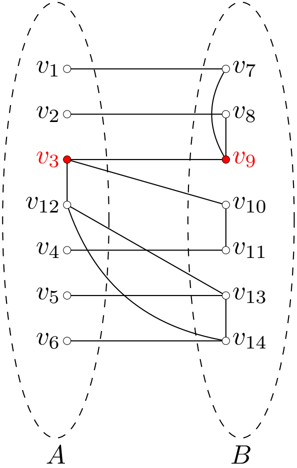


A Linear-Time Algorithm for Subcubic Graphs II

- We start by setting A as a maximal independent set and $B = V(G) \setminus A$.
- Moreover, we guarantee that the maximum degree in $G[A]$ is at most 1 for all changes.

Algorithm 1: Subcubic graphs.

$A \leftarrow$ A maximal independent set of G
$B \leftarrow V(G) \setminus A$
while there exists a vertex $v \in B$ of type $(1,2)$ **do**
    $u \leftarrow N_{G[A]}(v) \Longleftarrow$
    if u is of type $(2,0)$ or $(3,0)$ **then**
        $B \leftarrow B \setminus \{v\}$
        $A \leftarrow A \cup \{v\}$
    else
        $B \leftarrow \{B \setminus \{v\}\} \cup \{u\}$
        $A \leftarrow \{A \setminus \{u\}\} \cup \{v\}$
        if $z \in N_{G[B]}(v)$ is of type $(0,2)$ or $(0,3)$ **then**
            $B \leftarrow B \setminus \{z\}$
            $A \leftarrow A \cup \{z\}$
return $E\,(G[A] \cup G[B])$

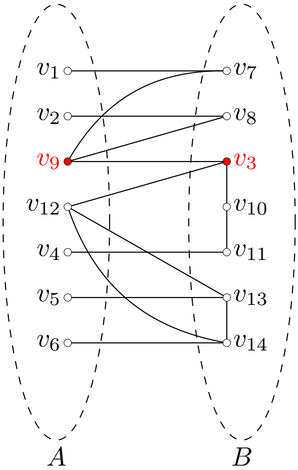


A Linear-Time Algorithm for Subcubic Graphs II

- We start by setting A as a maximal independent set and $B = V(G) \setminus A$.
- Moreover, we guarantee that the maximum degree in $G[A]$ is at most 1 for all changes.

Algorithm 1: Subcubic graphs.

```
A ← A maximal independent set of G
B ← V(G) \ A
while there exists a vertex v ∈ B of type (1,2) do
    u ← N_{G[A]}(v)
    if u is of type (2,0) or (3,0) then
        B ← B \ {v}
        A ← A ∪ {v}
    else
        B ← {B \ {v}} ∪ {u}
        A ← {A \ {u}} ∪ {v}
        if z ∈ N_{G[B]}(v) is of type (0,2) or (0,3) then
            B ← B \ {z}
            A ← A ∪ {z}
return E(G[A] ∪ G[B])
```

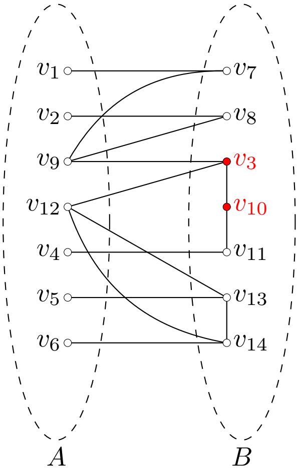


A Linear-Time Algorithm for Subcubic Graphs II

- We start by setting A as a maximal independent set and $B = V(G) \setminus A$.
- Moreover, we guarantee that the maximum degree in $G[A]$ is at most 1 for all changes.

Algorithm 1: Subcubic graphs.

$A \leftarrow$ A maximal independent set of G
$B \leftarrow V(G) \setminus A$
while there exists a vertex $v \in B$ of type $(1,2)$ **do**
    $u \leftarrow N_{G[A]}(v)$
    if u is of type $(2,0)$ or $(3,0)$ **then**
        $B \leftarrow B \setminus \{v\}$
        $A \leftarrow A \cup \{v\}$
    else
        $B \leftarrow \{B \setminus \{v\}\} \cup \{u\}$
        $A \leftarrow \{A \setminus \{u\}\} \cup \{v\}$;
        if $z \in N_{G[B]}(v)$ is of type $(0,2)$ or $(0,3)$ **then**
            $B \leftarrow B \setminus \{z\}$
            $A \leftarrow A \cup \{z\}$
return $E(G[A] \cup G[B])$

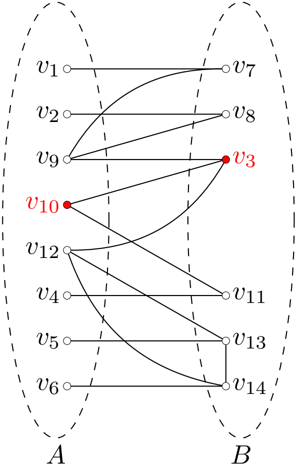


A Linear-Time Algorithm for Subcubic Graphs II

- We start by setting A as a maximal independent set and $B = V(G) \setminus A$.
- Moreover, we guarantee that the maximum degree in $G[A]$ is at most 1 for all changes.

