Reconfiguration of k-Path Vertex Covers

Hoang Anh Duc

December 05, 2019

hoanganhduc@ces.kyutech.ac.jp

Research Seminar at Kyutech Algorithms Group

Outline

1 Introduction
- k-Path Vertex Cover Reconfiguration
- Summary of Achieved Results

2 Reconfiguring k-Path Vertex Covers on Trees under TJ

3 Open Problems

Introduction

Reconfiguration Problems

- **Given:**
  - Two configurations A, B.
    E.g., two states of Rubik's Cube Puzzle.
  - A reconfiguration rule (defining whether two arbitrary configurations are adjacent).
    E.g., rotating one face of the cube 90, 180, or 270 degrees.
- **Question:** Does there exist a sequence of adjacent configurations that transforms A into B?

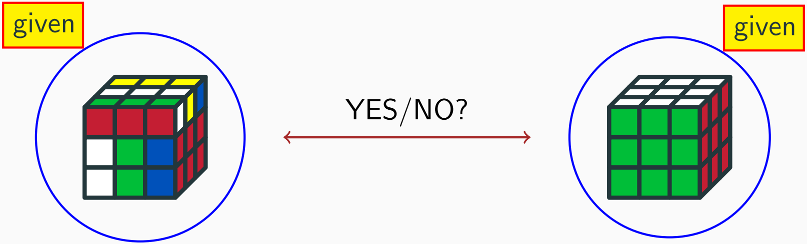


Figure 1: An instance of Rubik's Cube Puzzle.

k-Path Vertex Cover (k-PVC)

I is a k-path vertex cover ($k \geq 2$) of a graph G if each path on k vertices in G contains at least one vertex from I. If $k = 2$, the set I is a vertex cover.

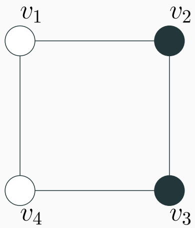


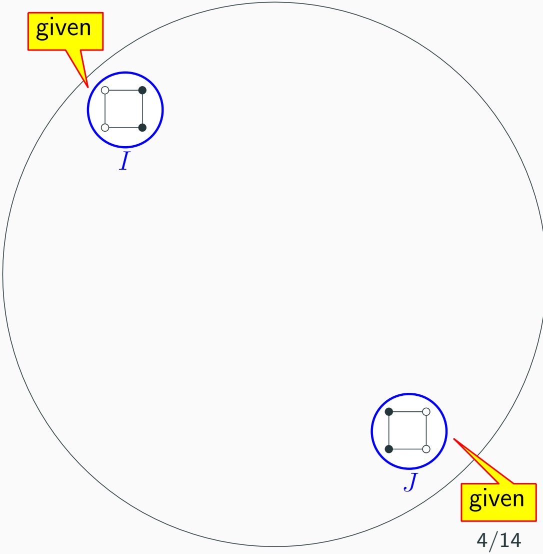
Figure 2: $I = \{v_2, v_3\}$ is a 3-path vertex cover but not a vertex cover.

Theorem (Brešar et al. 2011)

It is NP*-complete to decide if there is a k-PVC ($k \geq 2$) of size at most s, where s is a given positive integer.*

