Independent Set Reconfiguration via Dilworth Decompositions

Ziad Ismaili Alaoui

Abstract

The Token Jumping and Sliding Token problems are fundamental reconfiguration problems defined on the independent sets of an undirected graph. Given two independent sets I and J, each of size k, these problems ask whether there exists a sequence of elementary operations transforming I into J such that every intermediate configuration is also an independent set of size k. Suppose a token is placed on each vertex of I: in Sliding Token, an operation moves a token from a vertex $u \in I$ to an adjacent vertex $v \notin I$; in Token Jumping, the token may instead move to any vertex $v \notin I$. While both problems are PSPACE-complete on general graphs, polynomial-time algorithms for one or both variants have been developed for several graph classes, including trees, block graphs, bipartite permutation graphs, cographs, P_4-tidy graphs, and interval graphs.

In this paper, we prove that both problems are solvable in polynomial time on threshold signed graphs, also known as Dilworth-2 graphs. These are precisely the graphs whose vertex sets can be partitioned into at most two chains under the domination preorder $u \preceq v$ defined by $N(u) \subseteq N[v]$. Our approach exploits this chain structure. For Token Jumping, it extends to an arbitrary number of chains: we obtain an $n^{O(\mathcal{D}(G))}$-time algorithm, where the Dilworth number $\mathcal{D}(G)$ is the minimum number of chains in such a partition. Thus, Token Jumping belongs to XP when parameterised by the Dilworth number.

This graph class is a subclass of permutation graphs, for which the complexity of these problems remains open, and is incomparable with the class of bipartite permutation graphs studied by Fox-Epstein et al. (ISAAC, 2015). Our algorithms are based on the inclusion-chain structure that characterises threshold signed graphs, an algorithmic approach that may be useful beyond this graph class.

Keywords: independent set reconfiguration; token jumping; token sliding; Dilworth number; parameterised complexity; threshold signed graphs.

1 Introduction

Reconfiguration problems typically ask whether one valid solution can be transformed into another via a sequence of elementary steps (see [Nis18] for an introduction). A good example

School of Computer Science and Informatics, University of Liverpool, United Kingdom
Department of Informatics, Philipps-Universität Marburg, Germany
ziad.ismaili-alaoui@liverpool.ac.uk
https://www.ziadism.com

is Independent Set Reconfiguration, which is surprisingly simple to formulate, yet hides rich combinatorial structure. Among its most widely studied variants are Token Jumping and Sliding Token. Imagine placing tokens on the vertices of a graph so that they form an independent set I, and then moving one token at a time (along an edge in Sliding Token, or to any vertex in Token Jumping) until reaching another independent set J of the same size. Throughout the process, the occupied vertices must remain an independent set; i.e. two tokens cannot be adjacent. The question then is: Can all tokens on I be placed onto J in this way? More formally, the problem is defined as follows.

Independent Set Reconfiguration (Token Jumping and Sliding Token)	
Instance:	An undirected graph G and independent sets[1] $I, J \subseteq V(G)$, with $\lvert I\rvert = \lvert J\rvert = k$.
Question:	Does there exist a finite sequence of independent sets $I_1, \ldots, I_m$ such that $I_1 = I$, $I_m = J$, $\lvert I_i\rvert = k$ for all $i \in [m]$, $\lvert I_i \triangle I_{i+1}\rvert = 2$ for all $i \in [m-1]$, and, in the case of Sliding Token, $I_i \triangle I_{i+1} = \{u, v\} \in E(G)$ for all $i \in [m-1]$?

Although both variants are PSPACE-complete when we know nothing about the input graphs [HD05, IDH+11], only a handful of graph classes are known to have enough structure to admit polynomial-time algorithms. Since the introduction of Sliding Token by Hearn and Demaine [HD05] and of Token Jumping (seemingly independently) by Ito et al. [IDH+11] and Kamiński et al. [KMM12], a series of results has gradually expanded this list of tractable classes to include trees, interval graphs, cographs, and several others (see Table 1 for an overview[2]). Nevertheless, positive results remain remarkably scarce. As Bartier, Bousquet, and Mouawad recently observed in [BBM23], "almost no positive result is known for Token Sliding [*sic*][3] even for incredibly simple cases like bounded-degree graphs."
Beyond the theoretical appeal of this problem, some positive results appear to have direct applications in industrial fields such as robotics; for example, see [ZBA26], where Zucker and Ben-Abu show Sliding Token is solvable in linear time on Ptolemaic (gem-free chordal) graphs, which they relate to the hard-core lattice gas model in statistical mechanics.

Our contributions. In this paper, we further contribute to the complexity landscape by proving that both Token Jumping and Sliding Token are solvable in polynomial time on threshold signed graphs (also known as Dilworth-2 graphs). A graph $G = (V, E)$ is a *threshold signed graph* if there exist a mapping $a : V \to \mathbb{R}$ and positive real constants S and T such that $\lvert a(v)\rvert < \min\{S, T\}$ for all $v \in V$ and, for any distinct vertices $u, v \in V$, $\{u, v\} \in E$ if and only if $\lvert a(u) + a(v)\rvert \geq S$ or $\lvert a(u) - a(v)\rvert \geq T$. Threshold signed graphs have been extensively studied in the literature since the late 1970s (see, for example, [FH77, BHdW85b, CP14]) and are recognisable in linear time [BHdW85a]. We see two reasons why threshold signed graphs are a natural class to study.

