Reconfiguring spanning and induced subgraphs

Tesshu Hanaka[1,a)] Takehiro Ito[2,b)] Haruka Mizuta[2,c)] Benjamin Moore[3,d)]
Naomi Nishimura[3,e)] Vijay Subramanya[3,f)] Akira Suzuki[2,g)]
Krishna Vaidyanathan[3,h)]

Abstract: Subgraph reconfiguration is a family of problems focusing on the reachability of the solution space in which feasible solutions are subgraphs, represented either as sets of vertices or sets of edges, satisfying a prescribed graph structure property. Although there has been previous work that can be categorized as subgraph reconfiguration, most of the related results appear under the name of the property under consideration; for example, independent set, clique, and matching. In this paper, we systematically clarify the complexity status of subgraph reconfiguration with respect to graph structure properties.

1. Introduction

Combinatorial reconfiguration [5], [6], [12] studies the reachability/connectivity of the solution space formed by feasible solutions of an instance of a search problem. More specifically, consider a graph such that each node in the graph represents a feasible solution to an instance of a search problem P, and there is an edge between nodes representing any two feasible solutions that are "adjacent," according to a prescribed *reconfiguration rule* $\mathcal{A}$; such a graph is called the *reconfiguration graph* for P and $\mathcal{A}$. In the *reachability problem* for P and $\mathcal{A}$, we are given *source* and *target* solutions to P, and the goal is to determine whether or not there is a path between the two corresponding nodes in the reconfiguration graph for P and $\mathcal{A}$. We call a desired path a *reconfiguration sequence* between source and target solutions, where a *reconfiguration step* from one solution to another corresponds to an edge in the path.

1.1 Subgraph reconfiguration

In this paper, we use the term subgraph reconfiguration to describe a family of reachability problems that take subgraphs (more accurately, vertex subsets or edge subsets of a given graph) as feasible solutions. Each of the individual problems in the family can be defined by specifying the node set and the edge set of a reconfiguration graph, as follows. (We use the terms *node* for reconfiguration graphs and *vertex* for input graphs.)

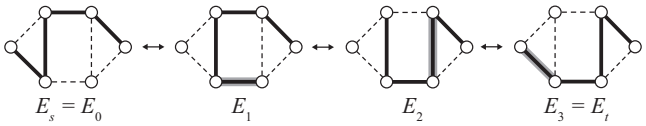

Fig. 1 A reconfiguration sequence $\langle E_0, E_1, E_2, E_3 \rangle$ in the edge variant under the TJ rule (also under the TS rule) with the property "a graph is a path," where the edges forming solutions are depicted by thick lines.

Nodes of a reconfiguration graph. The set of feasible solutions (i.e., subgraphs) can be defined in terms of a specified graph structure property Π which subgraphs must satisfy; for example, "a graph is a tree," "a graph is edgeless (an independent set)," and so on. By the choice of how to represent subgraphs, each specific problem in the family can be categorized into one of three variants. (See also Table 1.) If a subgraph is represented as an edge subset, which we will call the *edge variant*, then the subgraph formed (induced) by the edge subset must satisfy Π. For example, Fig. 1 illustrates four subgraphs represented as edge subsets, where Π is "a graph is a path." On the other hand, if a subgraph is represented as a vertex subset, we can opt either to require that the subgraph induced by the vertex subset satisfies Π or that the subgraph induced by the vertex subset contains at least one spanning subgraph that satisfies Π; we will refer to these as the *induced variant* and *spanning variant*, respectively. For example, if Π is "a graph is a path," then in the induced variant, the vertex subset must induce a path, whereas in the spanning variant, the vertex subset is feasible if its induced subgraph contains at least one Hamiltonian path. Figure 2 illustrates feasible vertex subsets of the induced variant and spanning variant. In the figure, the vertex subset V_1' is feasible in the spanning variant, but is

1 Chuo University, Tokyo, Japan.
2 Tohoku University, Sendai, Japan.
3 University of Waterloo, Waterloo, Canada.
a) hanaka.91t@g.chuo-u.ac.jp
b) takehiro@ecei.tohoku.ac.jp
c) haruka.mizuta.s4@dc.tohoku.ac.jp
d) brmoore@uwaterloo.ca
e) nishi@uwaterloo.ca
f) v7subram@uwaterloo.ca
g) a.suzuki@ecei.tohoku.ac.jp
h) kvaidyan@uwaterloo.ca