Imagine that each vertex of a k-PVC contains a token

Two k-PVCs I and J are given



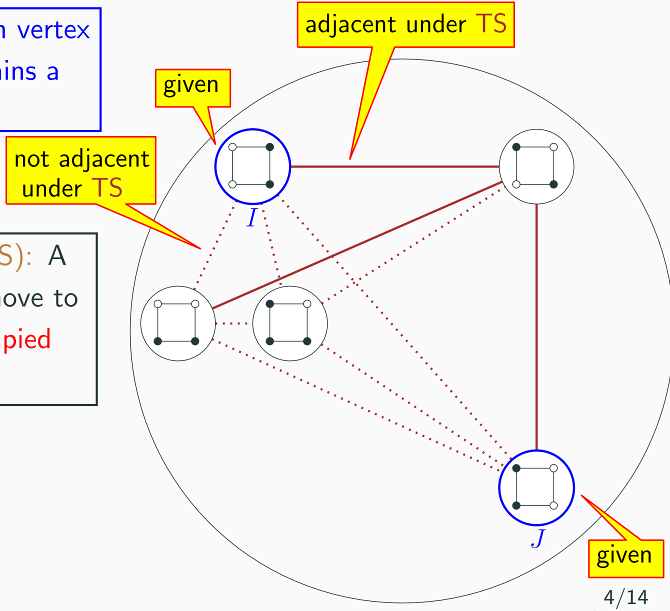
Example: 3-path vertex covers in C_4

k-Path Vertex Cover Reconfiguration under TS

Imagine that each vertex of a k-PVC contains a token

Two k-PVCs I and J are given

Token Sliding (TS): A token can only move to one of its unoccupied neighbors



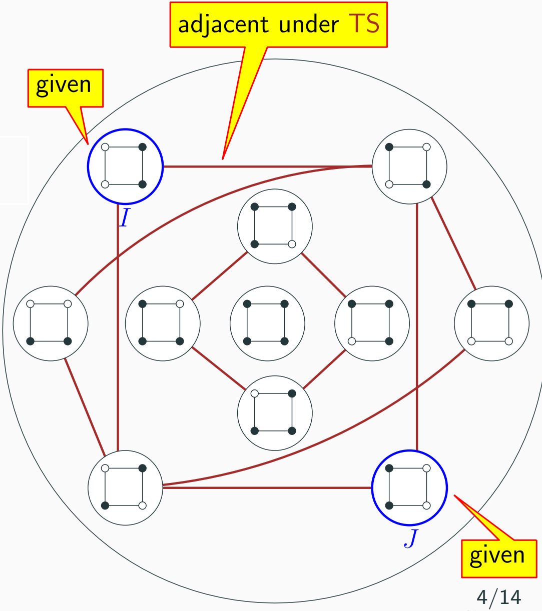
Example: 3-path vertex covers in C_4

Imagine that each vertex of a k-PVC contains a token

Two k-PVCs I and J are given

Token Sliding (TS): A token can only move to one of its unoccupied neighbors

Question: Is there a sequence of adjacent k-PVCs under TS that transforms I into J?



Example: 3-path vertex covers in C_4

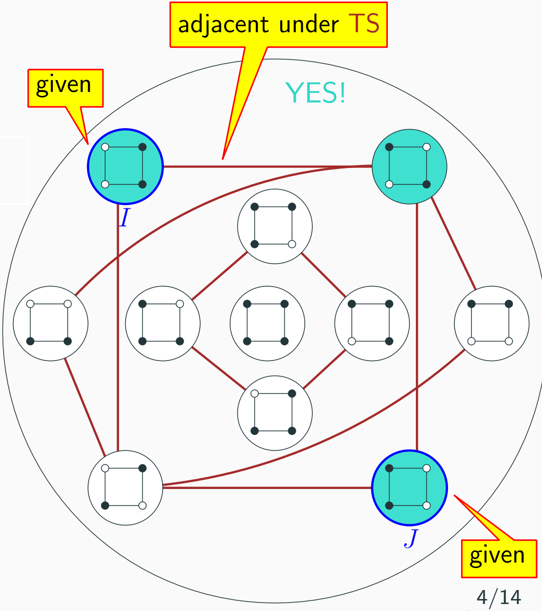
k-Path Vertex Cover Reconfiguration under TS

Imagine that each vertex of a k-PVC contains a token

Two k-PVCs I and J are given

Token Sliding (TS): A token can only move to one of its unoccupied neighbors

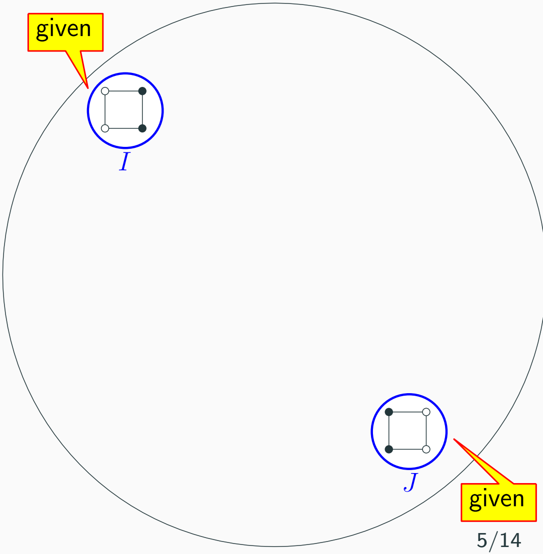
Question: Is there a sequence of adjacent k-PVCs under TS that transforms I into J?



Example: 3-path vertex covers in C_4

Imagine that each vertex of a k-PVC contains a token

Two k-PVCs I and J are given

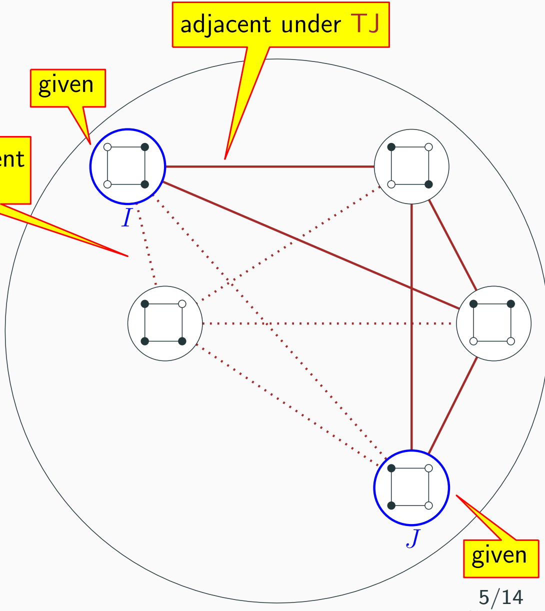


Example: 3-path vertex covers in C_4

Imagine that each vertex of a k-PVC contains a token

Two k-PVCs I and J are given

Token Jumping (TJ): A token can move to any unoccupied vertex (not necessarily its neighbor)