First, Benzaken et al. [BHdW85b] showed that they form a strict subclass of permutation graphs. Our result then identifies a new tractable non-trivial subclass, while the complexity of both problems on general permutation graphs remains open.[4] Moreover,

[1] We use the terms *independent set* and *configuration* interchangeably.

[2] For cacti under Sliding Token, it was claimed in [AU16] they were solvable in polynomial time, but a later note by the authors revealed that the algorithm was incorrect [Hoa].

[3] The exact name of the problem is not standardised in the literature. Variants such as Sliding Token, Sliding Tokens, and Token Sliding all appear. We arbitrarily follow the terminology of [DDFE+14], despite the other reconfiguration problem being named Token Jumping in our paper.

[4] The problems are known to be tractable on permutation graphs when $k = \alpha(G)$, where $\alpha(G)$ denotes the size of a maximum independent set. Aoike et al. showed this for Sliding Token [AKKO24]; Token Jumping follows from their arguments.

Graph Class	Jumping	Sliding
Arbitrary	PSPACE-c. [IDH+11]	PSPACE-c. [HD05]
Planar	PSPACE-c. [IDH+11, KMM12]	PSPACE-c. [HD05]
Bounded bandwidth	PSPACE-c. [Wro18]	PSPACE-c. [Wro18]
Bipartite	NP-complete [LM18]	PSPACE-c. [LM18]
$C_{2k\geq 4}$-free	P [KMM12]	PSPACE-c. [BKL+21]
Split	P ($C_{2k\geq 4}$-free)	PSPACE-c. [BKL+21]
Trees	P ($C_{2k\geq 4}$-free)	P [DDFE+14]
Chordal	P ($C_{2k\geq 4}$-free)	PSPACE-c. [BKL+21]
Claw-free	P [BKW14]	P [BKW14]
Interval	P ($C_{2k\geq 4}$-free)	P [BB17]
P_4-tidy	Open	P [BVP26]
Cacti	Open	Open
Block	P (chordal)	P [FP25]
Permutation	General: Open Cographs: P [Bon16] Bipartite: Open Threshold signed: P [Thm 1]	General: Open Cographs: P [KMM12] Bipartite: P [FEHOU15] Threshold signed: P [Thm 3]

Table 1: Known complexities for Independent Set Reconfiguration across various graph classes (not exhaustive). Our results contribute towards the open case of permutation graphs.

while Sliding Token is known to be solvable in polynomial time on bipartite permutation graphs [FEHOU15], they and threshold signed graphs are incomparable: neither contains the other (apart from K_2 and C_4, no non-trivial ladder graph is threshold signed[5]).

And second, threshold signed graphs are precisely the Dilworth-2 graphs, that is, graphs whose vertex set can be partitioned into two chains ordered by neighbourhood inclusion. Our algorithms rely almost entirely on this two-chain structure rather than on the numerical threshold representation. Dilworth-1 graphs (threshold graphs) are cographs and hence already well understood from the perspective of reconfiguration [Bon16], which makes Dilworth-2 a natural next step.

To the best of our knowledge, this is the first work on Independent Set Reconfiguration that systematically exploits the Dilworth decomposition of the input graph. We believe this technique may be useful for studying reconfiguration on other graph classes of bounded Dilworth number.

Both of our algorithms are based on the same core idea: Given two independent sets I and J, each algorithm attempts to transform them into a so-called *canonical* configuration, where two mutually reachable configurations have the same canonical configuration. The resulting canonical configurations are then compared: if they are equal, then there exists a sequence of slides or jumps transforming I into J; otherwise, no such sequence exists. This is justified by the fact that every valid reconfiguration sequence can be reversed. This approach has appeared repeatedly in the literature on Independent Set Reconfigura-

[5]Threshold signed graphs are $(C_4 \cup P_2)$-free, which ladder graphs are generally not.

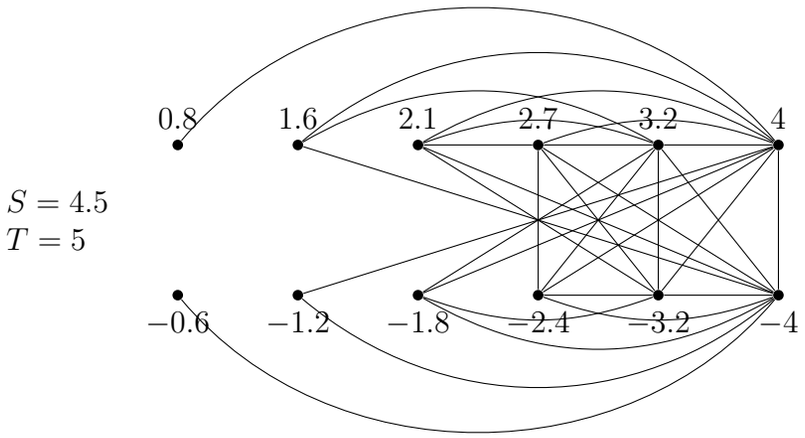

Figure 1: A threshold signed graph with a Dilworth-2 decomposition (the top vertices form one chain, and the bottom ones, the other). In each chain, vertices are ordered left-to-right by increasing neighbourhood inclusion (dominance toward the right). Note that a Dilworth-2 graph need not be a cograph.