Table 1 Subgraph representations and variants

Subgraph representations	Variant names	Known reachability problems
edge subset	edge	spanning tree [6] matching [6], [11], and b-matching [11]
vertex subset	induced	clique [7] independent set [6], [8] induced forest [10] induced bipartite [10] induced tree [13]
	spanning	clique [7]

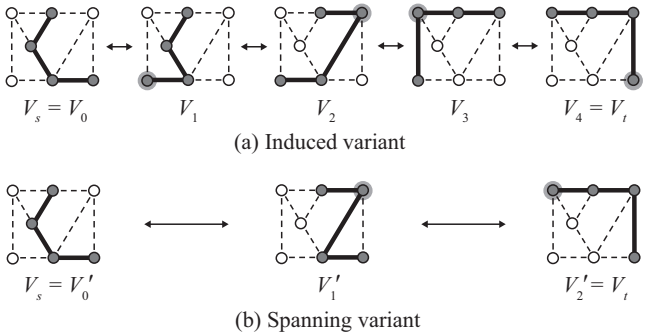

Fig. 2 Reconfiguration sequences $\langle V_0, V_1, V_2, V_3, V_4 \rangle$ in the induced variant under the TJ rule and $\langle V_0', V_1', V_2' \rangle$ in the spanning variant under the TJ rule with the property "a graph is a path," where the vertices forming solutions are depicted by colored circles, and the subgraphs satisfying the property by thick lines.

not feasible in the induced variant, because it contains a spanning path but does not induce a path. As can be seen by this simple example, in the spanning variant, we need to pay attention to the additional complexity of finding a spanning subgraph and the complications resulting from the fact that the subgraph induced by the vertex subset may contain more than one spanning subgraph which satisfies Π.

Edges of a reconfiguration graph. Since we represent a feasible solution by a set of vertices (or edges) in any variant, we can consider that tokens are placed on each vertex (resp., edge) in the feasible solution. Then, in this paper, we mainly deal with two well-known reconfiguration rules, called the token-jumping (TJ) [8] and token-sliding (TS) rules [2], [4], [8]. In the former, a token can move to any other vertex (edge) in a given graph, whereas in the latter it can move only to an adjacent vertex (adjacent edge, that is sharing a common vertex.) For example, Fig. 1 and Fig. 2 illustrate reconfiguration sequences under the TJ rule for each variant. Note that the sequence in Fig. 1 can also be considered as a sequence under the TS rule. In the reconfiguration graph, two nodes are adjacent if and only if one of the two corresponding solutions can be obtained from the other one by a single move of one token that follows the specified reconfiguration rule. Therefore, all nodes in a connected component of the reconfiguration graph represent subgraphs having the same number of vertices (edges).

We note in passing that since in most cases we wish to retain the same number of vertices and/or edges, we rarely use the token-addition-and-removal (TAR) rule [6], [8], where we can add or remove a single token at a time, for SUBGRAPH RECONFIGURATION problems.

1.2 Previous work

Although there has been previous work that can be categorized as SUBGRAPH RECONFIGURATION, most of the related results appear under the name of the property Π under consideration. Accordingly, we can view reconfiguration of independent sets [6], [8] as the induced variant of SUBGRAPH RECONFIGURATION such that the property Π is "a graph is edgeless." Other examples can be found in Table 1. We here explain only known results which are directly related to our contributions.

Reconfiguration of cliques can be seen as both the spanning and the induced variant; the problem is PSPACE-complete under any rule, even when restricted to perfect graphs [7]. Indeed, for this problem, the rules TAR, TJ, and TS have all been shown to be equivalent from the viewpoint of polynomial-time solvability. It is also known that reconfiguration of cliques can be solved in polynomial time for several well-known graph classes [7].

Wasa et al. [13] considered the induced variant under the TJ and TS rules with the property Π being "a graph is a tree." They showed that this variant under each of the TJ and TS rules is PSPACE-complete, and that under the TJ rule is W[1]-hard when parameterized by both the size of a solution and the length of a reconfiguration sequence. They also gave a fixed-parameter algorithm when parameterized by both the size of a solution and the maximum degree of an input graph, under both the TJ and TS rules. In closely related work, Mouawad et al. [10] considered the induced variants of SUBGRAPH RECONFIGURATION under the TAR rule with the properties Π being either "a graph is a forest" or "a graph is bipartite." They showed that these variants are W[1]-hard when parameterized by the size of a solution plus the length of a reconfiguration sequence.