Example: 3-path vertex covers in C_4

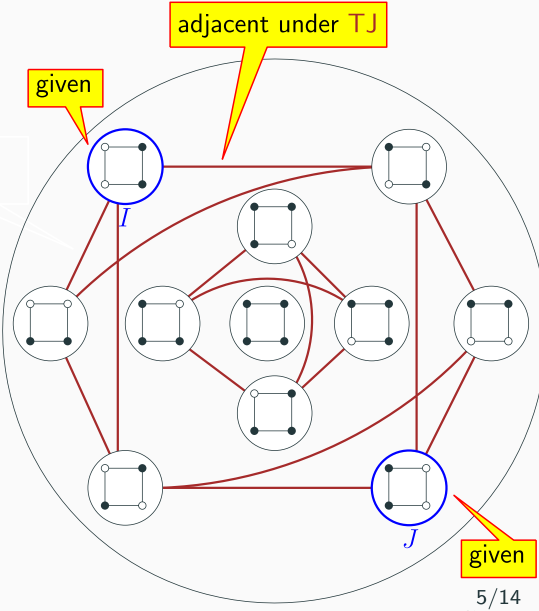
k-Path Vertex Cover Reconfiguration under TJ

Imagine that each vertex of a k-PVC contains a token

Two k-PVCs I and J are given

Token Jumping (TJ): A token can move to any unoccupied vertex (not necessarily its neighbor)

Question: Is there a sequence of adjacent k-PVCs under TJ that transforms I into J?



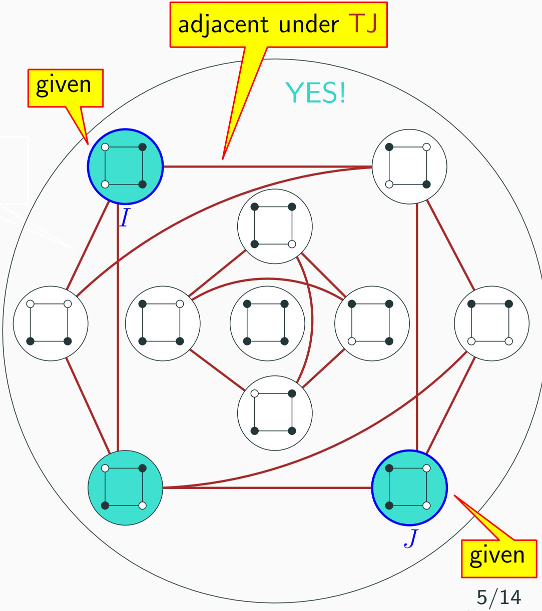
Example: 3-path vertex covers in C_4

Imagine that each vertex of a k-PVC contains a token

Two k-PVCs I and J are given

Token Jumping (TJ): A token can move to any unoccupied vertex (not necessarily its neighbor)

Question: Is there a sequence of adjacent k-PVCs under TJ that transforms I into J?

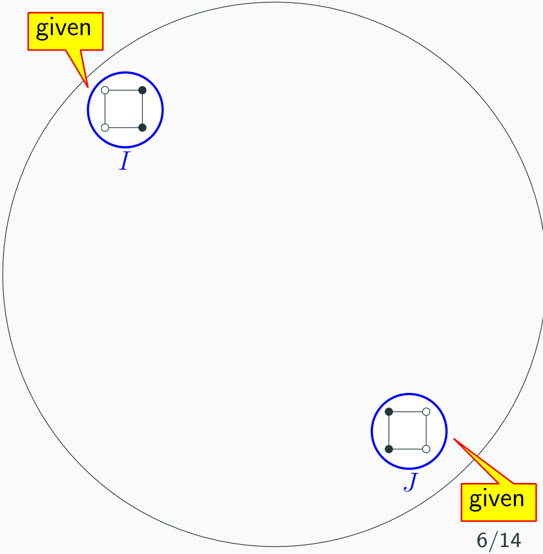


Example: 3-path vertex covers in C_4

k-**Path Vertex Cover Reconfiguration under** TAR(u)