tion. For example, Demaine et al. [DDFE+14] showed that every movable configuration on a tree can be transformed into a single canonical configuration. Their algorithm therefore reduces to identifying the tokens that are *rigid*, that is, unable to move in any valid reconfiguration sequence. Other examples include the algorithms for bipartite permutation graphs [FEHOU15], block graphs [FP25], and Ptolemaic graphs [ZBA26].

To give a flavour of our algorithm for Token Jumping, suppose the input graph is partitioned into two neighbourhood-inclusion chains A and B, as guaranteed by the Dilworth-2 property. Starting from an independent set I, we first greedily "flush" all tokens on each chain as far to the left as possible, where the neighbourhood of each vertex is contained in the neighbourhood of any vertex to its right. We then repeatedly move the rightmost token on chain B to the leftmost available vertex on chain A, whenever possible, until no further such move exists. The resulting independent set is the canonical configuration associated with I. We then repeat the same procedure from J to obtain another canonical configuration, and reachability between I and J is simply determined by comparing the two: they coincide if and only if I and J are reachable from one another. In fact, this comparison reduces to checking whether the number of tokens in A (or B by symmetry) is the same in both canonical configurations. We therefore obtain the following result.

Theorem 1. *There exists a polynomial-time algorithm to decide Token Jumping on threshold signed graphs.*

Beyond threshold signed graphs, the same structural observations give an XP algorithm for Token Jumping parameterised by the Dilworth number. As a consequence, this gives a polynomial-time algorithm on Dilworth-3 graphs and, more generally, on every graph class of bounded Dilworth number.

Theorem 2. *There exists an $n^{O(d)}$-time algorithm to decide Token Jumping on arbitrary n-vertex graphs of Dilworth number d.*

As one would expect, the situation for Sliding Token is a tad more delicate, as tokens cannot just go anywhere and must slide along incident edges. In particular, the "flushing" procedure as described for Token Jumping clearly does not work. Fortunately, the same

high-level idea still holds. We show that, at any point, only a small set of tokens needs to be considered for sliding, which gives us control over how configurations evolve. This leads to a simple greedy procedure that first slides tokens from A to B, and then from B back to A, until a canonical configuration is reached after a polynomial number of steps. This canonical configuration is determined purely by the exact positions of the tokens. We obtain a second canonical configuration by repeating the same procedure from J, and, again, reachability is decided by comparing the two.

Theorem 3. *There exists a polynomial-time algorithm to decide* Sliding Token *on threshold signed graphs.*

Beyond the specific polynomial-time algorithms presented here, we view the main conceptual contribution of this work as showing that the Dilworth decomposition of a graph into neighbourhood-inclusion chains can actually serve as a useful algorithmic tool for Independent Set Reconfiguration. We hope that this perspective will prove useful beyond threshold signed graphs, particularly for graph classes of bounded Dilworth number for Sliding Token and, more generally, in understanding the algorithmic role of neighbourhood-inclusion decompositions in graph reconfiguration in general.

Outline. Our paper is organised as follows. Section 2 introduces the notation and definitions used throughout the paper. Sections 3 and 4 present our algorithms for Token Jumping and Sliding Token, respectively. Finally, we conclude in Section 5.

2 Preliminaries

Basic notation. We write $[n]$ for $\{1, 2, \ldots, n\}$. All graphs in this paper are finite, simple, and undirected. For a graph $G = (V, E)$, we write $V(G)$ and $E(G)$ for its vertex and edge sets, respectively. We always refer to $|V(G)|$ as n. For a vertex $v \in V(G)$, the *open neighbourhood* of v is $\Gamma(v) = \{u \in V(G) : \{u, v\} \in E(G)\}$, and the *closed neighbourhood* is $\Gamma[v] = \Gamma(v) \cup \{v\}$. The degree of v is $|\Gamma(v)|$. For sets A, B, we write $A \triangle B = (A \setminus B) \cup (B \setminus A)$ for their symmetric difference. We refer to the subgraph of G induced on $S \subseteq V(G)$ as $G[S] = (S, E(G) \cap \binom{S}{2})$.

An *independent set* of G is a set of pairwise non-adjacent vertices. In the context of reconfiguration, we also refer to independent sets as *configurations*.