1.3 Our contributions

In this paper, we study the complexity of SUBGRAPH RECONFIGURATION under the TJ and TS rules. (Our results are summarized in Table 2, together with known results, where an (i, j)-*biclique* is a complete bipartite graph with the bipartition of i vertices and j vertices.) As mentioned above, because we consider the TJ and TS rules, it suffices to deal with subgraphs having the same number of vertices or edges. Subgraphs of the same size may be isomorphic for certain properties Π, such as "a graph is a path" and "a graph is a clique," because there is only one choice of a path or a clique of a particular size. On the other hand, for

Table 2 Previous and new results

Property Π	Edge variant	Induced variant	Spanning variant
path	NP-hard (TJ) [Theorem 2]	PSPACE-c. (TJ, TS) [Theorems 7]	PSPACE-c. (TJ, TS) [Theorems 7]
cycle	P (TJ, TS) [Theorem 3]	PSPACE-c. (TJ, TS) [Theorems 7]	PSPACE-c. (TJ, TS) [Theorems 7]
tree	P (TJ) [Theorem 6]	PSPACE-c. (TJ, TS) [13]	P (TJ) PSPACE-c. (TS) [Theorems 9, 8]
(i,j)**-biclique**	P (TJ, TS) [Theorem 5]	PSPACE-c. for $i=j$ (TJ) PSPACE-c. for fixed i (TJ) [Theorem 10]	NP-hard for $i=j$ (TJ) P for fixed i (TJ) [Theorems 11, 12]
clique	P (TJ, TS) [Theorem 4]	PSPACE-c. (TJ, TS) [7]	PSPACE-c. (TJ, TS) [7]
diameter two		PSPACE-c. (TS) [Theorem 13]	PSPACE-c. (TS) [Theorem 13]
any property	XP for solution size (TJ, TS) [Theorem 1]	XP for solution size (TJ, TS) [Theorem 1]	XP for solution size (TJ, TS) [Theorem 1]

the property "a graph is a tree," there are several choices of trees of a particular size. (We will show an example in Section 3 with Fig. 4.)

As shown in Table 2, we systematically clarify the complexity of SUBGRAPH RECONFIGURATION for several fundamental graph properties. In particular, we show that the edge variant under the TJ rule is computationally intractable for the property "a graph is a path" but tractable for the property "a graph is a tree." This implies that the computational (in)tractability does not follow directly from the inclusion relationship of graph classes required as the properties Π; one possible explanation is that the path property implies a specific graph, whereas the tree property allows several choices of trees, making the problem easier.

We omitted proofs for the claims marked with (*) from this extended abstract.

1.4 Preliminaries

Although we assume throughout the paper that an input graph G is simple, all our algorithms except for Theorem 1 can be easily extended, with small modifications, to graphs having multiple edges. We denote by (G, V_s, V_t) an instance of a spanning variant or an induced variant whose input graph is G and source and target solutions are vertex subsets V_s and V_t of G. Similarly, we denote by (G, E_s, E_t) an instance of the edge variant. We may assume without loss of generality that $|V_s| = |V_t|$ holds for the spanning and induced variants, and $|E_s| = |E_t|$ holds for the edge variant; otherwise, the answer is clearly no since under both the TJ and TS rules, all solutions must be of the same size.

2. General algorithm

In this section, we give a general XP algorithm when the size of a solution (that is, the size of a vertex or edge subset that represents a subgraph) is taken as the parameter. For notational convenience, we simply use *element* to represent a vertex (or an edge) for the spanning and induced variants (resp., the edge variant), and *candidate* to represent a set of elements (which does not necessarily satisfy the property Π). Furthermore, we define the *size* of a given graph as the number of elements in the graph.

Theorem 1 (*)**.** *Let Π be any graph structure property such that we can check if a candidate of size k satisfies Π in $f(k)$ time, where $f(k)$ is a computable function depending only on k. Then, all of the spanning, induced, and edge variants under the TJ or TS rules can be solved in time $O(n^{2k}k + n^k f(k))$, where n is the size of a given graph and k is the size of a source (and target) solution. Furthermore, a shortest reconfiguration sequence between source and target solutions can be found in the same time bound, if it exists.*

3. Edge variants

In this section, we study the edge variant of SUBGRAPH RECONFIGURATION for the properties of being paths, cycles, cliques, bicliques, and trees.

