On the Structural Parameterizations of Locating-Dominating Set and Test Cover

Dipayan Chakraborty[a,b], Florent Foucaud[a], Diptapriyo Majumdar[c] and Prafullkumar Tale[d]

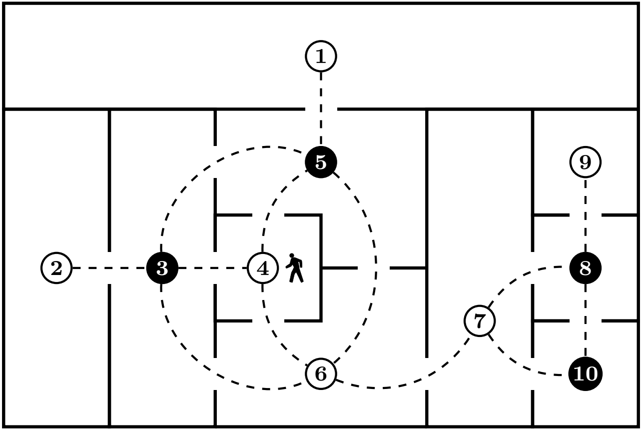
[a]Université Clermont Auvergne, CNRS, Mines Saint-Étienne, Clermont Auvergne INP, LIMOS, 63000 Clermont-Ferrand, France

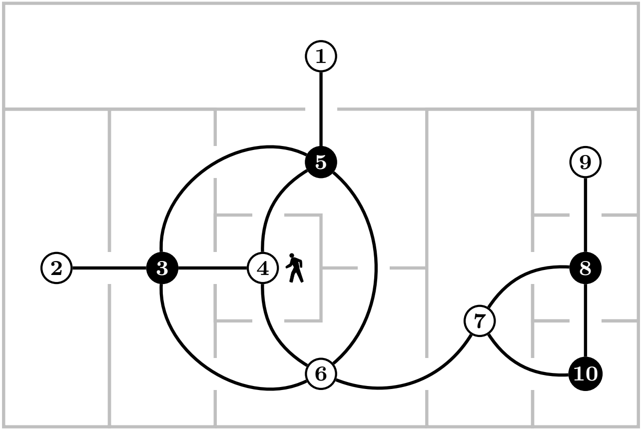
[b]Department of Mathematics and Applied Mathematics, University of Johannesburg, Auckland Park, 2006, South Africa

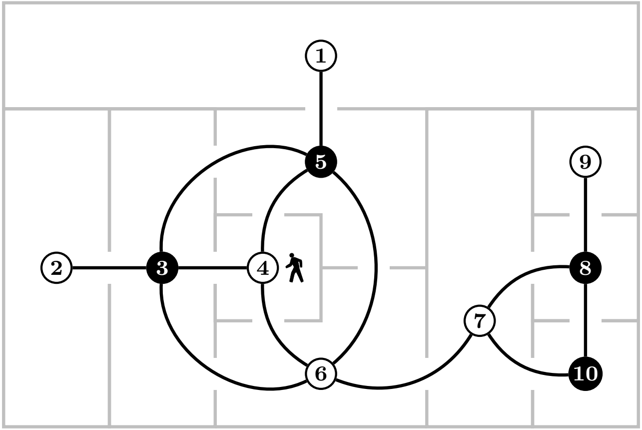
[c]Indraprastha Institute of Information Technology Delhi, Delhi, India