Algorithm 1: Subcubic graphs.

$A \leftarrow$ A maximal independent set of G
$B \leftarrow V(G) \setminus A$
while there exists a vertex $v \in B$ of type $(1,2)$ **do**
  $u \leftarrow N_{G[A]}(v)$
  if u is of type $(2,0)$ or $(3,0)$ **then**
    $B \leftarrow B \setminus \{v\}$
    $A \leftarrow A \cup \{v\}$
  else
    $B \leftarrow \{B \setminus \{v\}\} \cup \{u\}$
    $A \leftarrow \{A \setminus \{u\}\} \cup \{v\}$;
    if $z \in N_{G[B]}(v)$ is of type $(0,2)$ or $(0,3)$ **then**
      $B \leftarrow B \setminus \{z\}$
      $A \leftarrow A \cup \{z\} \Longleftarrow$
return $E\left(G[A] \cup G[B]\right)$

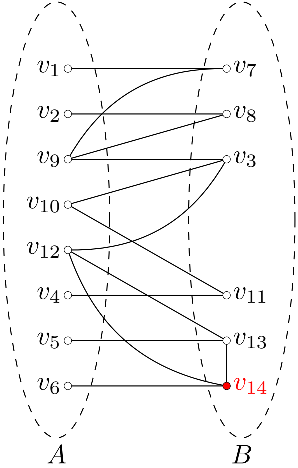


A Linear-Time Algorithm for Subcubic Graphs II

- We start by setting A as a maximal independent set and $B = V(G) \setminus A$.
- Moreover, we guarantee that the maximum degree in $G[A]$ is at most 1 for all changes.

Algorithm 1: Subcubic graphs.

$A \leftarrow$ A maximal independent set of G
$B \leftarrow V(G) \setminus A$
while there exists a vertex $v \in B$ of type $(1,2)$ **do**
    $u \leftarrow N_{G[A]}(v)$
    if u is of type $(2,0)$ or $(3,0)$ **then**
        $B \leftarrow B \setminus \{v\}$
        $A \leftarrow A \cup \{v\}$
    else
        $B \leftarrow \{B \setminus \{v\}\} \cup \{u\}$
        $A \leftarrow \{A \setminus \{u\}\} \cup \{v\}$;
        if $z \in N_{G[B]}(v)$ is of type $(0,2)$ or $(0,3)$ **then**
            $B \leftarrow B \setminus \{z\}$
            $A \leftarrow A \cup \{z\}$
return $E\,(G[A] \cup G[B])$

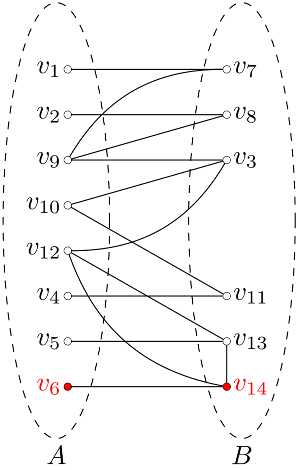


A Linear-Time Algorithm for Subcubic Graphs II

- We start by setting A as a maximal independent set and $B = V(G) \setminus A$.
- Moreover, we guarantee that the maximum degree in $G[A]$ is at most 1 for all changes.

Algorithm 1: Subcubic graphs.

$A \leftarrow$ A maximal independent set of G
$B \leftarrow V(G) \setminus A$
while there exists a vertex $v \in B$ of type $(1,2)$ **do**
  $u \leftarrow N_{G[A]}(v) \Longleftarrow$
  if u is of type $(2,0)$ or $(3,0)$ **then**
    $B \leftarrow B \setminus \{v\}$
    $A \leftarrow A \cup \{v\}$
  else
    $B \leftarrow \{B \setminus \{v\}\} \cup \{u\}$
    $A \leftarrow \{A \setminus \{u\}\} \cup \{v\}$;
    if $z \in N_{G[B]}(v)$ is of type $(0,2)$ or $(0,3)$ **then**
      $B \leftarrow B \setminus \{z\}$
      $A \leftarrow A \cup \{z\}$
return $E(G[A] \cup G[B])$



A Linear-Time Algorithm for Subcubic Graphs II

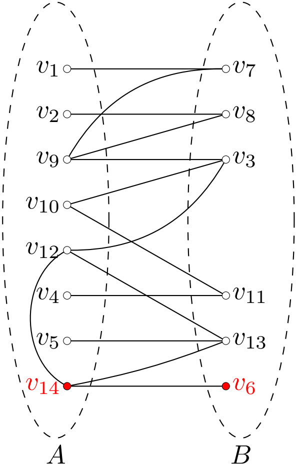
- We start by setting A as a maximal independent set and $B = V(G) \setminus A$.
- Moreover, we guarantee that the maximum degree in $G[A]$ is at most 1 for all changes.