Definition 1 (Threshold signed graph)**.** A graph $G = (V, E)$ is a *threshold signed graph* if there exist a mapping $a : V \to \mathbb{R}$ and positive real constants $S, T > 0$ such that $|a(v)| < \min\{S, T\}$, for all distinct $u, v \in V$,

$$\{u, v\} \in E \quad \text{if and only if} \quad |a(u) + a(v)| \geq S \ \text{ or } \ |a(u) - a(v)| \geq T.$$

We next introduce a combinatorial characterisation of this class. For vertices $u, v \in V(G)$, we write $u \preceq v$ if $\Gamma(u) \subseteq \Gamma[v]$; here, we say that v *dominates* u.

Definition 2 (Dilworth-k graph, Dilworth number)**.** A graph G is a *Dilworth-k graph* if $V(G)$ can be partitioned into at most k chains, where a chain $(v_1, \ldots, v_t)$ satisfies

$$v_i \preceq v_j \qquad \text{for all } 1 \leq i < j \leq t.$$

The minimum such k is the *Dilworth number* of G, denoted $\mathcal{D}(G)$.

Theorem 4 (Benzaken et al. [BHdW85b])**.** *A graph is threshold signed if and only if it is a Dilworth-2 graph.*

Hence, every threshold signed graph admits a partition of its vertex set into two neighbourhood-inclusion chains (note that a chain can be empty); in fact, it turns out $V(G) = \{v : a(v) < 0\} \sqcup \{v : a(v) \geq 0\}$ is always a valid partition.

Input. Since such a partition can be computed in polynomial time [BHdW85a, Ful56], we simply assume throughout that it is given as part of the input; vertices with equal neighbourhoods in a chain can be ordered arbitrarily. We thus write $V(G) = A \sqcup B$, where A and B are the fixed partitions of the neighbourhood-inclusion chains. For $X \in \{A, B\}$ and $u, v \in V(G)$, we write $u <_X v$ if $u, v \in X$ and u precedes v in the fixed linear order of X. In particular, $u <_X v$ implies $\Gamma(u) \subseteq \Gamma[v]$.[6]

We then assume the input graph to be labelled with respect to this order, with $A = \{a_1, a_2, \ldots, a_{|A|}\}$ and $B = \{b_1, b_2, \ldots, b_{|B|}\}$, where $a_1 <_A a_2 <_A \cdots <_A a_{|A|}$ and $b_1 <_B b_2 <_B \cdots <_B b_{|B|}$ are the fixed orders. By pure abuse of notation, we often refer to these vertex partitions as *chains*.

Reconfiguration. We now introduce the reconfiguration terminology. We consider tokens to be uniquely labelled. Fix $k \in [n]$. A size-k independent set is interpreted as a configuration of k tokens placed on vertices of G, with no two tokens on adjacent vertices. A *reconfiguration move* either (i) moves a token to an unoccupied vertex (Token Jumping), or (ii) moves a token along an edge to an unoccupied neighbour (Sliding Token).

Definition 3 (k-token reconfiguration graph)**.** The *k-token reconfiguration graph* $\mathcal{R}_k(G)$ has as vertex set all independent sets of G of size k. Two configurations X, Y are adjacent in $\mathcal{R}_k(G)$ if and only if $X \triangle Y = \{u, v\}$ and moving the token from u to v is a valid reconfiguration move under the given rule in the context being used.

Since every move is clearly reversible, observe that reachability in $\mathcal{R}_k(G)$ induces an equivalence relation on the set of size-k independent sets. For configurations I and J, we write $I \leftrightsquigarrow J$ if they lie in the same connected component of $\mathcal{R}_k(G)$, equivalently if one can be transformed into the other via a finite sequence of valid moves. The *reconfiguration component* of I is defined as

$$\mathcal{C}(I) = \{J \mid I \leftrightsquigarrow J\}.$$

In other words, $\mathcal{C}(I)$ is the set of all independent sets reachable from I. For a configuration I and a token t on $v \in I$, we say that v is *occupied*; in addition, v is *unoccupied* when $v \notin I$. The token t on $v \in I$ is *I-movable* if there exists a configuration J adjacent to I in $\mathcal{R}_k(G)$ such that t occupies $u \neq v$.

3 On Token Jumping

This section is devoted to the proofs of Theorems 1 and 2. Our algorithm takes advantage of the fact that neighbourhoods are linearly ordered within each chain of the Dilworth-2 partition. In particular, tokens in each chain can be moved from right to left without violating independence (that is, creating adjacencies among tokens), since neighbourhoods can only decrease along the order. This is a highly useful property since it allows us to transform any configuration into a canonical form (i.e. a *unique* representative independent set in the reconfiguration component) for both I and J by first greedily moving tokens within each chain, and then transferring tokens between the two chains in a controlled manner. Reachability is then simply decided by comparing the resulting canonical configurations.

Section-specific terminology. Let $G = (V, E)$ be a threshold signed graph, and let $V(G) = A \sqcup B$, with $a_1 <_A a_2 <_A \cdots <_A a_{|A|}$ and $b_1 <_B b_2 <_B \cdots <_B b_{|B|}$, be a fixed partition into two neighbourhood-inclusion chains, as guaranteed by the Dilworth-2 property.