We first consider the property "a graph is a path" under the TJ rule.

Theorem 2. *The edge variant of* SUBGRAPH RECONFIGURATION *under the TJ rule is NP-hard for the property "a graph is a path."*

Proof. We give a polynomial-time reduction from the HAMILTONIAN PATH problem. Recall that a *Hamiltonian path* in a graph G is a path that visits each vertex of G exactly once. Given a graph G and two vertices $s, t \in V(G)$ of G, the NP-complete problem HAMILTONIAN PATH is to determine whether or not G has a Hamiltonian path which starts from s and ends in t [3].

For an instance (G, s, t) of HAMILTONIAN PATH, we construct a corresponding instance (G', E_s, E_t) of our problem, as follows. (See also Fig. 3.) Let $n = |V(G)|$. We first add two new vertices v and x to G with two new edges $e_1 = xv$ and $e_2 = vs$. We then add two paths $P_s = \langle s_1, s_2, \ldots, s_{n+1}, x\rangle$ and $P_t = \langle t_1, t_2, \ldots, t_{n+1}, x\rangle$, where $s_1, s_2, \ldots, s_{n+1}$ and $t_1, t_2, \ldots, t_{n+1}$ are distinct new vertices. Each of P_s and P_t consists of $n+1$ edges; we denote by $e^s_1, e^s_2, \ldots, e^s_{n+1}$ the edges s_1s_2, s_2s_3, $\ldots$, $s_{n+1}x$ in P_s, respectively, and by $e^t_1, e^t_2, \ldots, e^t_{n+1}$ the edges

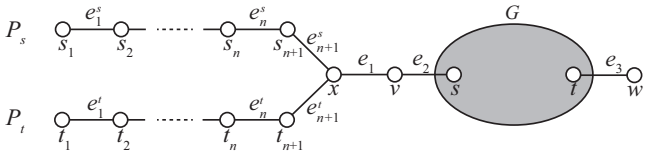

Fig. 3 Reduction to the edge variant under the TJ rule for the property "a graph is a path."

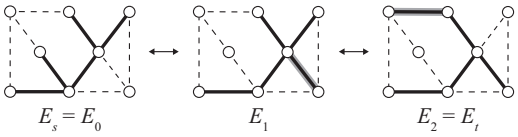

Fig. 4 Reconfiguration sequence $\langle E_0, E_1, E_2 \rangle$ in the edge variant under the TJ rule with the property "a graph is a tree."

$t_1t_2,\ t_2t_3,\ \ldots, t_{n+1}x$ in P_t, respectively. We finally add a new vertex w with an edge $e_3 = tw$, completing the construction of G'. We then set $E_s = \{e^s_1, e^s_2, \ldots, e^s_{n+1}, e_1, e_2\}$ and $E_t = \{e^t_1, e^t_2, \ldots, e^t_{n+1}, e_1, e_2\}$; these edge subsets clearly form paths in G'. We have thus constructed our corresponding instance (G', E_s, E_t) in polynomial time.

We now prove that an instance (G, s, t) of Hamiltonian path is a yes-instance if and only if the corresponding instance (G', E_s, E_t) is a yes-instance.

To prove the only-if direction, we first suppose that G has a Hamiltonian path P starting from s and ending in t. Then, we construct an actual reconfiguration sequence from E_s to E_t using the edges in P. Notice that P consists of $n-1$ edges. Thus, we first move the $n-1$ edges $e^s_1, e^s_2, \ldots, e^s_{n-1}$ in E_s to the edges in P one by one, and then move e^s_n to e_3. Next, we move e^s_{n+1} to e^t_{n+1}, and then move the edges in $E(P) \cup \{e_3\}$ to $e^t_n, e^t_{n-1}, \ldots, e^t_1$ one by one. By the construction of G', we know that each of the intermediate edge subsets forms a path in G', as required.