[d]Indian Institute of Science Education and Research Pune, Pune, India

Introduction: Locating dominating sets in graphs


1
5
9
2
3
4
8
7
6
10


1
2
3
4
5
6
7
8
9
10

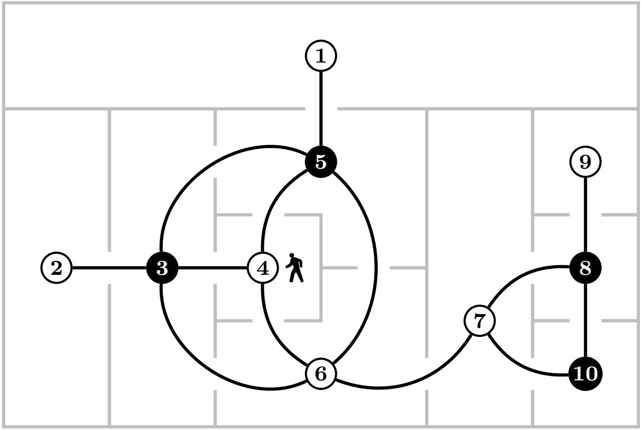


Graph $G = (V, E)$

Open neighborhood:
$N(v) = \{u : uv \in E\}$
$N(7) = \{6, 8, 10\}$

Closed neighborhood:
$N[v] = N(v) \cup \{v\}$
$N[7] = \{6, 8, 10, 7\}$

C: set of black vertices



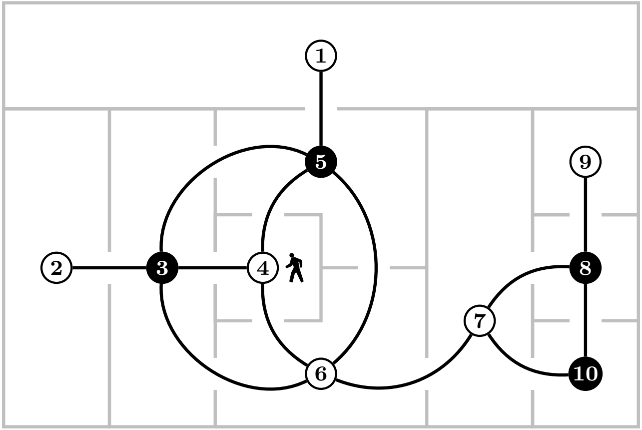
Graph $G = (V, E)$

Open neighborhood:
$N(v) = \{u : uv \in E\}$
$N(7) = \{6, 8, 10\}$

Closed neighborhood:
$N[v] = N(v) \cup \{v\}$
$N[7] = \{6, 8, 10, 7\}$

C: set of black vertices

(1) A detector can monitor upto distance 1



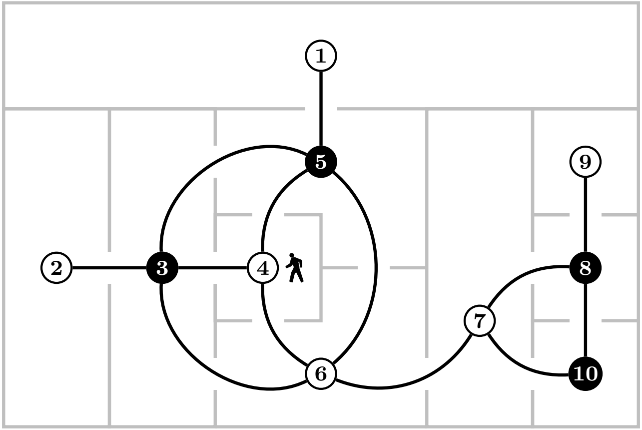
Graph $G = (V, E)$

<u>Open neighborhood:</u>
$N(v) = \{u : uv \in E\}$
$N(7) = \{6, 8, 10\}$

<u>Closed neighborhood:</u>
$N[v] = N(v) \cup \{v\}$
$N[7] = \{6, 8, 10, 7\}$

C: set of black vertices

Dominating set: A set $C \subseteq V$ such that $N[v] \cap C \neq \emptyset$ for all $v \in V$



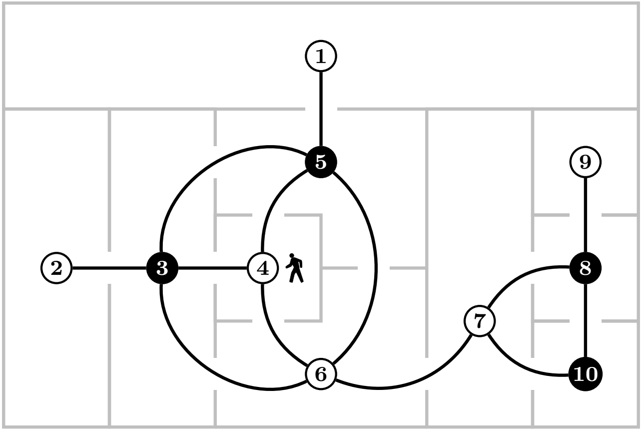
Graph $G = (V, E)$

Open neighborhood:
$N(v) = \{u : uv \in E\}$
$N(7) = \{6, 8, 10\}$

Closed neighborhood:
$N[v] = N(v) \cup \{v\}$
$N[7] = \{6, 8, 10, 7\}$

C: set of black vertices

Dominating set: A set $C \subseteq V$ such that $N[v] \cap C \neq \emptyset$ for all $v \in V$



Graph $G = (V, E)$

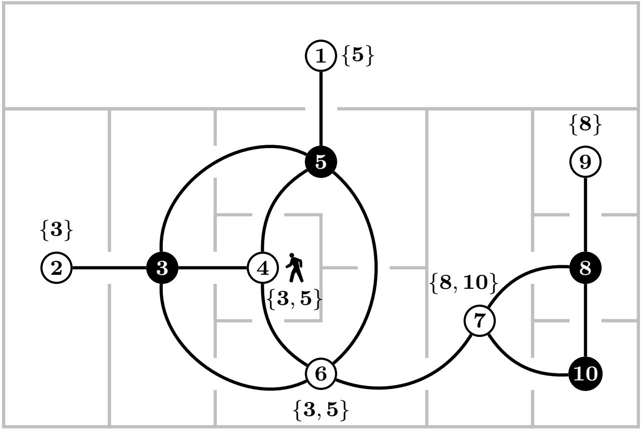
Open neighborhood:
$N(v) = \{u : uv \in E\}$
$N(7) = \{6, 8, 10\}$