Algorithm 1: Subcubic graphs.

$A \leftarrow$ A maximal independent set of G
$B \leftarrow V(G) \setminus A$

while there exists a vertex $v \in B$ of type $(1,2)$ **do**
  $u \leftarrow N_{G[A]}(v)$
  if u is of type $(2,0)$ or $(3,0)$ **then**
    $B \leftarrow B \setminus \{v\}$
    $A \leftarrow A \cup \{v\}$ $\Longleftarrow$
  else
    $B \leftarrow \{B \setminus \{v\}\} \cup \{u\}$
    $A \leftarrow \{A \setminus \{u\}\} \cup \{v\}$;
    if $z \in N_{G[B]}(v)$ is of type $(0,2)$ or $(0,3)$ **then**
      $B \leftarrow B \setminus \{z\}$
      $A \leftarrow A \cup \{z\}$

return $E\,(G[A] \cup G[B])$



A Linear-Time Algorithm for Subcubic Graphs II

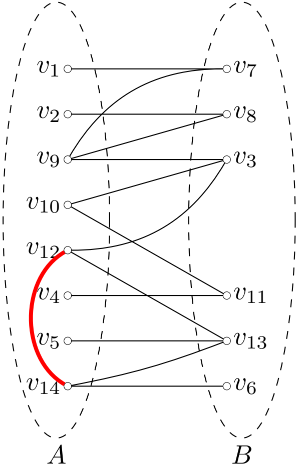
- We start by setting A as a maximal independent set and $B = V(G) \setminus A$.
- Moreover, we guarantee that the maximum degree in $G[A]$ is at most 1 for all changes.

Algorithm 1: Subcubic graphs.

$A \leftarrow$ A maximal independent set of G
$B \leftarrow V(G) \setminus A$

while there exists a vertex $v \in B$ of type $(1, 2)$ **do**
    $u \leftarrow N_{G[A]}(v)$
    if u is of type $(2, 0)$ or $(3, 0)$ **then**
        $B \leftarrow B \setminus \{v\}$
        $A \leftarrow A \cup \{v\}$
    else
        $B \leftarrow \{B \setminus \{v\}\} \cup \{u\}$
        $A \leftarrow \{A \setminus \{u\}\} \cup \{v\}$;
        if $z \in N_{G[B]}(v)$ is of type $(0, 2)$ or $(0, 3)$ **then**
            $B \leftarrow B \setminus \{z\}$
            $A \leftarrow A \cup \{z\}$

return $E\,(G[A] \cup G[B])$ $\Longleftarrow$



Main Theorem

- Cowen, Goddard, and Jesurum (1997) proved that it is NP-complete to determine whether a given graph is $(2, 1)$-colorable.

Main Theorem

- Cowen, Goddard, and Jesurum (1997) proved that it is NP-complete to determine whether a given graph is $(2, 1)$-colorable.
  - Even for graphs of maximum degree 4;

Main Theorem

- Cowen, Goddard, and Jesurum (1997) proved that it is NP-complete to determine whether a given graph is $(2, 1)$-colorable.
  - Even for graphs of maximum degree 4;
  - And even for planar graphs of maximum degree 5.

Main Theorem

- Cowen, Goddard, and Jesurum (1997) proved that it is NP-complete to determine whether a given graph is $(2,1)$-colorable.
  - Even for graphs of maximum degree 4;
  - And even for planar graphs of maximum degree 5.

Main theorem:

It remains NP-complete even for 3-colorable planar graphs of maximum degree 4.

Main Theorem

- Cowen, Goddard, and Jesurum (1997) proved that it is NP-complete to determine whether a given graph is $(2,1)$-colorable.
  - Even for graphs of maximum degree 4;
  - And even for planar graphs of maximum degree 5.

Main theorem:

It remains NP-complete even for 3-colorable planar graphs of maximum degree 4.

- In order to prove the Main theorem, we prove an auxiliary theorem.

Main Theorem

- Cowen, Goddard, and Jesurum (1997) proved that it is NP-complete to determine whether a given graph is $(2, 1)$-colorable.
  - Even for graphs of maximum degree 4;
  - And even for planar graphs of maximum degree 5.

Main theorem:

It remains NP-complete even for 3-colorable planar graphs of maximum degree 4.

- In order to prove the Main theorem, we prove an auxiliary theorem.
- Let F be a Boolean formula in 3-CNF such that:

Main Theorem

- Cowen, Goddard, and Jesurum (1997) proved that it is NP-complete to determine whether a given graph is $(2, 1)$-colorable.
  - Even for graphs of maximum degree 4;
  - And even for planar graphs of maximum degree 5.

Main theorem:

It remains NP-complete even for 3-colorable planar graphs of maximum degree 4.