We now prove the if direction by supposing that there exists a reconfiguration sequence $\langle E_s = E_0, E_1, \ldots, E_\ell = E_t \rangle$. Let E_q be the first edge subset in the sequence such that $E(P_t) \cap E_q \neq \emptyset$; we claim that E_q contains a Hamiltonian path in G. First, notice that the edge in $E(P_t) \cap E_q$ is e^t_{n+1}; otherwise the subgraph formed by E_q is disconnected. Since $|E_q| = |E_s| = n+3$ and $|E(P_s)| = n+1$, we can observe that E_q contains no edge in P_s; otherwise the degree of x would be three, or E_q would form a disconnected subgraph. Therefore, the $n+2$ edges in $E_q \setminus \{e^t_{n+1}\}$ must be chosen from $E(G) \cup \{e_1, e_2, e_3\}$. Since $|V(G)| = n$ and E_q must form a path in G', we know that $E_q \setminus \{e^t_{n+1}\}$ consists of e_1, e_2, e_3 and $n-1$ edges in G. Thus, $E_q \setminus \{e^t_{n+1}, e_1, e_2, e_3\}$ forms a Hamiltonian path in G starting from s and ending in t, as required. □

We now consider the property "a graph is a cycle," as follows.

Theorem 3 (*). *The edge variant of* subgraph reconfiguration *under each of the TJ and TS rules can be solved in linear time for the property "a graph is a cycle."*

The same statement holds for the property "a graph is a clique," and we obtain the following theorem. We note that, for this property, both induced and spanning variants (i.e., when solutions are represented by vertex subsets) are PSPACE-complete under any rule [7].

Theorem 4 (*). *The edge variant of* subgraph reconfiguration *under each of the TJ and TS rules can be solved in linear time for the property "a graph is a clique."*

We next consider the property "a graph is an (i,j)-biclique," as follows.

Theorem 5 (*). *The edge variant of* subgraph reconfiguration *under each of the TJ and TS rules can be solved in polynomial time for the property "a graph is an (i,j)-biclique" for any pair of positive integers i and j.*

We finally consider the property "a graph is a tree" under the TJ rule. As we have mentioned in the introduction, for this property, there are several choices of trees even of a particular size, and a reconfiguration sequence does not necessarily consist of isomorphic trees (see Fig. 4). This "flexibility" of subgraphs may yield the contrast between Theorem 2 for the path property and the following theorem for the tree property.

Theorem 6. *The edge variant of* subgraph reconfiguration *under the TJ rule can be solved in linear time for the property "a graph is a tree."*

Proof. Suppose that (G, E_s, E_t) is a given instance. We may assume without loss of generality that $|E_s| = |E_t| \geq 2$; otherwise $|E_s| = |E_t| \leq 1$ holds, and hence the instance is trivially a yes-instance. We will prove below that any instance with $|E_s| = |E_t| \geq 2$ is a yes-instance if and only if all the edges in E_s and E_t are contained in the same connected component of G. Note that this condition can be checked in linear time.

We first prove the only-if direction of our claim. Since $|E_s| = |E_t| \geq 2$ and subgraphs always must retain a tree structure (more specifically, they must be connected graphs), observe that we can exchange edges only in the same connected component of G. Thus, the only-if direction follows.

To complete the proof, it suffices to prove the if direction of our claim. For notational convenience, for any feasible solution E_i we denote by T_i the tree represented by E_i, and by V_i the vertex set of T_i. In this direction, we consider the following two cases: (a) $V_s \cap V_t = \emptyset$, and (b) $V_s \cap V_t \neq \emptyset$.

We consider Case (a), that is, $V_s \cap V_t = \emptyset$. Since T_s and T_t are contained in one connected component of G, there exists a path $\langle v_0, v_1, \ldots, v_\ell \rangle$ in G such that $v_0 \in V_s$, $v_\ell \in V_t$ and $v_i \notin V_s \cup V_t$ for all $i \in \{1, 2, \ldots, \ell - 1\}$. Since T_s is a tree, it has at least two degree-one vertices. Let v_s be any degree-one vertex in $V_s \setminus \{v_0\}$, and let e_s be the leaf edge of T_s incident to v_s. Then, we can exchange e_s with v_0v_1, and obtain another tree represented by the resulting edge subset $(E_s \cup \{v_0v_1\}) \setminus \{e_s\}$. By repeatedly applying this operation along the path $\langle v_1, v_2, \ldots, v_\ell \rangle$, we can obtain a solution E_k such that $V_k \cap V_t = \{v_\ell\} \neq \emptyset$; this case will be considered below.