Closed neighborhood:
$N[v] = N(v) \cup \{v\}$
$N[7] = \{6, 8, 10, 7\}$

C: set of black vertices

Dominating set: A set $C \subseteq V$ such that $N[v] \cap C \neq \emptyset$ for all $v \in V$

(2) A detector can distinguish between itself and a neighbor



Graph $G = (V, E)$

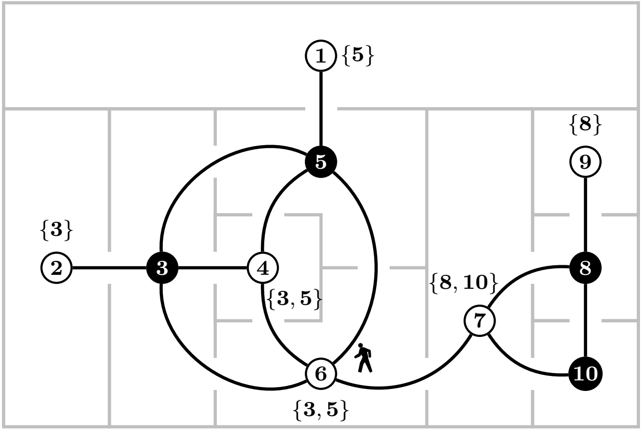
Open neighborhood:
$N(v) = \{u : uv \in E\}$
$N(7) = \{6, 8, 10\}$

Closed neighborhood:
$N[v] = N(v) \cup \{v\}$
$N[7] = \{6, 8, 10, 7\}$

C: set of black vertices

Dominating set: A set $C \subseteq V$ such that $N[v] \cap C \neq \emptyset$ for all $v \in V$

(2) A detector can distinguish between itself and a neighbor



Graph $G = (V, E)$

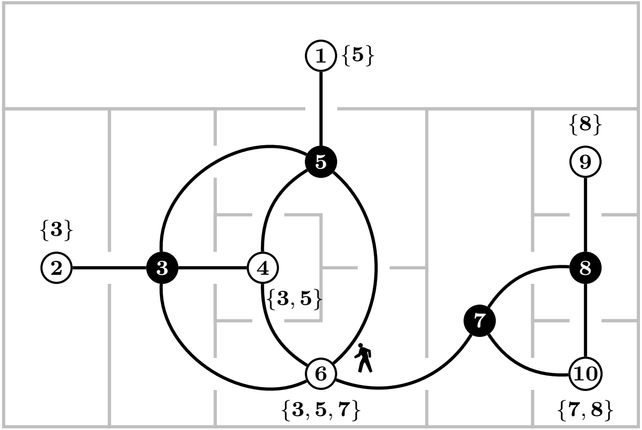
Open neighborhood:
$N(v) = \{u : uv \in E\}$
$N(7) = \{6, 8, 10\}$

Closed neighborhood:
$N[v] = N(v) \cup \{v\}$
$N[7] = \{6, 8, 10, 7\}$

C: set of black vertices

Dominating set: A set $C \subseteq V$ such that $N[v] \cap C \neq \emptyset$ for all $v \in V$

(2) A detector can distinguish between itself and a neighbor



Graph $G = (V, E)$

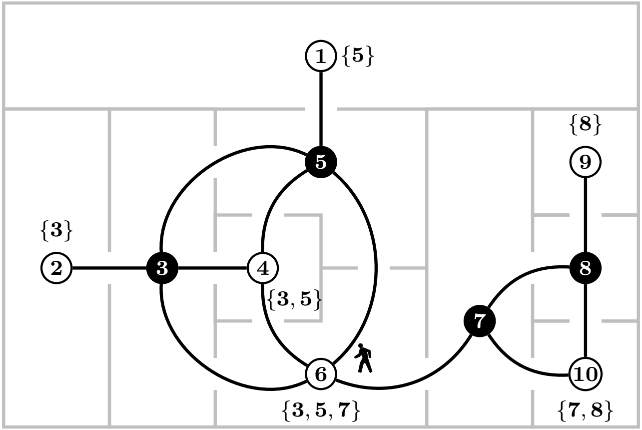
Open neighborhood:
$N(v) = \{u : uv \in E\}$
$N(7) = \{6, 8, 10\}$

Closed neighborhood:
$N[v] = N(v) \cup \{v\}$
$N[7] = \{6, 8, 10, 7\}$

C: set of black vertices

Dominating set: A set $C \subseteq V$ such that $N[v] \cap C \neq \emptyset$ for all $v \in V$

