Sequentially Swapping Tokens: Further on Graph Classes

Hironori Kiya[1], **Yuto Okada**[2], Hirotaka Ono[2], and Yota Otachi[2]

2023-01-17 SOFSEM 2023

[1]Kyushu University [2]Nagoya University

Background of the Study

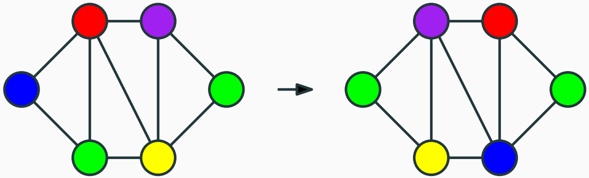
At first, I will show a video game.

The problem we study is inspired by the game.

(But the game does not appear after this.)

Abstract

We consider a puzzle on graphs.



Sequential Token Swapping (STS) is a decision problem:
Is there a “reconfiguration sequence” of length at most k?

This problem is NP-complete in general,
but may be tractable for some graph classes.

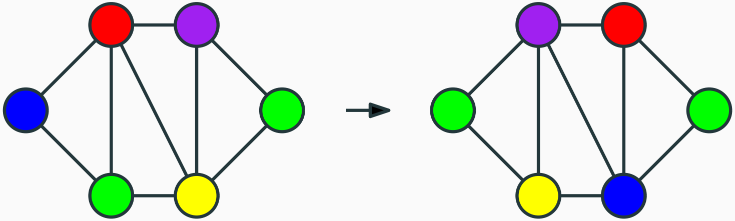
We study STS on some graph classes,
and show P / NP-complete cases.

The Model

The puzzle consists of:

- a graph $G = (V, E)$ with n vertices
- n tokens colored with c colors $\{1, 2, \dots, c\}$

Each vertex in G has a token.

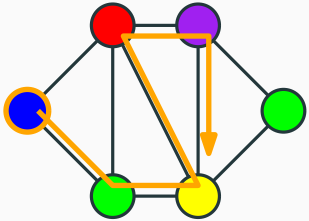


We define a configuration as a map $f : V \to \{1, 2, \dots, c\}$, that returns the color of the token.

The Transformation Rule

You take the following steps only once.

1. Pick a token. (We call this the moving token.)
2. Choose a walk starting from the moving token.
3. Move the moving token along the walk.



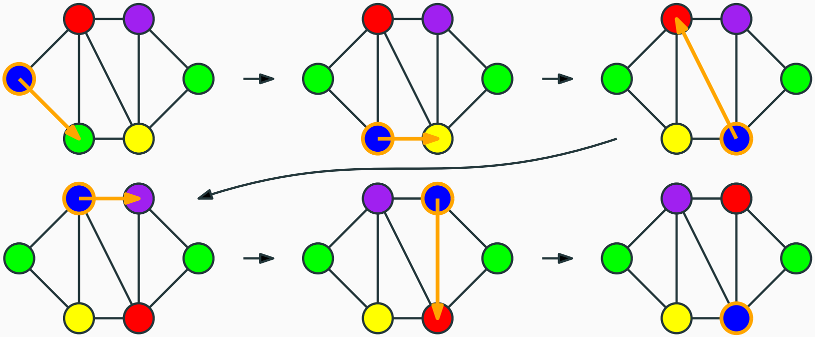
Every time you move the moving token, the configuration changes. (Next page)

The Transformation Rule

The configuration changes as follows:

Token Swapping

When the moving token passes the edge $\{u, v\}$,
tokens on u and v are swapped.



Formal Definition of the Problem

We consider the following problem:

Problem: STS

Input:

A graph G, configurations f, f', and an integer k.

Question:

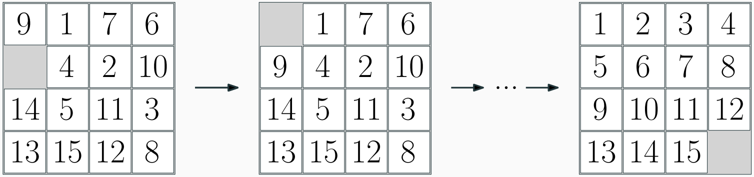
Is it possible to change the configuration f to f' by

1. Picking a token as the moving token,
2. choosing a walk (starting from the moving token) with length at most k,
3. moving the moving token along the walk?