We finally consider Case (b), that is, $V_s \cap V_t \neq \emptyset$. Consider

the graph $(V_s \cap V_t, E_s \cap E_t)$. Then, $(V_s \cap V_t, E_s \cap E_t)$ is a forest, and let $G' = (V', E')$ be a connected component (i.e., a tree) of $(V_s \cap V_t, E_s \cap E_t)$ whose edge set is of maximum size. We now prove that there is a reconfiguration sequence between E_s and E_t by induction on $k = |E_s \setminus E'| = |E_t \setminus E'|$. If $k = 0$, then $E_s = E' = E_t$ and hence the claim holds. We thus consider the case where $k > 0$ holds. Since G' is a proper subtree of T_t, there exists at least one edge e_t in $E_t \setminus E'$ such that one endpoint of e_t is contained in V' and the other is not. We claim that there exists an edge e_s in $E_s \setminus E'$ which can be moved into e_t, that is, the subgraph represented by the resulting edge subset $(E_s \cup \{e_t\}) \setminus \{e_s\}$ forms a tree. If both endpoints of e_t are contained in V_s (not just V'), $E_s \cup \{e_t\}$ contains a cycle; let $C \subseteq E_s \cup \{e_t\}$ be the edge set of the cycle. Since the subgraph T_t has no cycle, there exists at least one edge in $C \setminus E_t$, and we choose one of them as e_s. On the other hand, if just one endpoint of e_t is contained in V_s, then we choose a leaf edge of T_s in $E_s \setminus E'$ as e_s. Note that there exists such a leaf edge since G' is a proper subtree of T_s. From the choice of e_s and e_t, we know the subgraph represented by the resulting edge subset $(E_s \cup \{e_t\}) \setminus \{e_s\}$ forms a tree; let $E_k = (E_s \cup \{e_t\}) \setminus \{e_s\}$. Furthermore, since $E_k \cap E_t$ includes $E' \cup \{e_t\}$ and the subgraph formed by $E' \cup \{e_t\}$ is connected, the subgraph formed by $E_k \cap E_t$ has a connected component whose edge set has size at least $|E'| + 1$. Therefore, we can conclude that E_k is reconfigurable into E_t by the induction hypothesis. □

4. Induced and spanning variants

In this section, we deal with the induced and spanning variants where subgraphs are represented as vertex subsets.

4.1 Path and cycle

The following is the main theorem of this subsection.

Theorem 7 (*). *Both the induced and spanning variants of* SUBGRAPH RECONFIGURATION *under the TJ and TS rules are PSPACE-complete for each of the properties "a graph is a path" and "a graph is a cycle."*

To prove the theorem, we give polynomial-time reductions from the SHORTEST PATH RECONFIGURATION problem, which can be seen as a SUBGRAPH RECONFIGURATION problem under the TJ rule. This problem is known to be PSPACE-complete [1].

4.2 Tree

Wasa et al. [13] showed that the induced variant under the TJ and TS rules is PSPACE-complete for the property "a graph is a tree." In this subsection, we show that the spanning variant for this property is also PSPACE-complete under the TS rule, while it is linear-time solvable under the TJ rule.

We first give the hardness result.

Theorem 8 (*). *The spanning variant of* SUBGRAPH RECONFIGURATION *under the TS rule is PSPACE-complete for the property "a graph is a tree."*

In contrast to Theorem 8, the spanning variant under the TJ rule is solvable in linear time.

Theorem 9 (*). *The spanning variant of* SUBGRAPH RECONFIGURATION *under the TJ rule can be solved in linear time for the property "a graph is a tree."*

4.3 Biclique

For the property "a graph is an (i, j)-biclique," we show that the induced variant under the TJ rule is PSPACE-complete even if $i = j$ holds, or i is fixed. On the other hand, the spanning variant under the TJ rule is NP-hard even if $i = j$ holds, while it is polynomial-time solvable when i is fixed.

We first give the following theorem.

Theorem 10 (*). *For the property "a graph is an (i, j)-biclique," the induced variant of* SUBGRAPH RECONFIGURATION *under the TJ rule is PSPACE-complete even if $i = j$ holds, or i is any fixed positive integer.*

To show the theorem, we give a polynomial-time reduction from the MAXIMUM INDEPENDENT SET RECONFIGURATION problem [14], which can be seen as a SUBGRAPH RECONFIGURATION problem. This problem is known to be PSPACE-complete under the TJ and TS rules [14].

We next give the following theorem.

Theorem 11 (*). *For the property "a graph is an (i, j)-biclique," the spanning variant of* SUBGRAPH RECONFIGURATION *under the TJ rule is NP-hard even if $i = j$ holds.*

The theorem can be shown by a polynomial-time reduction from the BALANCED COMPLETE BIPARTITE SUBGRAPH problem, which is known to be NP-hard [3].