[6] Again, vertices that dominate each other are ordered arbitrarily, with this order fixed throughout.

Recall that vertices with equal neighbourhoods are ordered arbitrarily as long as that remains fixed. We always refer to the number of tokens as k.

For a chain $X \in \{A, B\}$ and an independent set I, let $\nabla_I(X) = |I \cap X|$ denote the number of tokens placed on X. For $u, v \in X$, we say that u is on the *left* of v when $u <_X v$; symmetrically, v is on the *right* of u. A *leftward move* in a chain X is a token move from an occupied vertex $v \in X$ to an unoccupied vertex $u \in X$ satisfying $u <_X v$. A *transfer* from a chain X to a chain Y is a token move from the rightmost occupied vertex $u \in X$ to the leftmost unoccupied vertex $v \in Y$ not adjacent to a vertex in $V(G) \setminus \{u\}$ occupied by a token.

An independent set is *left-normalised* if neither chain admits a leftward move of a token. A *left-normalisation* of I is the following procedure performed as long as possible: Find any token on a vertex u in some chain X, and move it to an unoccupied vertex $v \in X$ such that $v <_X u$. Let $L(I)$ be the resulting configuration; clearly, $\nabla_I(X) = \nabla_{L(I)}(X)$ for $X \in \{A, B\}$. Let $T_{B \to A}(I)$ denote the configuration obtained after *attempting* a single transfer from B to A from I; such a transfer may not be possible, in which case $I = T_{B \to A}(I)$.

Canonical configuration. We define the canonical configuration $\mathrm{can}(I)$ of I as follows:

$$\mathrm{can}(I) = \bigcup_{i=1}^{m(I)} \{a_i\} \cup \bigcup_{i=1}^{k-m(I)} \{b_i\}$$

where $m(I) = \max_{J \in \mathcal{C}(I)} \nabla_J(A)$. In other words, $\mathrm{can}(I)$ is the left-normalised configuration of a reachable configuration that maximises the number of tokens on the vertices of A.

Algorithm. The algorithm is quite easy: Starting from I, compute $L(I)$ by greedily moving tokens onto vertices on the left of their respective chains, as long as possible (this is the "flushing" idea). Then, from $L(I)$, as long as possible, perform transfers from B to A. The resulting configuration $\mathrm{can}(I)$ is then the unique canonical configuration of $\mathcal{C}(I)$. Repeat the same procedure from J to obtain $\mathrm{can}(J)$, and return "yes" if and only if $\mathrm{can}(I) = \mathrm{can}(J)$.

Algorithm 1: Token-Jumping-Reachability

Input: Threshold signed graph $G = (V, E)$ and independent sets I, J.

$I \leftarrow L(I)$ ▷ From I, left-normalise.
while $T_{B \to A}(I) \neq I$ **do**
  $I \leftarrow T_{B \to A}(I)$ ▷ Transfer as many tokens from B to A from $L(I)$ as possible.
$\mathrm{can}(I) \leftarrow I$ ▷ We obtain the canonical configuration.
$J \leftarrow L(J)$ ▷ The same procedure as above goes from J.
while $T_{B \to A}(J) \neq J$ **do**
  $J \leftarrow T_{B \to A}(J)$
$\mathrm{can}(J) \leftarrow J$
return $\mathrm{can}(I) = \mathrm{can}(J)$

Proof of correctness. First, we show that the left-normalisation procedure always successfully moves all t tokens on a chain to the first t leftmost vertices of that same chain while preserving independence.

Lemma 1. *Let $X \in \{A, B\}$. If $\nabla_I(X) = t$, then $L(I) \cap X = \bigcup_{i=1}^{t} \{x_i\}$.*

Proof. Suppose not; then some vertex x_i where $i \leq t$ is unoccupied, while some x_j with $j > t$ is occupied. Since $x_i <_X x_j$, we have $\Gamma(x_i) \subseteq \Gamma[x_j]$. Moving the token from x_j to x_i therefore preserves independence, since, otherwise, there would exist an occupied vertex $z \in \Gamma(x_i)$ such that $z \notin \Gamma[x_j]$, which is impossible. Hence, a leftward move is possible, which is a contradiction. □

Observation 1. *Let $I = L(I)$ be a left-normalised independent set, then $T_{B\to A}(I) = L(T_{B\to A}(I))$.*

Proof. If no token can be transferred in I, we then have $T_{B\to A}(I) = I$ and the claim clearly follows as $I = L(I)$, so assume otherwise. Suppose a transfer moves the rightmost token of B from x onto a vertex $v \in A$ such that there exists an unoccupied vertex $u <_A v$. Since $\Gamma(u) \subseteq \Gamma[v]$, moving the token from v to u preserves independence. However, this implies v is not the leftmost unoccupied vertex $z \in A$ not adjacent to an occupied vertex in $V(G) \setminus \{x\}$, a contradiction. □

In light of Lemma 1, the corollary below follows immediately.