Related Problem: 15 Puzzle

In fact, our puzzle is a variant of the 15 puzzle[1].

Why?

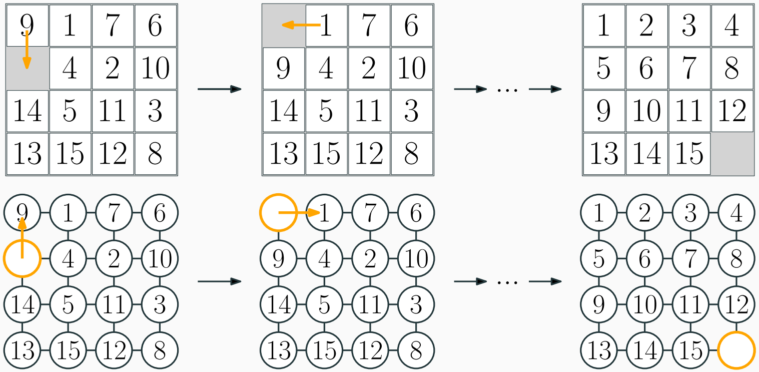


[1] The legal move is to slide an adjacent tile into the hole. Given a configuration, the goal is to make the right configuration.

The Same Behavior with 15 Puzzle

Consider the hole as the moving token.

Then both puzzles behave the same way.



The Difference from 15 Puzzle

There are two differences from the 15 puzzle:

1. Moving token (hole)

   In the 15 puzzle, the position of the hole is given.
   In our puzzle, you need to choose the moving token.

2. Same colors

   In the 15 puzzle, each token has a distinct color, $1, 2, \dots, 15$ and 16 (=hole).
   In our puzzle, we allow the tokens to have the same color.

Previous Work: Reachability

Problem: Reachability ver. of STS

Input: A graph G and configurations f, f'.

Question: Is it possible to change from f to f'?

(Under the same transformation rule)

In this problem, there is no limit to the number of moves.

Known Results

A full characterization is known [Trakultraipruk, 2013].

This can be checked in polynomial time.

Previous Work: Complexity

Yamanaka et al., 2019

STS is NP-complete in general.
But is polynomial-time solvable on some graph classes.

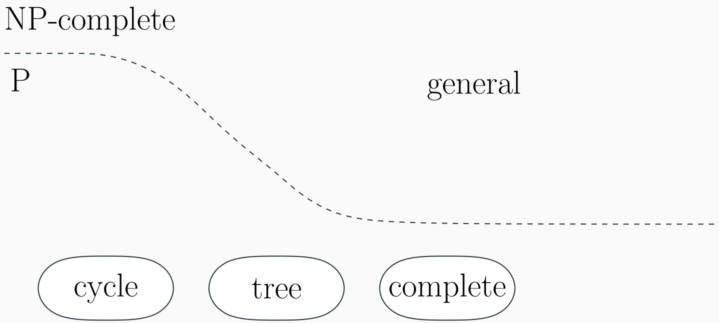


Figure 1: P/NP-complete cases on previous work

Previous Work: Complexity with 2 colors

As a natural question, we want to consider the case where the number of the colors is fixed.

But the following result means that STS remains intractable if we fix the number of colors.

Yamanaka et al., 2019

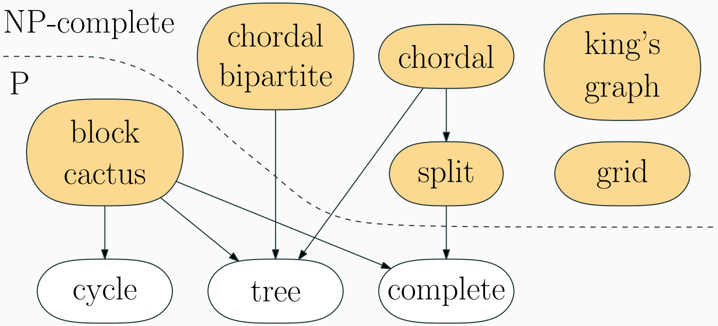
There exists a constant $\varepsilon > 0$ such that even if there are only 2 colors there is no polynomial-time algorithm for $(1 + \varepsilon)$-approximation.

(unless P = NP)

Our Results

Our results are shown in orange.