(2) A detector can distinguish between itself and a neighbor



Graph $G = (V, E)$

Open neighborhood:
$N(v) = \{u : uv \in E\}$
$N(7) = \{6, 8, 10\}$

Closed neighborhood:
$N[v] = N(v) \cup \{v\}$
$N[7] = \{6, 8, 10, 7\}$

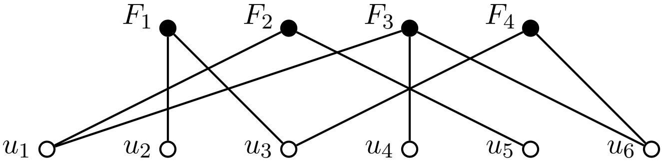
C: set of black vertices

Dominating set: A set $C \subseteq V$ such that $N[v] \cap C \neq \emptyset$ for all $v \in V$

Locating set: A set $C \subseteq V$ such that $N(u) \cap C \neq N(v) \cap C$ for all $u, v \in V \setminus C$

Definition

Locating dominating (LD) set: is a set $C \subseteq V$ of a graph $G = (V, E)$ such that C is
(i) a dominating set of G; and
(ii) $N(u) \cap C \neq N(v) \cap C$ for all $u, v \in V \setminus C$.



Definition

Test Cover: Given a set U (of *items*) and a set $\mathcal{F}$ of subsets (called *tests*) of U, the set $\mathcal{F}$ is called a **test cover** if given any pair of distinct $u, v \in U$, there exists a set $F \in \mathcal{F}$ such that either $u \in F$, $v \notin F$ or $v \in F$, $u \notin F$.

Decision version of finding the minimum LD-set in a graph:

Minimum LD-Set
Input: (G, k): A graph G and a positive integer k.
Question: Does there exist an LD-set C of G such that $|C| \leq k$?

Minimum LD-Set is NP-hard!

Decision version of finding the minimum LD-set in a graph:

MINIMUM LD-SET **Input:** (G, k): A graph G and a positive integer k. **Question:** Does there exist an LD-set C of G such that $\|C\| \leq k$?

MINIMUM LD-SET is NP-hard!

What about *Fixed Parameter Tractable* (FPT) algorithms?
i.e. given a graph parameter k, can we find an algorithm to find a minimum code in time $f(k) \cdot n^{O(1)}$? e.g. $f(k) = 2^k, 2^{k^2}, 2^{2^k} \dots$

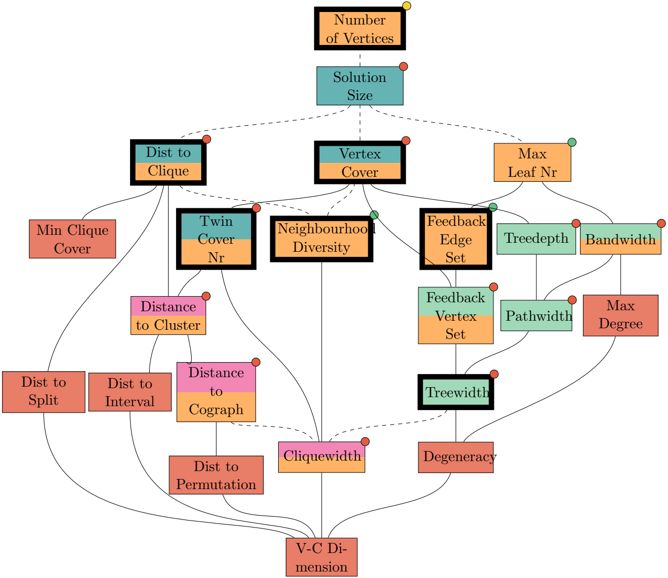
Decision version of finding the minimum LD-set in a graph:

MINIMUM LD-SET **Input:** (G, k): A graph G and a positive integer k. **Question:** Does there exist an LD-set C of G such that $\vert C\vert \leq k$?

MINIMUM LD-SET is NP-hard!

What about *Fixed Parameter Tractable* (FPT) algorithms?
i.e. given a graph parameter k, can we find an algorithm to find a minimum code in time $f(k) \cdot n^{O(1)}$? e.g. $f(k) = 2^k, 2^{k^2}, 2^{2^k} \dots$

Note: MINIMUM LD-SET is FPT when parameterized by solution size k.
Reason: $|V(G)| = O(2^k)$. Thus brute force gives $2^{O(k^2)} \cdot n^{O(1)}$ runtime.



single-exp FPT slightly super-exp FPT double-exp FPT FPT para-NP-h.

linear kernel (tight) quadratic kernel no polynomial kernel

Theorem (C., Foucaud, Majumdar & Tale, 2024)

Minimum LD-Set *admits an algorithm running in time* $2^{O(\mathsf{vc}\log\mathsf{vc})} \cdot n^{O(1)}$, *where* vc *is the vertex cover number of the input graph.*

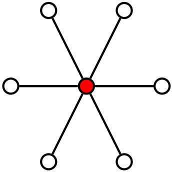
Theorem (C., Foucaud, Majumdar & Tale, 2024)