Imagine that each vertex of a k-PVC contains a token

Two k-PVCs I and J are given



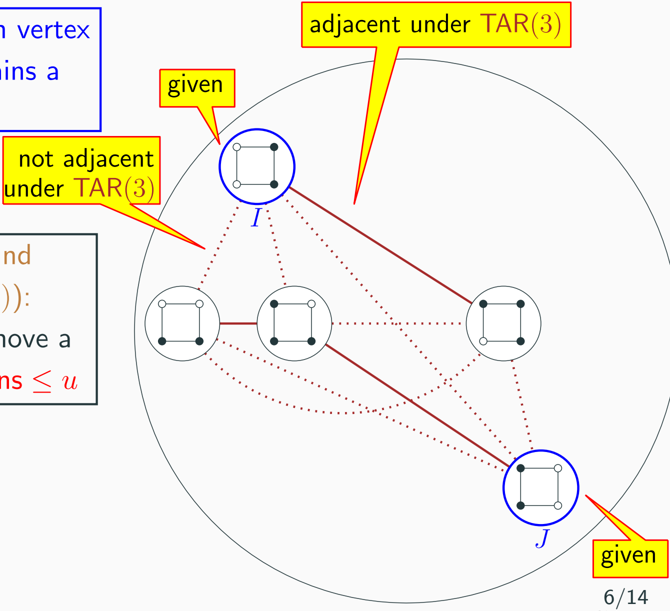
Example: 3-path vertex covers in C_4

k-Path Vertex Cover Reconfiguration under TAR(u)

Imagine that each vertex of a k-PVC contains a token

Two k-PVCs I and J are given

Token Addition and Removal (TAR(u)):
Either add or remove a token s.t. #tokens $\leq u$



Example: 3-path vertex covers in C_4

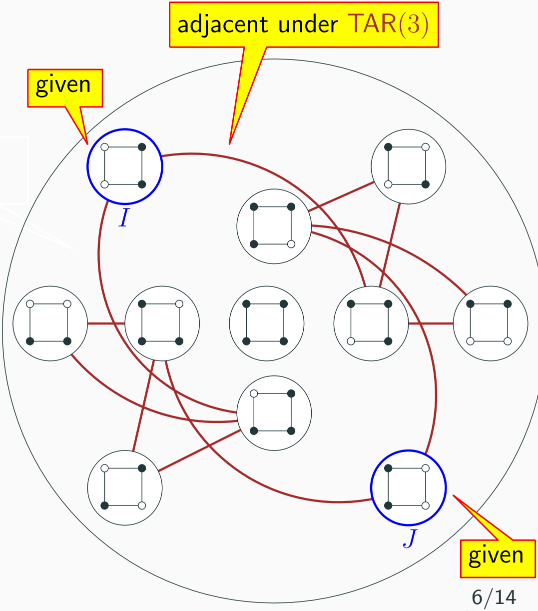
k-Path Vertex Cover Reconfiguration under TAR(u)

Imagine that each vertex of a k-PVC contains a token

Two k-PVCs I and J are given

Token Addition and Removal (TAR(u)): Either add or remove a token s.t. #tokens $\leq u$

Question: Is there a sequence of adjacent k-PVCs under TAR(u) that transforms I into J?



Example: 3-path vertex covers in C_4

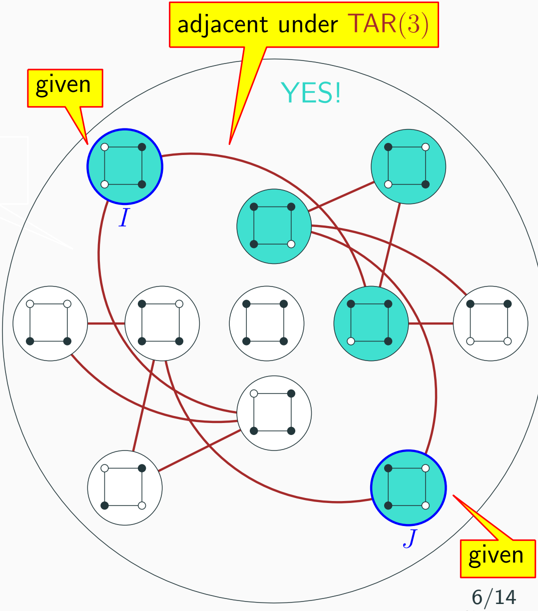
k-Path Vertex Cover Reconfiguration under TAR(u)

Imagine that each vertex of a k-PVC contains a token

Two k-PVCs I and J are given

Token Addition and Removal (TAR(u)): Either add or remove a token s.t. #tokens $\leq u$

Question: Is there a sequence of adjacent k-PVCs under TAR(u) that transforms I into J?