Corollary 1. *For any k-vertex independent sets I, J, if $\nabla_I(A) = \nabla_J(A)$, then $L(I) = L(J)$.*

Corollary 1 implies that $\mathrm{can}(I)$ is well-defined and unique in each connected component of the reconfiguration graph. In the following lemma, we show that, after termination of transfers, we obtain the canonical configuration.

Lemma 2. *Let $I = L(I)$ be a left-normalised independent set. If there exists an independent set J such that $I \leftrightsquigarrow J$ and $\nabla_J(A) > \nabla_I(A)$, then $T_{B\to A}(I) \neq I$.*

Proof. Obviously, we have $I \cap B \neq \emptyset$. Consider one independent set reachable from I, say $I' \in \mathcal{C}(I)$, satisfying $\nabla_{I'}(A) = \nabla_I(A) + 1$. Such a configuration must exist given that any path $I, I_1, I_2, \ldots, I_m$ in $\mathcal{R}_k(G)$ where $I_m = J$ must include some $i \in [m]$ such that $\nabla_{I_i}(A) = \nabla_I(A) + 1$.

Now consider $L(I')$. Since we have $I \leftrightsquigarrow I' \leftrightsquigarrow L(I')$, it then follows that we have $I \leftrightsquigarrow L(I')$ by transitivity. By Lemma 1, $I \cap A = \{a_1, a_2, \ldots, a_{\nabla_I(A)}\}$, $I \cap B = \{b_1, b_2, \ldots, b_{k-\nabla_I(A)}\}$, $L(I') \cap A = \{a_1, a_2, \ldots, a_{\nabla_I(A)+1}\}$, and $L(I') \cap B = \{b_1, b_2, \ldots, b_{k-\nabla_I(A)-1}\}$. Hence, $I \triangle L(I') = \{b_{k-\nabla_I(A)}, a_{\nabla_I(A)+1}\}$. Since $L(I')$ is an independent set, $T_{B\to A}(I) = (I \setminus \{b_{k-\nabla_I(A)}\}) \sqcup \{a_{\nabla_I(A)+1}\} = L(I')$. Therefore, $T_{B\to A}(I) \neq I$ and the lemma holds. □

Lemma 3. *For a given independent set I, the algorithm correctly computes* $\mathrm{can}(I)$.

Proof. Suppose, for the sake of contradiction, that the algorithm terminates at I' immediately after transfers, even though there exists an independent set J with $J \leftrightsquigarrow I$ and $\nabla_J(A) > \nabla_{I'}(A)$.

Since the initial configuration I was first left-normalised and only transfers occurred up to I', $I' = L(I')$ by Observation 1. By Lemma 2, the existence of J implies we then have $T_{B\to A}(I') \neq I'$, which contradicts the definition of the algorithm. Thus I' maximises the number of tokens on A within $\mathcal{C}(I)$. Since we established $\nabla_{I'}(A) = \nabla_{\mathrm{can}(I)}(A)$, it remains to show that $I' = \mathrm{can}(I)$. First, observe that $\mathrm{can}(I) = L(\mathrm{can}(I))$ by definition. Since $I' = L(I')$, we then have $I' = L(I') = L(\mathrm{can}(I)) = \mathrm{can}(I)$ by Corollary 1. □

The correctness of the algorithm follows immediately.

Complexity. Note that the algorithm clearly runs in polynomial time. Indeed, left-normalisation can be done in $O(n)$ time by counting the tokens in each chain, and each transfer can be naïvely implemented to take $O(n)$ time. Since there are at most k transfers, the total running time is $O(nk)$. We thus obtain the following.

Theorem 1. *There exists a polynomial-time algorithm to decide* Token Jumping *on threshold signed graphs.*

Parameterised complexity. Interestingly, the "flushing" idea extends to graphs of arbitrary Dilworth number. Indeed, once a partition into neighbourhood-inclusion chains is fixed, a left-normalised configuration is uniquely determined by the number of tokens in each chain. We therefore represent these configurations by vectors of token counts and construct a graph whose edges correspond to legal token jumps between them. Since there are at most $(k+1)^d$ such vectors, where d is the number of chains, this graph can be constructed and its connectivity tested in $n^{O(d)}$ time, which gives the following.

Theorem 2. *There exists an $n^{O(d)}$-time algorithm to decide* Token Jumping *on arbitrary n-vertex graphs of Dilworth number d.*

Proof. Let $V(G) = X_1 \sqcup \cdots \sqcup X_d$ be a partition into neighbourhood-inclusion chains, where $x_{i,1} <_{X_i} \cdots <_{X_i} x_{i,|X_i|}$. Such a partition can be computed in polynomial time using, for example, the algorithm from Fulkerson [Ful56].