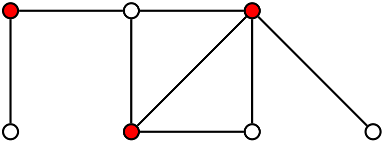
Minimum LD-Set *admits a kernel with* $\mathcal{O}(\mathsf{fes})$ *vertices and edges, where* fes *is the feedback edge set number of the input graph.*

Theorem (C., Foucaud, Majumdar & Tale, 2024)

Minimum LD-Set *does not admit a polynomial compression of size* $\mathcal{O}(n^{2-\epsilon})$ *for any* $\epsilon > 0$, *unless* $\mathsf{NP} \subseteq \mathsf{coNP}/poly$, *where* n *denotes the number of vertices of the input graph.*

Vertex cover: A set $S \subset V$ such that $V \setminus S$ is an independent set.
Vertex cover number: $\mathsf{vc} = \min\{|S| : S$ is a vertex cover of $G\}$





Theorem (C., Foucaud, Majumdar & Tale, 2024)

LD-Code *admits an algorithm running in time* $2^{O(\mathsf{vc} \log \mathsf{vc})} \cdot n^{O(1)}$.

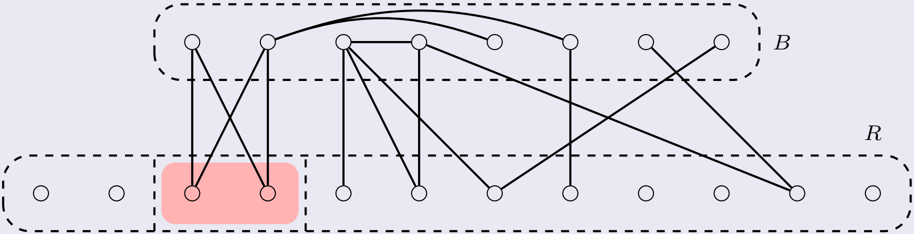
Algorithm (by dynamic programming):

- Find a minimum vertex cover in time $1.2528^{\mathsf{vc}} \cdot n^{O(1)}$ [Harris & Narayanaswamy, STACS 2024].
- Build optimum solution by the **dynamic programming**:

$$\mathtt{opt}[i, \mathcal{P}, S] = \min \begin{cases} \mathtt{opt}[i-1, \mathcal{P}, S], \\ 1 + \min\limits_{\substack{\mathcal{P}' \Cap \mathcal{P}(r_i) = \mathcal{P}, \\ S' \cup N(r_i) = S}} \mathtt{opt}[i-1, \mathcal{P}', S']. \end{cases}$$

$$\boxed{\begin{aligned} &\mathtt{opt}[i, \mathcal{P}, S] = \min |C|, \\ &C \subset \{r_1, r_2, \ldots, r_i\}, \\ &C \rightsquigarrow (\mathcal{P}, S) \end{aligned}}$$

- Algorithm **brute forces** all partitions of vertex cover.



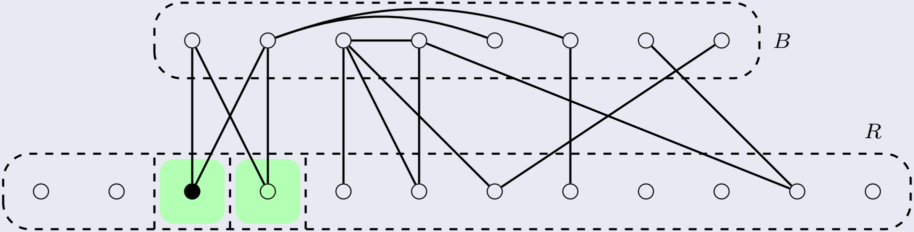
Algorithm (by dynamic programming):

- Find a minimum vertex cover in time $1.2528^{\mathsf{vc}} \cdot n^{O(1)}$ [Harris & Narayanaswamy, STACS 2024].
- Build optimum solution by the **dynamic programming**:

$$\mathtt{opt}[i, \mathcal{P}, S] = \min \begin{cases} \mathtt{opt}[i-1, \mathcal{P}, S], \\ 1 + \min\limits_{\substack{\mathcal{P}' \Cap \mathcal{P}(r_i) = \mathcal{P}, \\ S' \cup N(r_i) = S}} \mathtt{opt}[i-1, \mathcal{P}', S']. \end{cases}$$

$$\boxed{\begin{array}{l} \mathtt{opt}[i, \mathcal{P}, S] = \min |C|, \\ C \subset \{r_1, r_2, \ldots, r_i\}, \\ C \rightsquigarrow (\mathcal{P}, S) \end{array}}$$

- Algorithm **brute forces** all partitions of vertex cover.