- In order to prove the Main theorem, we prove an auxiliary theorem.
- Let F be a Boolean formula in 3-CNF such that:
  - $\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$ is its variable set;
  - $\mathbf{C} = \{C_1, C_2, \ldots, C_m\}$ is its clause set.

Main Theorem

- Cowen, Goddard, and Jesurum (1997) proved that it is NP-complete to determine whether a given graph is $(2,1)$-colorable.
  - Even for graphs of maximum degree 4;
  - And even for planar graphs of maximum degree 5.

Main theorem:

It remains NP-complete even for 3-colorable planar graphs of maximum degree 4.

- In order to prove the Main theorem, we prove an auxiliary theorem.
- Let F be a Boolean formula in 3-CNF such that:
  - $\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$ is its variable set;
  - $\mathbf{C} = \{C_1, C_2, \ldots, C_m\}$ is its clause set.
  - The **associated graph** $G_F = (V, E)$ of F is the bipartite graph with $V(G_F) = (\mathbf{X}, \mathbf{C})$, such that $X_iC_j \in E(G_F)$ if and only if C_j contains either x_i or $\overline{x_i}$.

Main Theorem

- Cowen, Goddard, and Jesurum (1997) proved that it is NP-complete to determine whether a given graph is $(2,1)$-colorable.
  - Even for graphs of maximum degree 4;
  - And even for planar graphs of maximum degree 5.

Main theorem:

It remains NP-complete even for 3-colorable planar graphs of maximum degree 4.

- In order to prove the Main theorem, we prove an auxiliary theorem.
- Let F be a Boolean formula in 3-CNF such that:
  - $\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$ is its variable set;
  - $\mathbf{C} = \{C_1, C_2, \ldots, C_m\}$ is its clause set.
  - The **associated graph** $G_F = (V, E)$ of F is the bipartite graph with $V(G_F) = (\mathbf{X}, \mathbf{C})$, such that $X_iC_j \in E(G_F)$ if and only if C_j contains either x_i or $\overline{x_i}$.
  - We say that F is a **planar formula** if and only if G_F is planar.

- Let PLANAR 1-IN-3-SAT$_3$ be the problem of deciding if there exists a truth assignment to a planar formula F, where:

Planar 1-In-3-SAT$_3$

- Let Planar 1-In-3-SAT$_3$ be the problem of deciding if there exists a truth assignment to a planar formula F, where:
  - Each clause has either 2 or 3 literals;

Planar 1-In-3-SAT$_3$

- Let Planar 1-In-3-SAT$_3$ be the problem of deciding if there exists a truth assignment to a planar formula F, where:
  - Each clause has either 2 or 3 literals;
  - Each variable occurs at most 3 times;

PLANAR 1-IN-3-SAT$_3$

- Let PLANAR 1-IN-3-SAT$_3$ be the problem of deciding if there exists a truth assignment to a planar formula F, where:
  - Each clause has either 2 or 3 literals;
  - Each variable occurs at most 3 times;
  - Each positive literal occurs at most twice;

Planar 1-In-3-SAT$_3$

- Let Planar 1-In-3-SAT$_3$ be the problem of deciding if there exists a truth assignment to a planar formula F, where:
  - Each clause has either 2 or 3 literals;
  - Each variable occurs at most 3 times;
  - Each positive literal occurs at most twice;
  - Every negative literal occurs at most once.

PLANAR 1-IN-3-SAT$_3$

- Let PLANAR 1-IN-3-SAT$_3$ be the problem of deciding if there exists a truth assignment to a planar formula F, where:
  - Each clause has either 2 or 3 literals;
  - Each variable occurs at most 3 times;
  - Each positive literal occurs at most twice;
  - Every negative literal occurs at most once.
  - For each clause, exactly one literal is true.

PLANAR 1-IN-3-SAT$_3$

- Let PLANAR 1-IN-3-SAT$_3$ be the problem of deciding if there exists a truth assignment to a planar formula F, where:
  - Each clause has either 2 or 3 literals;
  - Each variable occurs at most 3 times;
  - Each positive literal occurs at most twice;
  - Every negative literal occurs at most once.
  - For each clause, exactly one literal is true.

Auxiliary Theorem:

PLANAR 1-IN-3-SAT$_3$ is NP-complete.

Planar 1-In-3-SAT$_3$

- Let Planar 1-In-3-SAT$_3$ be the problem of deciding if there exists a truth assignment to a planar formula F, where:
  - Each clause has either 2 or 3 literals;
  - Each variable occurs at most 3 times;
  - Each positive literal occurs at most twice;
  - Every negative literal occurs at most once.
  - For each clause, exactly one literal is true.

Auxiliary Theorem:

Planar 1-In-3-SAT$_3$ is NP-complete.

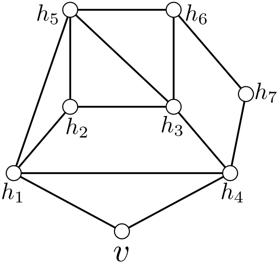
- We present a polynomial-time reduction from Planar 1-In-3-SAT$_3$ to BM.