Let I, J be independent sets of size k. For any independent set S, define $\boldsymbol{\nabla}(S) = (\nabla_S(X_1), \ldots, \nabla_S(X_d))$ to be a vector representing the number of tokens in each chain. Observe that the proof of Lemma 1 not only applies to two chains, but to any number: indeed, $S \leftrightsquigarrow L(S)$ and $L(S) \cap X_i = \{x_{i,1}, \ldots, x_{i,\nabla_S(X_i)}\}$ for every $i \in [d]$; so we then have that $\boldsymbol{\nabla}(S) = \boldsymbol{\nabla}(S')$ implies $L(S) = L(S')$.

Now define a graph $\mathcal{H}_k$ whose vertices are the left-normalised independent sets of size k, that is, $V(\mathcal{H}_k) = \{L(S) : S \in \mathcal{R}_k(G)\}$. Add an edge between two vertices $S, S' \in V(\mathcal{H}_k)$ whenever there exist distinct i, j such that $\nabla_{S'}(X_i) = \nabla_S(X_i) - 1$, $\nabla_{S'}(X_j) = \nabla_S(X_j) + 1$, and $\nabla_{S'}(X_p) = \nabla_S(X_p)$ for every $p \notin \{i, j\}$. In other words, S' is then obtained from S by jumping the token at $x_{i,\nabla_S(X_i)}$ to $x_{j,\nabla_S(X_j)+1}$. Thus every edge of $\mathcal{H}_k$ is a valid token jump and, conversely, if $S \to S'$ is a valid token jump, then either $L(S) = L(S')$, when the jump stays within one chain, or $L(S)L(S') \in E(\mathcal{H}_k)$. It clearly follows from this that $I \leftrightsquigarrow J$ if and only if $L(I)$ and $L(J)$ belong to the same connected component of $\mathcal{H}_k$.

Observe that every vertex of $\mathcal{H}_k$ is uniquely determined by its vector $\boldsymbol{\nabla}(S)$. To bound the number of vertices, count that there are at most $\prod_{i=1}^{d}(\min\{|X_i|, k\}+1) \leq (k+1)^d$ candidate vectors. For each such candidate vector $\mathbf{c} = (c_1, \ldots, c_d)$ with $0 \leq c_i \leq \min\{|X_i|, k\}$ and $\sum_{i=1}^{d} c_i = k$, we test whether $\bigcup_{i=1}^{d}\{x_{i,1}, \ldots, x_{i,c_i}\}$ is an independent set, which can be done in $O(n^2)$ time. We then keep precisely those vectors for which this test succeeds. For each pair of vectors we keep, adjacency in $\mathcal{H}_k$ can be easily checked in $O(d)$ time. Since $k \leq n$, constructing $\mathcal{H}_k$ and testing whether $L(I)$ and $L(J)$ belong to the same connected component takes $n^{O(d)}$ time. □

4 On Sliding Token

For Sliding Token, moves are more constrained than in Token Jumping, since a token may only move along an edge of the graph. As a consequence, the mechanism presented in the previous section cannot be applied. However, the neighbourhood-inclusion structure of threshold signed graphs still allows us to define a canonical configuration that will then be used to decide reachability. Unsurprisingly, the correctness proofs are slightly more involved.

Section-specific terminology I. Let $G = (V, E)$ be a threshold signed graph, and let $V(G) = A \sqcup B$ be a fixed partition into two neighbourhood-inclusion chains, where $a_1 <_A a_2 <_A \cdots <_A a_{|A|}$ and $b_1 <_B b_2 <_B \cdots <_B b_{|B|}$. For a chain $X \in \{A, B\}$ and $u, v \in X$, we say that u is on the *left* of v when $u <_X v$; symmetrically, v is on the *right* of u. Denote by G_v the maximal connected component containing v in G and define $d_G(u, v)$ to be the length of a+ shortest path between u and v in G, where $d_G(u, v) = \infty$ if $G_u \neq G_v$. For a token occupying a vertex $v \in I$, let G_v^I and $M_I(v)$ be defined as:

$$G_v^I = G_v\left[V(G_v) \setminus \bigcup_{u \in I \setminus \{v\}} \Gamma[u]\right] \quad \text{and} \quad M_I(v) = \{u : u \in V(G_v^I) \wedge d_{G_v^I}(u, v) \neq \infty\}.$$

Equivalently, $u \in M_I(v)$ if and only if there exists a path $v = v_0, v_1, \ldots, v_t = u$ such that

$$\Gamma[v_i] \cap (I \setminus \{v\}) = \emptyset \qquad \text{for every } 0 \leq i \leq t.$$

In other words, the token on v can be slid onto u without moving any other token. Note that $v \in M_I(v)$ and all vertices in $M_I(v) \setminus \{v\}$ are unoccupied. Furthermore, notice the following, which follows from the reversibility of moves:

Observation 2. *For $u \in M_I(v)$, let $I_u = (I \setminus \{v\}) \cup \{u\}$. Then, $M_{I_u}(u) = M_I(v)$.*