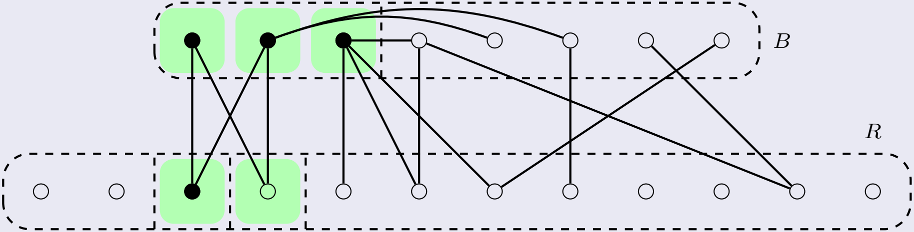
Algorithm (by dynamic programming):

- Find a minimum vertex cover in time $1.2528^{\mathsf{vc}} \cdot n^{O(1)}$ [Harris & Narayanaswamy, STACS 2024].
- Build optimum solution by the **dynamic programming**:

$$\mathtt{opt}[i, \mathcal{P}, S] = \min \begin{cases} \mathtt{opt}[i-1, \mathcal{P}, S], \\ 1 + \min\limits_{\substack{\mathcal{P}' \Cap \mathcal{P}(r_i) = \mathcal{P}, \\ S' \cup N(r_i) = S}} \mathtt{opt}[i-1, \mathcal{P}', S']. \end{cases}$$

$$\boxed{\begin{aligned} &\mathtt{opt}[i, \mathcal{P}, S] = \min |C|, \\ &C \subset \{r_1, r_2, \ldots, r_i\}, \\ &C \rightsquigarrow (\mathcal{P}, S) \end{aligned}}$$

- Algorithm **brute forces** all partitions of vertex cover.



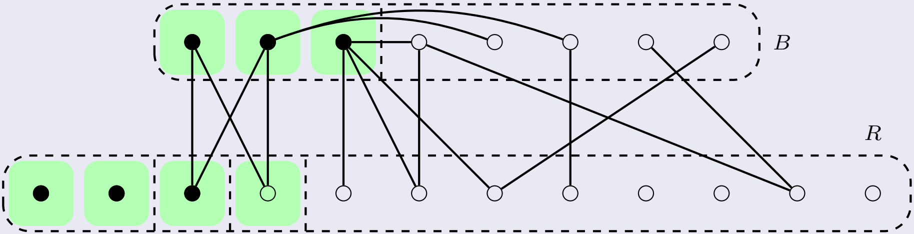
Algorithm (by dynamic programming):

- Find a minimum vertex cover in time $1.2528^{\mathsf{vc}} \cdot n^{O(1)}$ [Harris & Narayanaswamy, STACS 2024].
- Build optimum solution by the **dynamic programming**:

$$\texttt{opt}[i, \mathcal{P}, S] = \min \begin{cases} \texttt{opt}[i-1, \mathcal{P}, S], \\ 1 + \min\limits_{\substack{\mathcal{P}' \Cap \mathcal{P}(r_i) = \mathcal{P}, \\ S' \cup N(r_i) = S}} \texttt{opt}[i-1, \mathcal{P}', S']. \end{cases}$$

$$\boxed{\begin{array}{l} \texttt{opt}[i, \mathcal{P}, S] = \min |C|, \\ C \subset \{r_1, r_2, \ldots, r_i\}, \\ C \rightsquigarrow (\mathcal{P}, S) \end{array}}$$

- Algorithm **brute forces** all partitions of vertex cover.



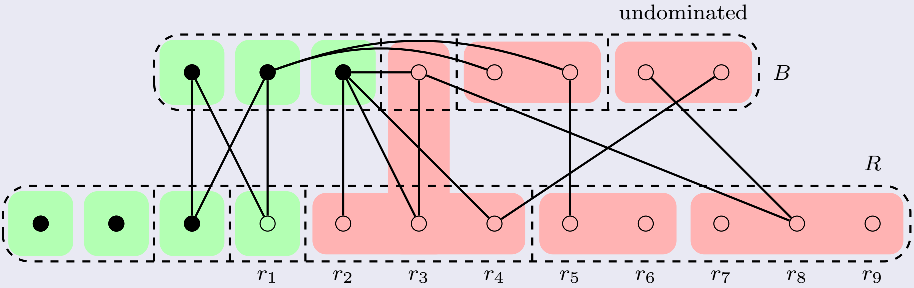
Algorithm (by dynamic programming):

- Find a minimum vertex cover in time $1.2528^{\mathsf{vc}} \cdot n^{O(1)}$ [Harris & Narayanaswamy, STACS 2024].
- Build optimum solution by the **dynamic programming**:

$$\mathtt{opt}[i, \mathcal{P}, S] = \min \begin{cases} \mathtt{opt}[i-1, \mathcal{P}, S], \\ 1 + \min\limits_{\substack{\mathcal{P}' \Cap \mathcal{P}(r_i) = \mathcal{P}, \\ S' \cup N(r_i) = S}} \mathtt{opt}[i-1, \mathcal{P}', S']. \end{cases}$$

$$\boxed{\begin{array}{l} \mathtt{opt}[i, \mathcal{P}, S] = \min |C|, \\ C \subset \{r_1, r_2, \ldots, r_i\}, \\ C \rightsquigarrow (\mathcal{P}, S) \end{array}}$$

- Algorithm **brute forces** all partitions of vertex cover.