Example: 3-path vertex covers in C_4

k-Path Vertex Cover Reconfiguration under TJ/TAR(u)

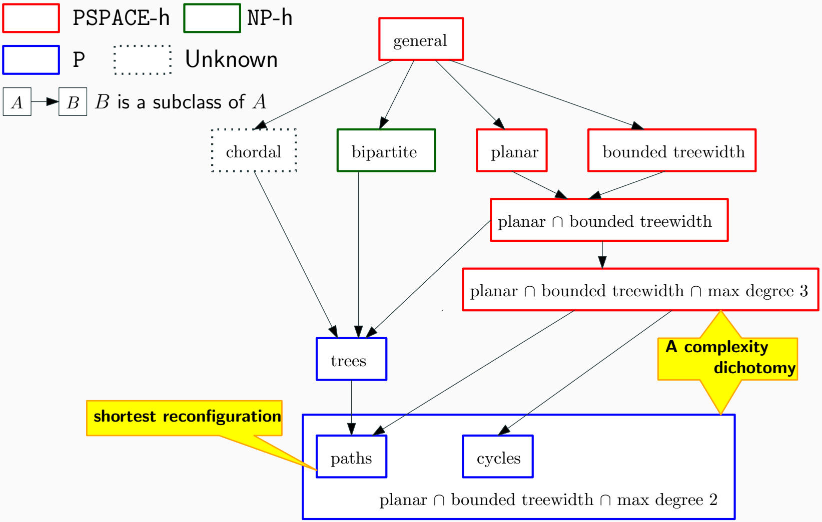


Figure 3: Our Results (joint work with **A. Suzuki** and **T. Yagita**). To appear in WALCOM 2020. `arXiv:1911.03026`

k-Path Vertex Cover Reconfiguration under TS

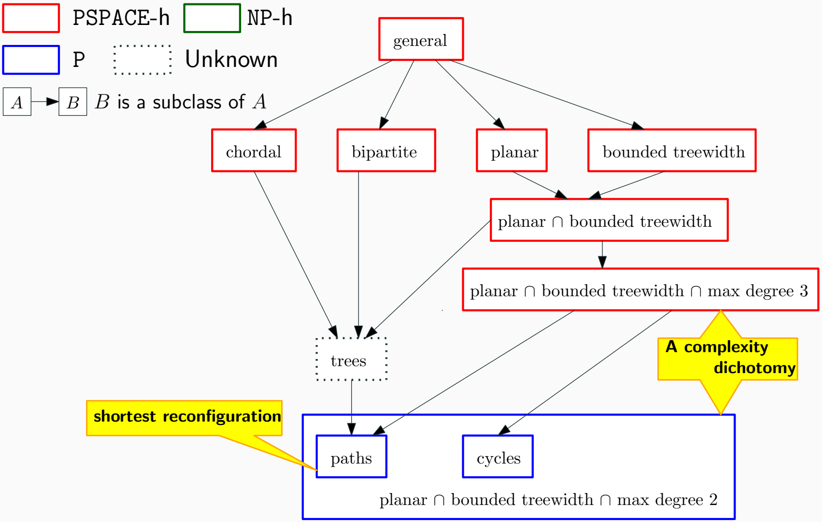
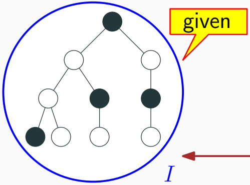


Figure 4: Our Results (joint work with **A. Suzuki** and **T. Yagita**). To appear in WALCOM 2020. `arXiv:1911.03026`

Reconfiguring k-Path Vertex Covers on Trees under TJ

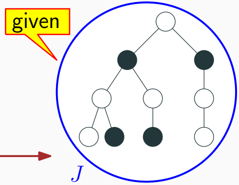
Example: $k = 3$



Theorem (Hoang, Suzuki, and Yagita 2019)

(T, I, J, TJ) *is a yes-instance* $\Leftrightarrow |I| = |J| = s$

YES

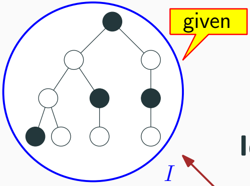


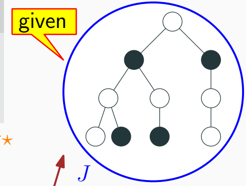
Example: $k = 3$

Theorem (Hoang, Suzuki, and Yagita 2019)

(T, I, J, TJ) *is a yes-instance* $\Leftrightarrow |I| = |J| = s$

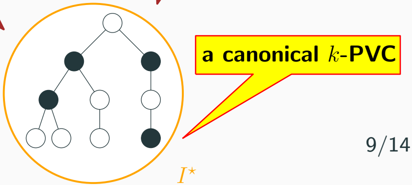
Idea: Find a canonical k-PVC $I^\star$



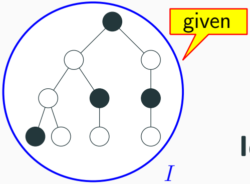


YES

YES

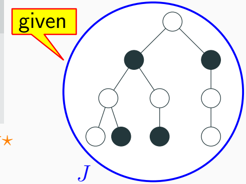


Example: $k = 3$



Theorem (Hoang, Suzuki, and Yagita 2019)