We now give a polynomial-time algorithm solving the spanning variant for a fixed constant $i \geq 1$.

Theorem 12. *For the property "a graph is an (i, j)-biclique," the spanning variant of* SUBGRAPH RECONFIGURATION *under the TJ rule is solvable in polynomial time when $i \geq 1$ is a fixed constant.*

We give such an algorithm as a proof of Theorem 12. We will refer to the i vertices in the bounded-size part of the biclique as *hubs*, and the j vertices in the other part as *terminals*. Let $H \subseteq V(G)$ be an arbitrary vertex subset such that $|H| = i$. We denote by $C(H) \subseteq V(G)$ the set of all common neighbors of H in G, i.e., $C(H) = \bigcap_{v \in H} \{u \in V(G) \mid uv \in E(G)\}$. We write $C[H] = C(H) \cup H$. We denote by $\mathcal{S}(H)$ the set of all solutions that contain (i, j)-bicliques with the hub set H. We know that $\mathcal{S}(H) \neq \emptyset$ if and only if $|C(H)| \geq j$ holds; if $|C(H)| \geq j$, then $H \cup T$ is in $\mathcal{S}(H)$ for any subset $T \subseteq C(H)$ such that $|T| = j$. We also observe that $W \subseteq C[H]$ holds for any solution $W \in \mathcal{S}(H)$. It should be noted that a solution in the spanning variant is simply a vertex subset V' of $V(G)$, and there is no restriction on how to choose a hub set from V'. (For example, if a solution V' induces a clique of size five, then there are ten ways to choose a hub set from V' for $(2, 3)$-bicliques.) Therefore, $\mathcal{S}(H) \cap \mathcal{S}(H') \neq \emptyset$ may hold for distinct hub sets H, H'.

We describe two key observations in the following. The

first one is that for a hub set H, any two solutions $W, W' \in \mathcal{S}(H)$ are reconfigurable because we can always move vertices in $W \setminus W'$ into ones in $W' \setminus W$ one by one. The second one is that for any two distinct hub sets H_a and H_b, if there exist $V_a \in \mathcal{S}(H_a)$ and $V_b \in \mathcal{S}(H_b)$ such that $|V_a \setminus V_b| = |V_b \setminus V_a| \leq 1$ (this means that V_a and V_b are reconfigurable by one reconfiguration step, or $V_a = V_b$), then all pairs of solutions in $\mathcal{S}(H_a) \cup \mathcal{S}(H_b)$ are reconfigurable.

Based on these observations, we construct an *auxiliary graph* A for a given instance (G, V_s, V_t), as follows. Each node in A corresponds to a set H of i vertices (hubs) in the input graph G such that $|C(H)| \geq j$; we represent a node in A simply by the corresponding hub set H. Two nodes H_a and H_b are adjacent in A if there exist $V_a \in \mathcal{S}(H_a)$ and $V_b \in \mathcal{S}(H_b)$ such that $|V_a \setminus V_b| = |V_b \setminus V_a| \leq 1$. We first prove the following lemma.

Lemma 1 (*). *Let H_s and H_t be any two nodes in A such that $V_s \in \mathcal{S}(H_s)$ and $V_t \in \mathcal{S}(H_t)$, respectively. Then, there is a reconfiguration sequence between V_s and V_t if and only if there is a path in A between H_s and H_t.*

Our algorithm first constructs an auxiliary graph A, and checks whether or not there is a path between H_s and H_t; the correctness of the algorithm follows directly from Lemma 1. However, it is not so obvious how to construct the auxiliary graph A in the desired running time. Observe that we can construct the node set of A in the desired running time, because we just need to check whether $|C(H)| \geq j$ or not for any vertex subset $H \subseteq V(G)$ of exactly i vertices. To construct the edge subset, we give the following lemma.

Lemma 2 (*). *Any two nodes H_a and H_b in A are joined by an edge in A if and only if all the following four conditions hold*:

(a) $|C[H_a] \cap C[H_b]| \geq i + j - 1$;
(b) $|H_a \setminus C[H_b]| \leq 1$;
(c) $|H_b \setminus C[H_a]| \leq 1$; *and*
(d) $|H_a \cup H_b| \leq i + j + 1$.