The Head Graph

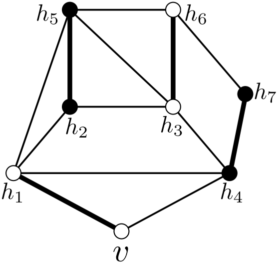
- Let us call by **head** be the following graph:

The Head Graph

- Let us call by **head** be the following graph:



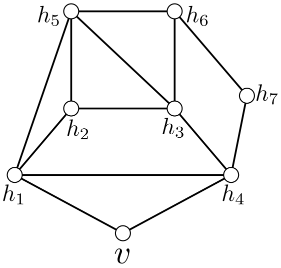
(a) The head graph H.



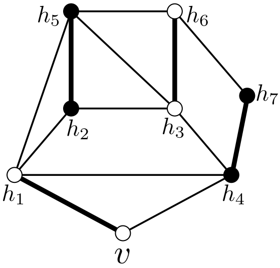
(b) The unique bipartizing matching of H.

The Head Graph

- Let us call by **head** be the following graph:



(a) The head graph H.

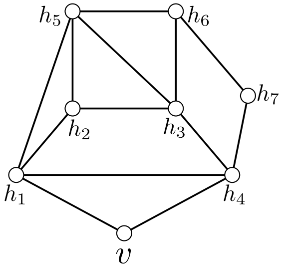


(b) The unique bipartizing matching of H.

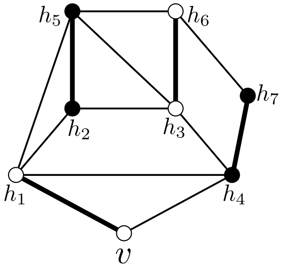
- We call v as the **neck** of the head.

The Head Graph

- Let us call by **head** be the following graph:



(a) The head graph H.

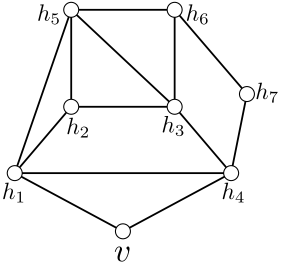


(b) The unique bipartizing matching of H.

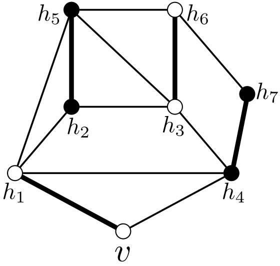
- We call v as the **neck** of the head.
- The head has only one bipartizing matching, as in Figure (b).

The Head Graph

- Let us call by **head** be the following graph:



(a) The head graph H.



(b) The unique bipartizing matching of H.

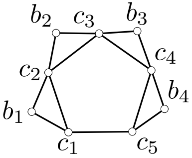
- We call v as the **neck** of the head.
- The head has only one bipartizing matching, as in Figure (b).
- Note that if a graph G contains a head as subgraph, then every bipartizing matching of G cannot include any other incident edge to v.

The Odd-pool

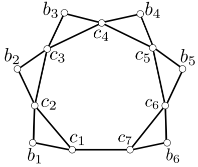
- Remember the k-pool graph G by removing a border, for $k \geq 3$ and odd.

The Odd-pool

- Remember the k-pool graph G by removing a border, for $k \geq 3$ and odd.



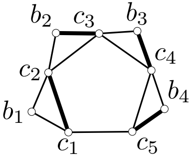
(a) 5-pool.



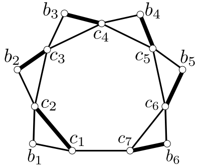
(b) 7-pool.

The Odd-pool

- Remember the k-pool graph G by removing a border, for $k \geq 3$ and odd.



(a) 5-pool.

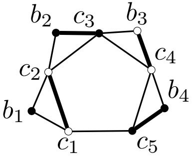


(b) 7-pool.

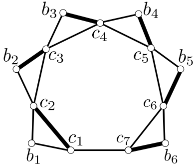
- We can see that every bipartizing matching M contains exactly one edge of the internal cycle.
  - Except that one with no border.

The Odd-pool

- Remember the k-pool graph G by removing a border, for $k \geq 3$ and odd.



(a) 5-pool.

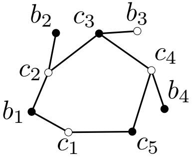


(b) 7-pool.

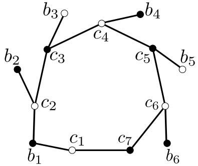
- We can see that every bipartizing matching M contains exactly one edge of the internal cycle.
  - Except that one with no border.
- Moreover, if either $c_1c_2 \in M$ or $c_2c_3 \in M$, then:

The Odd-pool

- Remember the k-pool graph G by removing a border, for $k \geq 3$ and odd.



(a) 5-pool.