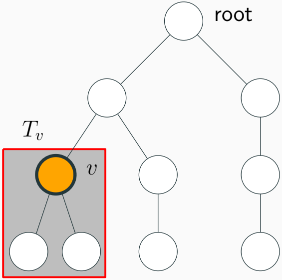
(T, I, J, TJ) *is a yes-instance*
$\Leftrightarrow |I| = |J| = s$



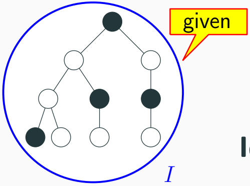
Idea: Find a canonical k-PVC $I^\star$

How to find $I^\star$?

- Partition T (assume it is rooted)
  - Find a subtree T_v: rooted at v and $T_v - v$ has no k-path
  - Place a token at v
  - Repeat with $T - T_v$

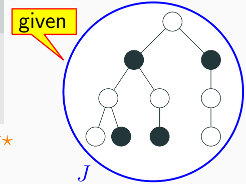


Example: $k = 3$



Theorem (Hoang, Suzuki, and Yagita 2019)

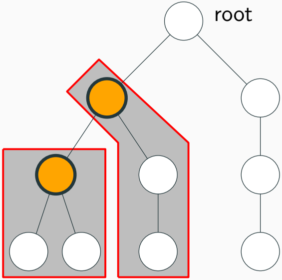
(T, I, J, TJ) *is a yes-instance* $\Leftrightarrow |I| = |J| = s$



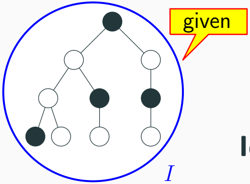
Idea: Find a canonical k-PVC $I^\star$

How to find $I^\star$?

- Partition T (assume it is rooted)
  - Find a subtree T_v: rooted at v and $T_v - v$ has no k-path
  - Place a token at v
  - Repeat with $T - T_v$

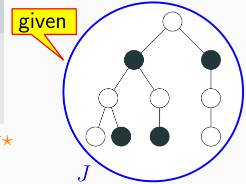


Example: $k = 3$



Theorem (Hoang, Suzuki, and Yagita 2019)

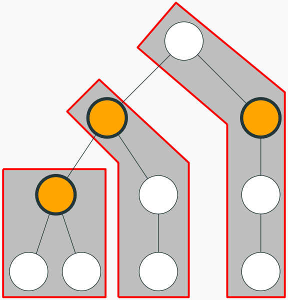
(T, I, J, TJ) *is a yes-instance*
$\Leftrightarrow |I| = |J| = s$



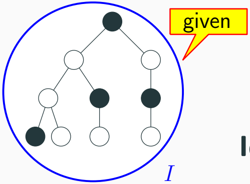
Idea: Find a canonical k-PVC $I^\star$

How to find $I^\star$?

- Partition T (assume it is rooted)
  - Find a subtree T_v: rooted at v and $T_v - v$ has no k-path
  - Place a token at v
  - Repeat with $T - T_v$

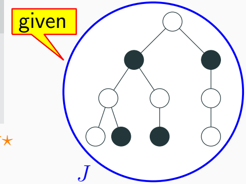


Example: $k = 3$



Theorem (Hoang, Suzuki, and Yagita 2019)

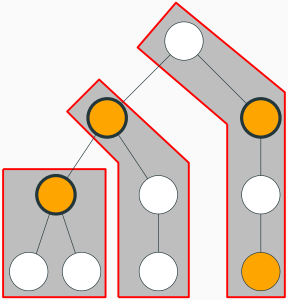
(T, I, J, TJ) *is a yes-instance* $\Leftrightarrow |I| = |J| = s$



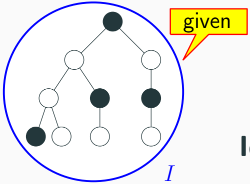
Idea: Find a canonical k-PVC $I^\star$

How to find $I^\star$?

- Partition T (assume it is rooted)
  - Find a subtree T_v: rooted at v and $T_v - v$ has no k-path
  - Place a token at v
  - Repeat with $T - T_v$
- Add token(s) arbitrarily until having s tokens

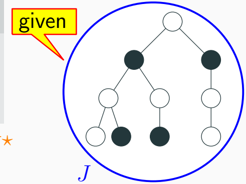


Example: $k = 3$



Theorem (Hoang, Suzuki, and Yagita 2019)

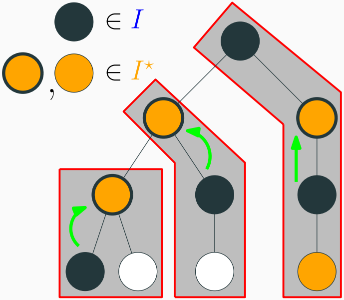
(T, I, J, TJ) *is a yes-instance* $\Leftrightarrow |I| = |J| = s$



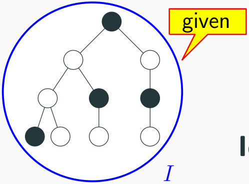
Idea: Find a canonical k-PVC $I^\star$

How to transform I into $I^\star$?