Finally, we give the following lemma, which completes the proof of Theorem 12.

Lemma 3 (*). *The algorithm runs in $O(n^{2i+1})$ time.*

4.4 Diameter-two graph

In this subsection, we consider the property "a graph has diameter at most two." Note that the induced and spanning variants are the same for this property.

Theorem 13 (*). *Both induced and spanning variants of* SUBGRAPH RECONFIGURATION *under the TS rule are PSPACE-complete for the property "a graph has diameter at most two."*

5. Conclusions

The work in this paper initiates a systematic study of SUBGRAPH RECONFIGURATION. Although we have identified graph structure properties which are harder for the induced variant than the spanning variant, it remains to be seen whether this pattern holds in general. For the general case, questions of the roles of diameter and the number of subgraphs satisfying the property are worthy of further investigation. Another obvious direction for further research is an investigation into the fixed-parameter complexity of SUBGRAPH RECONFIGURATION.

Acknowledgments This work is partially supported by JST ERATO Grant Number JPMJER1201, JST CREST Grant Number JPMJCR1402, and JSPS KAKENHI Grant Numbers JP16K00004, JP17K12636, JP18H04091, and JP18H06469, Japan. Research by Canadian authors is supported by the Natural Science and Engineering Research Council of Canada.

References

[1] Bonsma, P.: The complexity of rerouting shortest paths, *Theoretical Computer Science*, Vol. 510, pp. 1–12 (2013).
[2] Bonsma, P. and Cereceda, L.: Finding paths between graph colourings: PSPACE-completeness and superpolynomial distances, *Theoretical Computer Science*, Vol. 410, No. 50, pp. 5215–5226 (2009).
[3] Garey, M. R. and Johnson, D. S.: *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman (1979).
[4] Hearn, R. A. and Demaine, E. D.: PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation, *Theoretical Computer Science*, Vol. 343, No. 1–2, pp. 72–96 (2005).
[5] van den Heuvel, J.: The complexity of change, *Surveys in Combinatorics 2013*, London Mathematical Society Lecture Note Series, Vol. 409, Cambridge University Press, pp. 127–160 (2013).
[6] Ito, T., Demaine, E. D., Harvey, N. J. A., Papadimitriou, C. H., Sideri, M., Uehara, R. and Uno, Y.: On the complexity of reconfiguration problems, *Theoretical Computer Science*, Vol. 412, No. 12–14, pp. 1054–1065 (2011).
[7] Ito, T., Ono, H. and Otachi, Y.: Reconfiguration of cliques in a graph, *Proceedings of the 12th Annual Conference on Theory and Applications of Models of Computation* (*TAMC 2015*), Lecture Notes in Computer Science, Vol. 9076, pp. 212–223 (2015).
[8] Kamiński, M., Medvedev, P. and Milanič, M.: Complexity of independent set reconfigurability problems., *Theoretical Computer Science*, Vol. 439, pp. 9–15 (2012).
[9] Moore, B., Nishimura, N. and Subramanya, V.: Reconfiguration of graph minors, *Proceedings of the 43rd International Symposium on Mathematical Foundations of Computer Science* (*MFCS 2018*), Leibniz International Proceedings in Informatics, Vol. 117, pp. 75:1–75:15 (2018).
[10] Mouawad, A. E., Nishimura, N., Raman, V., Simjour, N. and Suzuki, A.: On the parameterized complexity of reconfiguration problems, *Algorithmica*, Vol. 78, No. 1, pp. 274–297 (2017).
[11] Mühlenthaler, M.: Degree-constrained subgraph reconfiguration is in P, *Proceedings of the 40th International Symposium on Mathematical Foundations of Computer Science* (*MFCS 2015*), Lecture Notes in Computer Science, Vol. 9235, pp. 505–516 (2015).
[12] Nishimura, N.: Introduction to reconfiguration, *Algorithms*, Vol. 11, No. 4, p. 52 (2018).
[13] Wasa, K., Yamanaka, K. and Arimura, H.: The complexity of induced tree reconfiguration problems, *Proceedings of the 10th International Conference of Language and Automata Theory and Applications* (*LATA 2016*), Lecture Notes in Computer Science, Vol. 9618, pp. 330–342 (2016).
[14] Wrochna, M.: Reconfiguration in bounded bandwidth and tree-depth, *Journal of Computer and System Sciences*, Vol. 93, pp. 1–10 (2018).