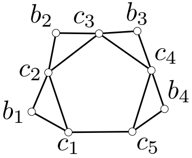
(b) 7-pool.

- We can see that every bipartizing matching M contains exactly one edge of the internal cycle.
  - Except that one with no border.

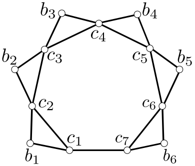
- Moreover, if either $c_1c_2 \in M$ or $c_2c_3 \in M$, then:
  - b_1 and b_2 are in the same part of $G - M$.
  - b_i and b_{i+1} are in different parts of $G - M$, for $i \geq 3$ and odd.

The Odd-pool

- Remember the k-pool graph G by removing a border, for $k \geq 3$ and odd.



(a) 5-pool.



(b) 7-pool.

- We can see that every bipartizing matching M contains exactly one edge of the internal cycle.
  - Except that one with no border.

- Moreover, if either $c_1c_2 \in M$ or $c_2c_3 \in M$, then:
  - b_1 and b_2 are in the same part of $G - M$.
  - b_i and b_{i+1} are in different parts of $G - M$, for $i \geq 3$ and odd.

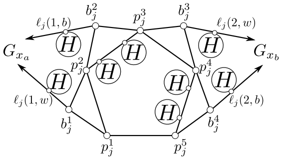
- We can generalize this for each pair c_i, c_{i+1} of edges of the internal cycle, for $i \geq 1$ and odd.

The Clause Gadgets

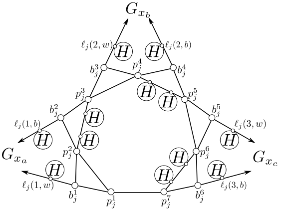
- Based on the previous observations and some more technical details, we obtain the clause gadgets of C_j in our reduction from a planar formula F.

The Clause Gadgets

- Based on the previous observations and some more technical details, we obtain the clause gadgets of C_j in our reduction from a planar formula F.
  - Each rounded H is an induced head graph connected by its neck vertex.



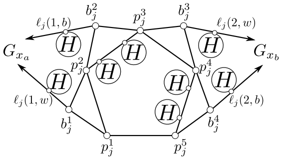
(a) For clauses of size two.



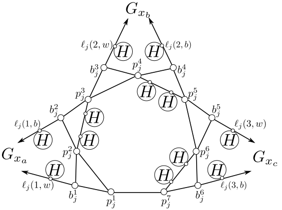
(b) For clauses of size three.

The Clause Gadgets

- Based on the previous observations and some more technical details, we obtain the clause gadgets of C_j in our reduction from a planar formula F.
  - Each rounded H is an induced head graph connected by its neck vertex.



(a) For clauses of size two.

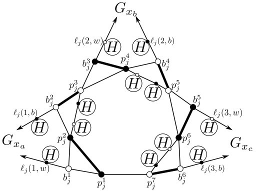


(b) For clauses of size three.

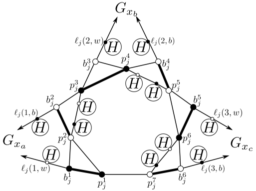
- We connect the pairs $\ell_j(i,b), \ell_j(i,w)$ to other two vertices in the variable gadgets, $i \in \{1,2,3\}$.

The Clause Gadgets

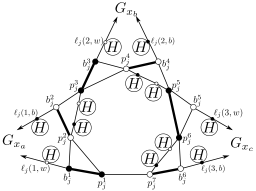
- Based on the previous observations and some more technical details, we obtain the clause gadgets of C_j in our reduction from a planar formula F.
  - Each rounded H is an induced head graph connected by its neck vertex.



(a) $p_j^1 p_j^2 \in M$.



(b) $p_j^3 p_j^4 \in M$.

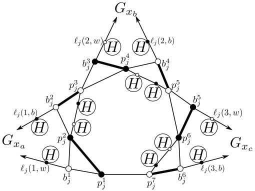


(c) $p_j^5 p_j^6 \in M$.

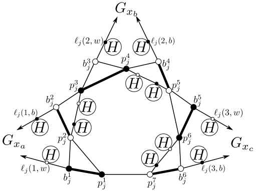
- We connect the pairs $\ell_j(i, b), \ell_j(i, w)$ to other two vertices in the variable gadgets, $i \in \{1, 2, 3\}$.
- We associate a literal of a clause as **true** if and only if both $\ell_j(i, b), \ell_j(i, w)$ are in the same part of $G - M$.

The Clause Gadgets

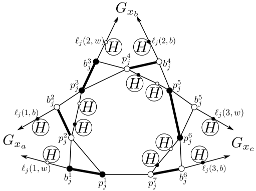
- Based on the previous observations and some more technical details, we obtain the clause gadgets of C_j in our reduction from a planar formula F.
  - Each rounded H is an induced head graph connected by its neck vertex.



(a) $p_j^1 p_j^2 \in M$.



(b) $p_j^3 p_j^4 \in M$.