- Handle tokens ● $\in I^\star$
  - Each partition has exactly one token ● $\in I^\star$ and at least one token ● $\in I$
  - Jump a token ● $\in I$ inside the partition to that vertex

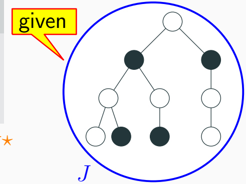


Example: $k = 3$



Theorem (Hoang, Suzuki, and Yagita 2019)

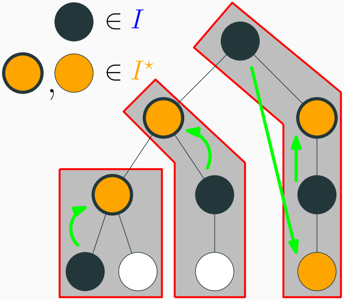
(T, I, J, TJ) *is a yes-instance* $\Leftrightarrow |I| = |J| = s$



Idea: Find a canonical k-PVC $I^\star$

How to transform I into $I^\star$?

- Handle tokens ● $\in I^\star$
  - Each partition has exactly one token ● $\in I^\star$ and at least one token ● $\in I$
  - Jump a token ● $\in I$ inside the partition to that vertex
- Handle tokens ● $\in I^\star$
  - Jump arbitrarily



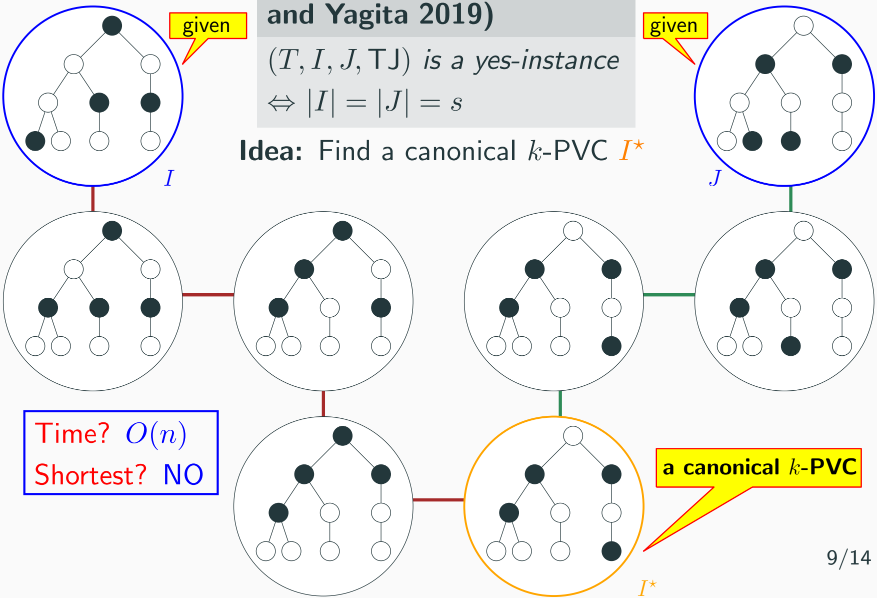
Reconfiguring k-Path Vertex Covers on Trees under TJ

Example: $k = 3$

Theorem (Hoang, Suzuki, and Yagita 2019)

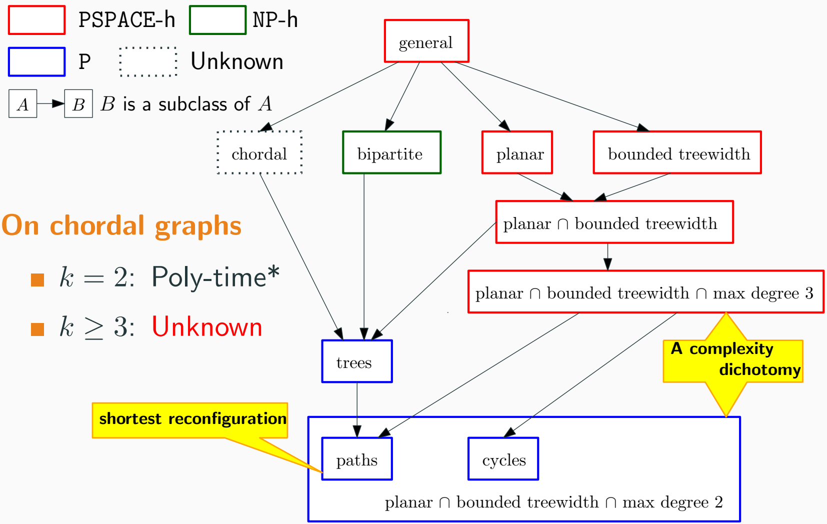
(T, I, J, TJ) *is a yes-instance*
$\Leftrightarrow |I| = |J| = s$