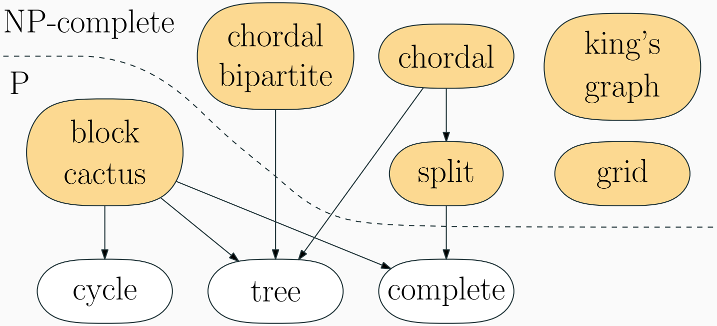
- Extend and unify all known polynomial-time solvable cases.
- Show NP-completeness for some graph classes.



Our Results

In addition, we present general tools to show the NP-completeness: If a graph class C satisfies some conditions, then STS on C is NP-complete.

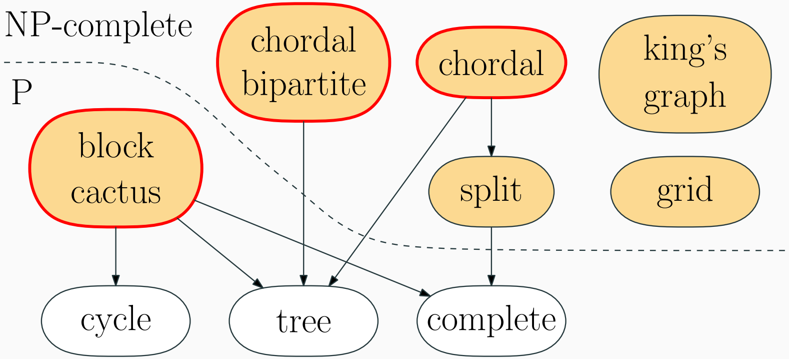
Using these tools, we show the NP-completeness on chordal / chordal bipartite graphs.



Our Results

Today, I will talk about the following results only:

- The poly-time algorithm for block-cactus graphs.
- General tools to show the NP-completeness.



Block-Cactus Graphs

block-cactus graphs

A graph is a **block-cactus graph** if every biconnected component is a cycle or a complete graph.

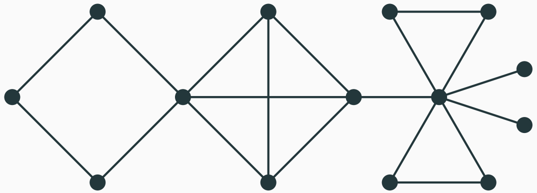
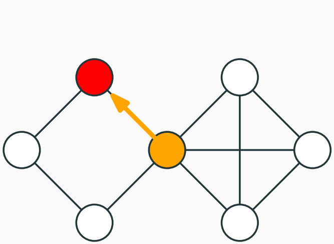


Figure 2: An example of block-cactus graphs

Key Observation

Key Observation

Except for the moving token, a token cannot escape from its biconnected component.

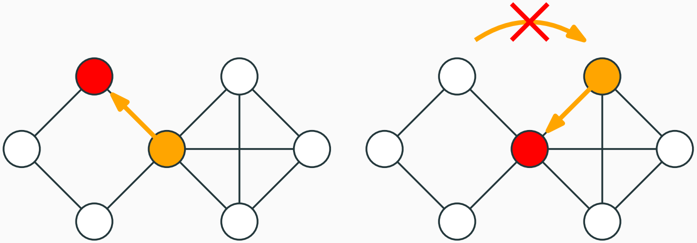


To move **the red token** to the right biconnected component (excluding the cut vertex), you need to do the above two steps.

Key Observation

To move the red token to the right, you need to

1. place the moving token on the cut vertex and swap.
2. place the moving token on the right side and swap, **keeping the red token placed on the cut vertex**.



This is impossible. To keep the red token unmoved, the moving token must avoid visiting the cut vertex.

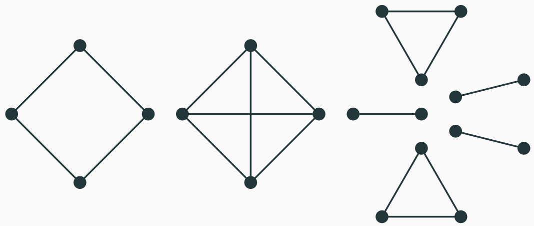
How We Use the Key Observation

Key Observation

Except for the moving token, a token cannot escape from its biconnected component.