(c) $p_j^5 p_j^6 \in M$.

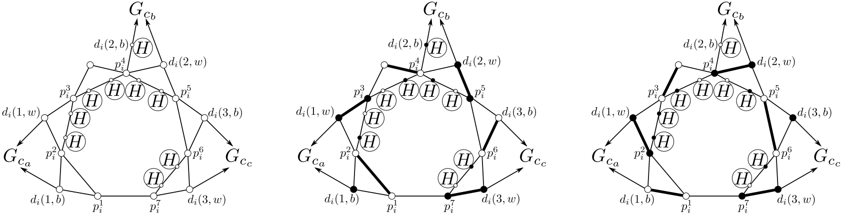
- We connect the pairs $\ell_j(i,b), \ell_j(i,w)$ to other two vertices in the variable gadgets, $i \in \{1,2,3\}$.
- We associate a literal of a clause as **true** if and only if both $\ell_j(i,b), \ell_j(i,w)$ are in the same part of $G - M$.
  - Hence, each clause gadget has only one true literal.

The Variable Gadget

- Similarly to the clause gadgets, we obtain our variable gadget of X_i.

The Variable Gadget

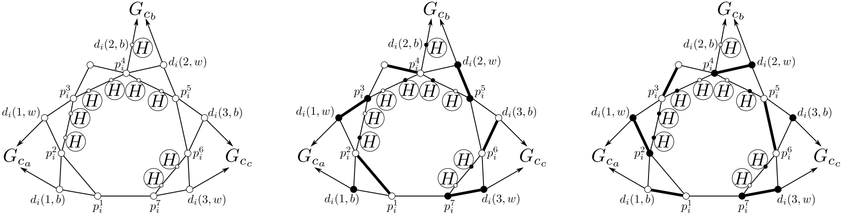
- Similarly to the clause gadgets, we obtain our variable gadget of X_i.



(a) Variable gadget. (b) $p_j^1 p_j^2 \in M$. (c) $p_j^5 p_j^6 \in M$.

The Variable Gadget

- Similarly to the clause gadgets, we obtain our variable gadget of X_i.

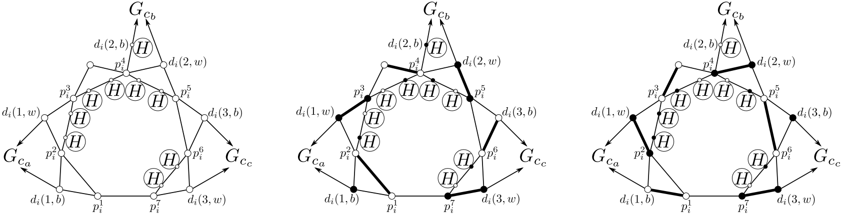


(a) Variable gadget. (b) $p_j^1 p_j^2 \in M$. (c) $p_j^5 p_j^6 \in M$.

- We connect the pairs $d_j(i,b), d_j(i,w)$ to the vertices $\ell_j(i,b), \ell_j(i,w)$ in the clause gadgets, $i \in \{1,2,3\}$.

The Variable Gadget

- Similarly to the clause gadgets, we obtain our variable gadget of X_i.

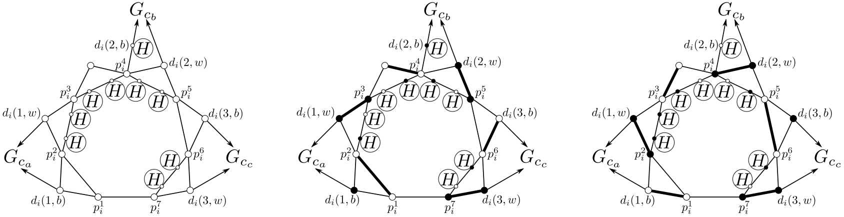


(a) Variable gadget. (b) $p_j^1 p_j^2 \in M$. (c) $p_j^5 p_j^6 \in M$.

- We connect the pairs $d_j(i,b), d_j(i,w)$ to the vertices $\ell_j(i,b), \ell_j(i,w)$ in the clause gadgets, $i \in \{1,2,3\}$.
- The pair $d_i(3,b)$ and $d_i(3,w)$ has opposite assignment to the other two corresponding pairs.

The Variable Gadget

- Similarly to the clause gadgets, we obtain our variable gadget of X_i.

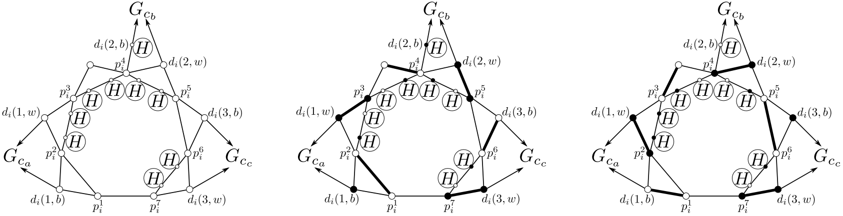


(a) Variable gadget. (b) $p_j^1 p_j^2 \in M$. (c) $p_j^5 p_j^6 \in M$.