Idea: Find a canonical k-PVC $I^\star$



Time? $O(n)$
Shortest? NO

Open Problems

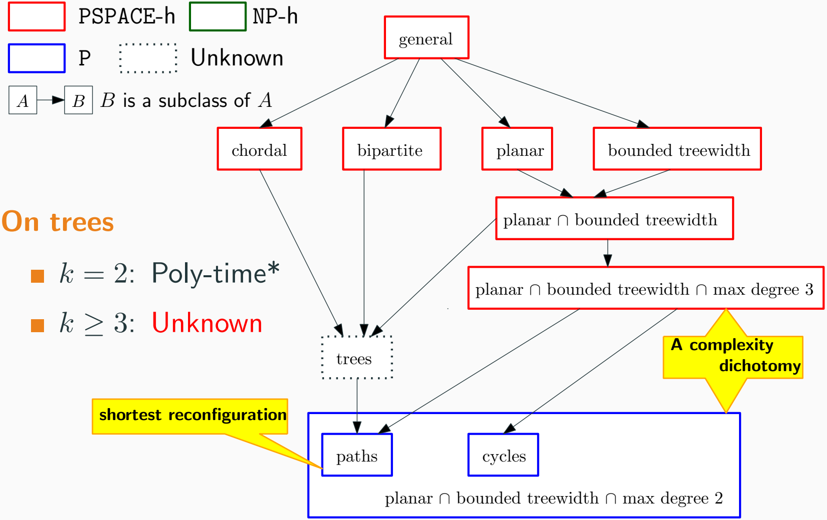


On chordal graphs

- $k = 2$: Poly-time*
- $k \geq 3$: Unknown

* Marcin Kamiński, Paul Medvedev, and Martin Milanič (2012). "Complexity of Independent Set Reconfigurability Problems". In: *Theoretical Computer Science* 439, pp. 9–15. DOI: 10.1016/j.tcs.2012.03.004

k-Path Vertex Cover Reconfiguration under TS



On trees

- $k = 2$: Poly-time*
- $k \geq 3$: Unknown

* Erik D. Demaine et al. (2015). "Linear-Time Algorithm for Sliding Tokens on Trees". In: *Theoretical Computer Science* 600, pp. 132–142. DOI: 10.1016/j.tcs.2015.07.037

- For $k = 2$, the problem is PSPACE-complete for bounded treewidth graphs under each of TS, TJ, or TAR [Wrochna 2018]. It is unknown whether it can be solved in polynomial time even for outerplanar graphs (whose treewidth ≤ 2).[1]

[1] This is an open problem proposed by Amer Mouawad (American University of Beirut, Lebanon) at CoRe 2019. See `https://pagesperso.g-scop.grenoble-inp.fr/~bousquen/CoRe_2019/CoRe_2019_Open_Problems.pdf` for a list of open reconfiguration problems proposed at this conference.

Bibliography i

Brešar, Boštjan, František Kardoš, Ján Katrenič, and Gabriel Semanišin (2011). "Minimum k-path vertex cover". In: *Discrete Applied Mathematics* 159.12, pp. 1189–1195. DOI: 10.1016/j.dam.2011.04.008.

Demaine, Erik D., Martin L. Demaine, Eli Fox-Epstein, Duc A. Hoang, Takehiro Ito, Hirotaka Ono, Yota Otachi, Ryuhei Uehara, and Takeshi Yamada (2015). "Linear-Time Algorithm for Sliding Tokens on Trees". In: *Theoretical Computer Science* 600, pp. 132–142. DOI: 10.1016/j.tcs.2015.07.037.

Hoang, Duc A., Akira Suzuki, and Tsuyoshi Yagita (2019). "Reconfiguring k-path vertex covers". In: *arXiv preprints*. arXiv: 1911.03026.

Kamiński, Marcin, Paul Medvedev, and Martin Milanič (2012). "Complexity of Independent Set Reconfigurability Problems". In: *Theoretical Computer Science* 439, pp. 9–15. DOI: 10.1016/j.tcs.2012.03.004.

Wrochna, Marcin (2018). "Reconfiguration in Bounded Bandwidth and Treedepth". In: *Journal of Computer and System Sciences* 93, pp. 1–10. DOI: 10.1016/j.jcss.2017.11.003.