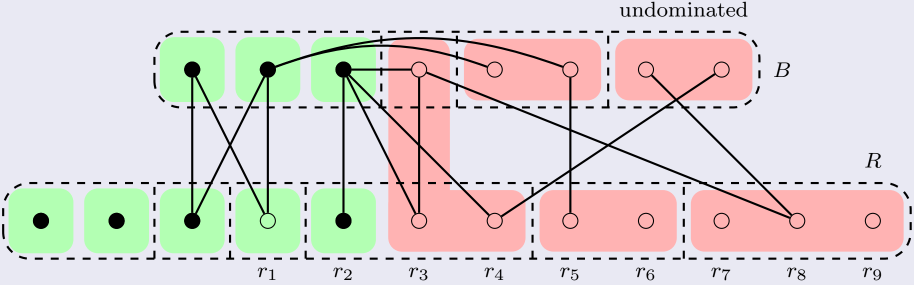
Algorithm (by dynamic programming):

- Find a minimum vertex cover in time $1.2528^{\mathsf{vc}} \cdot n^{O(1)}$ [Harris & Narayanaswamy, STACS 2024].
- Build optimum solution by the **dynamic programming**:

$$\mathtt{opt}[i, \mathcal{P}, S] = \min \begin{cases} \mathtt{opt}[i-1, \mathcal{P}, S], \\ 1 + \min\limits_{\substack{\mathcal{P}' \Cap \mathcal{P}(r_i) = \mathcal{P}, \\ S' \cup N(r_i) = S}} \mathtt{opt}[i-1, \mathcal{P}', S']. \end{cases}$$

$$\boxed{\begin{aligned} &\mathtt{opt}[i, \mathcal{P}, S] = \min |C|, \\ &C \subset \{r_1, r_2, \ldots, r_i\}, \\ &C \rightsquigarrow (\mathcal{P}, S) \end{aligned}}$$

- Algorithm **brute forces** all partitions of vertex cover.



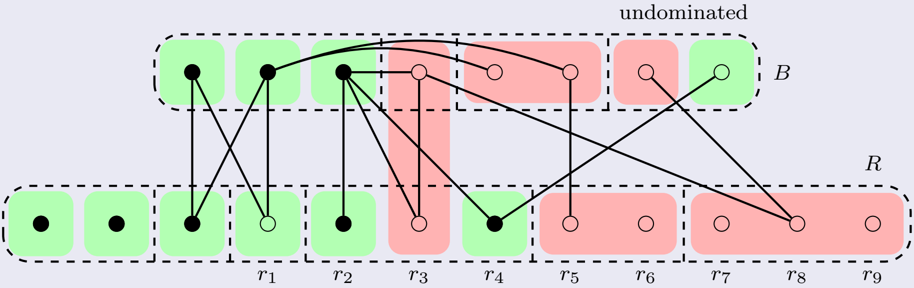
Algorithm (by dynamic programming):

- Find a minimum vertex cover in time $1.2528^{\mathsf{vc}} \cdot n^{O(1)}$ [Harris & Narayanaswamy, STACS 2024].
- Build optimum solution by the **dynamic programming**:

$$\texttt{opt}[i, \mathcal{P}, S] = \min \begin{cases} \texttt{opt}[i-1, \mathcal{P}, S], \\ 1 + \min\limits_{\substack{\mathcal{P}' \Cap \mathcal{P}(r_i) = \mathcal{P}, \\ S' \cup N(r_i) = S}} \texttt{opt}[i-1, \mathcal{P}', S']. \end{cases}$$

$$\boxed{\begin{array}{l} \texttt{opt}[i, \mathcal{P}, S] = \min |C|, \\ C \subset \{r_1, r_2, \ldots, r_i\}, \\ C \rightsquigarrow (\mathcal{P}, S) \end{array}}$$

- Algorithm **brute forces** all partitions of vertex cover.



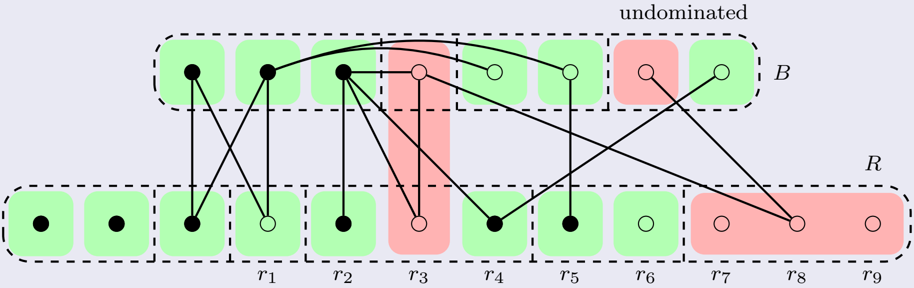
Algorithm (by dynamic programming):