- We connect the pairs $d_j(i,b), d_j(i,w)$ to the vertices $\ell_j(i,b), \ell_j(i,w)$ in the clause gadgets, $i \in \{1,2,3\}$.
- The pair $d_i(3,b)$ and $d_i(3,w)$ has opposite assignment to the other two corresponding pairs.
  - Hence, $d_i(3,b)$ and $d_i(3,w)$ represent $\overline{x_i}$ while the other pairs represent x_i.

The Variable Gadget

- Similarly to the clause gadgets, we obtain our variable gadget of X_i.

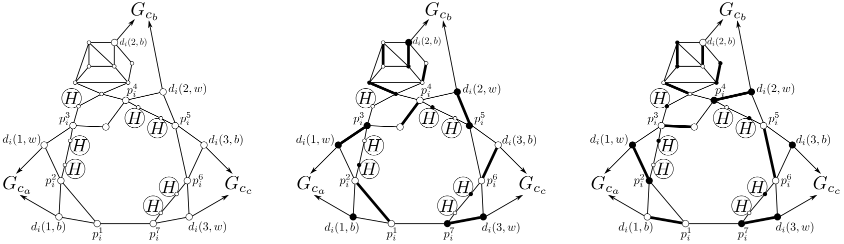


(a) Variable gadget. (b) $p_j^1 p_j^2 \in M$. (c) $p_j^5 p_j^6 \in M$.

- We connect the pairs $d_j(i,b), d_j(i,w)$ to the vertices $\ell_j(i,b), \ell_j(i,w)$ in the clause gadgets, $i \in \{1,2,3\}$.
- The pair $d_i(3,b)$ and $d_i(3,w)$ has opposite assignment to the other two corresponding pairs.
  - Hence, $d_i(3,b)$ and $d_i(3,w)$ represent $\overline{x_i}$ while the other pairs represent x_i.
- Note that p_i^4 is the only vertex of degree 5.

The Variable Gadget

- Similarly to the clause gadgets, we obtain our variable gadget of X_i.



(a) New variable gadget. (b) $p_j^1 p_j^2 \in M$. (c) $p_j^5 p_j^6 \in M$.

- We connect the pairs $d_j(i,b), d_j(i,w)$ to the vertices $\ell_j(i,b), \ell_j(i,w)$ in the clause gadgets, $i \in \{1,2,3\}$.
- The pair $d_i(3,b)$ and $d_i(3,w)$ has opposite assignment to the other two corresponding pairs.
  - Hence, $d_i(3,b)$ and $d_i(3,w)$ represent $\overline{x_i}$ while the other pairs represent x_i.
- Note that p_i^4 is the only vertex of degree 5.
  - Hence, we slightly modify the variable gadget in order to obtain a graph of maximum degree 4.

Other Results

- We have obtained other polynomial time results, such for:

Other Results

- We have obtained other polynomial time results, such for:
  - Graphs of bounded dominating set;

Other Results

- We have obtained other polynomial time results, such for:
  - Graphs of bounded dominating set;

    P_5-free graphs;

Other Results

- We have obtained other polynomial time results, such for:
    - Graphs of bounded dominating set;

      P_5-free graphs;

      graphs in which every odd-cycle subgraph is a triangle.

Other Results

- We have obtained other polynomial time results, such for:
  - Graphs of bounded dominating set;

    P_5-free graphs;

    graphs in which every odd-cycle subgraph is a triangle.
- We also considered parameterized complexity aspects.

Other Results

- We have obtained other polynomial time results, such for:
  - Graphs of bounded dominating set;

    P_5-free graphs;

    graphs in which every odd-cycle subgraph is a triangle.
- We also considered parameterized complexity aspects.
  - We show that BM is FPT when parameterized by the clique-width, presenting a Monadic Second Order Logic (MSOL 1) formulation.

Other Results

- We have obtained other polynomial time results, such for:
  - Graphs of bounded dominating set;

    P_5-free graphs;

    graphs in which every odd-cycle subgraph is a triangle.
- We also considered parameterized complexity aspects.
  - We show that BM is FPT when parameterized by the clique-width, presenting a Monadic Second Order Logic (MSOL 1) formulation.
  - As a corollary, we prove that there exists polynomial-time algorithms for several graph classes.

Other Results

- We have obtained other polynomial time results, such for:
  - Graphs of bounded dominating set;

    P_5-free graphs;

    graphs in which every odd-cycle subgraph is a triangle.
- We also considered parameterized complexity aspects.
  - We show that BM is FPT when parameterized by the clique-width, presenting a Monadic Second Order Logic (MSOL 1) formulation.
  - As a corollary, we prove that there exists polynomial-time algorithms for several graph classes.
  - In particular, it is polynomial-time solvable for chordal graphs.

Thank You!