⇒ Configurations on biconnected components are almost independent.

⇒ Can we divide the problem into the subproblems on biconnected components?



Subproblem on Biconnected Component

If we fix the start and end vertices of the moving token, STS can be reduced to the following problem on each biconnected component.

Sub-STS (optimization problem)

Input:

A graph G, configurations f, f', and a vertex set P.

Output:

The minimum length of the walk such that:

- It is a solution for the STS instance (G, f, f').
- It visits all the vertices in P.

The Algorithm for Block-Cacutus Graphs

Overview

Try for all $(s, t) \in V^2$:

1. Fix the start and end vertices to s, t.
2. For each biconnected component, solve Sub-STS.
3. Check if “(the sum of Sub-STS) $\leq k$”.

If there exists a pair (s, t) that satisfies the condition, then output YES (otherwise NO).

We give polynomial-time algorithms for Sub-STS on cycles and complete graphs. Thus, the above algorithm runs in polynomial time on block-cactus graphs.

Fundamental Intractability in STS

Lastly, I will talk about the general tools.

Sub-STS (optimization problem)

Input:

A graph G, configurations f, f', and a vertex set P.

Output:

The minimum length of the walk such that:

- It is a solution for the STS instance (G, f, f').
- It visits all the vertices in P.

As we saw in Sub-STS, STS contains a problem like

"compute the minimum length to visit all given vertices."

Fundamental Intractability in STS

"compute the minimum length to visit all given vertices."

Problems of this kind are often intractable.

For instance, the following problems are NP-complete.

Hamiltonian path problem

Decide if there exists a path that visits all vertices exactly once.

Steiner tree problem

Given a vertex set $K \subseteq V$ and an integer l,
decide if there exists a subgraph, with at most l vertices, that contains K.

General Tool via Hamiltonian Path

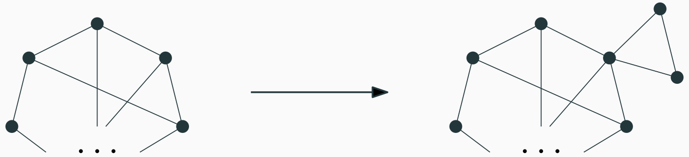
We use those problems in general tools.

Theorem

Let C be a graph class that satisfies the following:

- the Hamiltonian path problem is NP-complete on C.
- C is closed under “attaching a fixed-size cycle”.

Then STS on C is NP-complete even if the number of colors is a constant (≥ 2).



General Tool via Steiner Tree

In contrast, the Steiner tree problem can be used for the case where each token has a distinct color.

Theorem

Let C be a graph class that satisfies the following:

- the Steiner tree problem is NP-complete on C.
- C is closed under “attaching a fixed-size cycle”.

Then STS on C is NP-complete even if the number of colors is n.

Comparison of Two tools

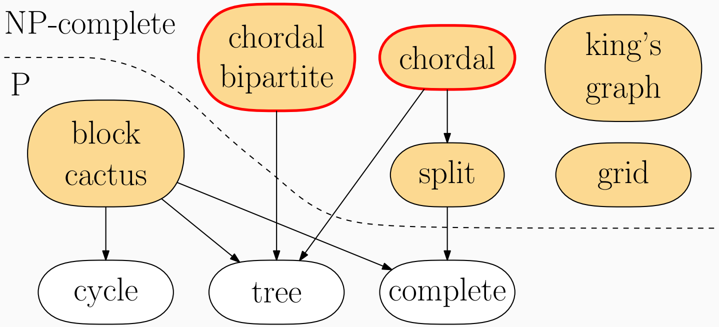
Let C be a graph class that is closed under “attaching a fixed-size cycle”. Then,

- Hamiltonian path on C is NP-complete
  $\Rightarrow$ STS is NP-complete even if the number of colors is a constant.
- Steiner tree on C is NP-complete
  $\Rightarrow$ STS is NP-complete even if the number of colors is n (each token has a distinct color).

Summary

Revealed the complexity of STS on some graph classes.

For some of them, we applied the general tools.



Future Direction

FPT with structural graph parameters (e.g. treewidth)?