- Find a minimum vertex cover in time $1.2528^{\mathsf{vc}} \cdot n^{O(1)}$ [Harris & Narayanaswamy, STACS 2024].
- Build optimum solution by the **dynamic programming**:

$$\texttt{opt}[i, \mathcal{P}, S] = \min \begin{cases} \texttt{opt}[i-1, \mathcal{P}, S], \\ 1 + \min\limits_{\substack{\mathcal{P}' \Cap \mathcal{P}(r_i) = \mathcal{P}, \\ S' \cup N(r_i) = S}} \texttt{opt}[i-1, \mathcal{P}', S']. \end{cases}$$

$$\boxed{\begin{array}{l} \texttt{opt}[i, \mathcal{P}, S] = \min |C|, \\ C \subset \{r_1, r_2, \ldots, r_i\}, \\ C \rightsquigarrow (\mathcal{P}, S) \end{array}}$$

- Algorithm **brute forces** all partitions of vertex cover.



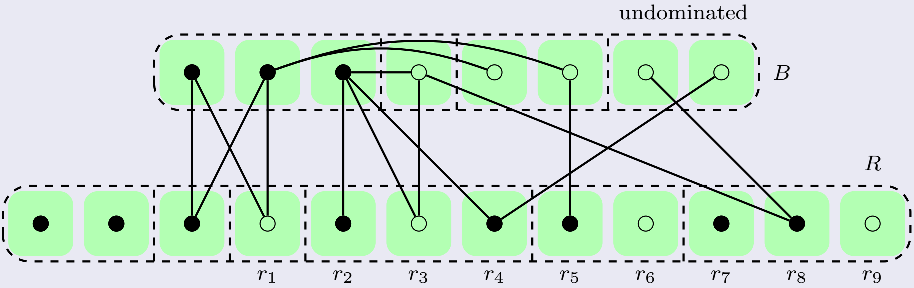
Algorithm (by dynamic programming):

- Find a minimum vertex cover in time $1.2528^{\mathsf{vc}} \cdot n^{O(1)}$ [Harris & Narayanaswamy, STACS 2024].
- Build optimum solution by the **dynamic programming**:

$$\mathtt{opt}[i, \mathcal{P}, S] = \min \begin{cases} \mathtt{opt}[i-1, \mathcal{P}, S], \\ 1 + \min\limits_{\substack{\mathcal{P}' \Cap \mathcal{P}(r_i) = \mathcal{P}, \\ S' \cup N(r_i) = S}} \mathtt{opt}[i-1, \mathcal{P}', S']. \end{cases}$$

$$\boxed{\begin{aligned} &\mathtt{opt}[i, \mathcal{P}, S] = \min |C|, \\ &C \subset \{r_1, r_2, \ldots, r_i\}, \\ &C \rightsquigarrow (\mathcal{P}, S) \end{aligned}}$$

- Algorithm **brute forces** all partitions of vertex cover.



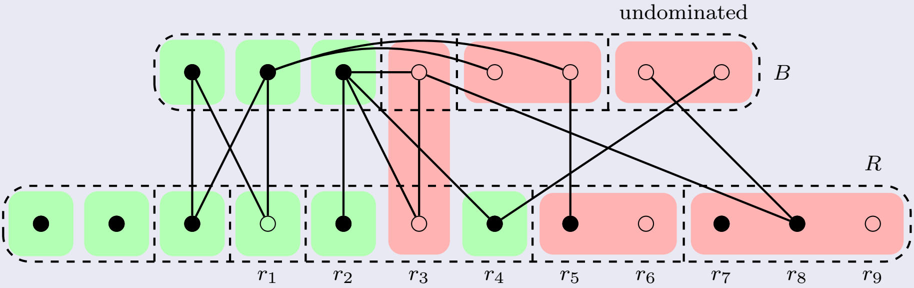
Algorithm (by dynamic programming):

- Find a minimum vertex cover in time $1.2528^{\mathsf{vc}} \cdot n^{O(1)}$ [Harris & Narayanaswamy, STACS 2024].
- Build optimum solution by the **dynamic programming**:

$$\texttt{opt}[i, \mathcal{P}, S] = \min \begin{cases} \texttt{opt}[i-1, \mathcal{P}, S], \\ 1 + \min\limits_{\substack{\mathcal{P}' \Cap \mathcal{P}(r_i) = \mathcal{P}, \\ S' \cup N(r_i) = S}} \texttt{opt}[i-1, \mathcal{P}', S']. \end{cases}$$

Running time:
$\mathcal{P}: \; 2^{\mathsf{vc} \log \mathsf{vc}} \cdot |R|$
$S: \; 2^{\mathsf{vc}}$

- Algorithm **brute forces** all partitions of vertex cover.



Thank You!