A *left-normalisation* of a configuration I consists of repeatedly performing the following as long as possible: Choose a chain $X \in \{A, B\}$ where $|X \cap I| \geq 1$ and let $v = \max_{<_X}(I \cap X)$ be its rightmost occupied vertex. Then, slide the token on v onto $u = \min_{<_X}(M_I(v) \cap X)$ along a path $v = v_0, v_1, v_2, \ldots, v_t = u$ in G satisfying $u <_X v$ and $\Gamma[v_i] \cap (I \setminus \{v\}) = \emptyset$ for all $i \in [t]$; i.e. without moving any other token. Such a path exists by the definition of $M_I(v)$. If both chains admit such a move, perform it on A. Let $I \leftarrow (I \setminus \{v\}) \cup \{u\}$, and repeat until no such move can be performed on either chain. Define $L(I)$ to be the configuration resulting from the left-normalisation of I. Since the selection of the chain and the destination of each move are uniquely determined, $L(I)$ is well-defined.

For distinct $X, Y \in \{A, B\}$, a *transfer* from X to Y of a configuration I, where $|X \cap I| \geq 1$ and $M_I(v) \cap Y \neq \emptyset$, consists of sliding the token on $v = \max_{<_X}(I \cap X)$ onto $u = \min_{<_Y}(M_I(v) \cap Y)$ along a path $v = v_0, v_1, v_2, \ldots, v_t = u$ in G satisfying $\Gamma[v_i] \cap (I \setminus \{v\}) = \emptyset$ for all $i \in [t]$. Informally, it consists of sliding the rightmost token of X onto the leftmost unoccupied vertex in Y reachable without moving any other token. Define $T_{X \to Y}(I) = (I \setminus \{v\}) \cup \{u\}$ to be the resulting configuration. For convenience, if no such transfer is possible in I, let $T_{X \to Y}(I) = I$. Again, since the destination is uniquely determined, $T_{X \to Y}(I)$ is well-defined.

Canonical configuration. Assign weights to the vertices by setting $w(a_i) = 2^{n-i+1}$ and $w(b_i) = 2^{|B|-i+1}$. We define the canonical configuration $\mathrm{can}(I)$ as follows:

$$\mathrm{can}(I) = \arg\max_{I \leftrightsquigarrow J} W(J),$$

where $W(I) = \sum_{v \in I} w(v)$. Observe that since the weights are powers of 2, distinct configurations have distinct weights, and hence the canonical configuration is unique and well-defined for each connected component of the reconfiguration graph.

Algorithm. Starting from S, the algorithm repeatedly performs a full pass as follows. First, it left-normalises the current configuration. It then performs transfers from A to B as long as possible, left-normalising after each transfer. Once no further transfer from A to B is possible, it performs transfers from B to A in the same way, again left-normalising after each transfer. The algorithm terminates when the configuration at the end of a full pass is equal to the configuration at its beginning, and returns the resulting configuration. We prove below that this configuration is $\mathrm{can}(S)$.

Algorithm 2: SLIDING-TOKEN-REACHABILITY

Input: A threshold signed graph $G = (V, E)$ and two independent sets I and J.

Function `Canonical`(S)
  repeat
    $S_{\text{start}} \leftarrow S$
    $S \leftarrow L(S)$
    while $T_{A\to B}(S) \neq S$ **do**
      $S \leftarrow L(T_{A\to B}(S))$
    while $T_{B\to A}(S) \neq S$ **do**
      $S \leftarrow L(T_{B\to A}(S))$
  until $S_{start} = S$
  return S
$\mathrm{can}(I) \leftarrow$ `Canonical`(I)
$\mathrm{can}(J) \leftarrow$ `Canonical`(J)
return $\mathrm{can}(I) = \mathrm{can}(J)$

Proof of correctness. To give a bird's-eye view before diving into the technicalities, the proof proceeds in three steps. First, we establish some structural restrictions imposed by the two neighbourhood-inclusion chains. In particular, only the rightmost token of each chain can slide, and tokens also preserve their relative order. Second, we show that, between the beginning and the end of a full loop of `Canonical`, the tokens are always placed on vertices whose weights are not lower. As a consequence, every loop that changes the configuration strictly increases its total weight computed by W. Finally, we prove that a configuration left unchanged by a full loop must be canonical. Indeed, this can be shown by contradiction: Suppose otherwise, and consider a reconfiguration sequence to a configuration of larger total weight. Relative to the starting configuration, consider the first move that places a token on a vertex of strictly greater weight than its initial vertex. We show that the existence of such a move implies that a full loop of `Canonical` would change the starting configuration, contradicting our assumption.

In what follows, we show that in any configuration, only the rightmost token of a chain can be slid, and when a token is slid onto another chain, it becomes its rightmost token.

Lemma 4. *Let I, I' be adjacent configurations in $\mathcal{R}_k(G)$, and write $I \setminus I' = \{v\}$. If $v \in X$, where $X \in \{A, B\}$, then $v = \max_{<_X}(I \cap X)$.*

Proof. Let $I' \setminus I = \{u\}$. Since the move is a token slide, $u \in \Gamma(v)$. Suppose there exists $w \in I \cap X$ with $v <_X w$. Then $\Gamma(v) \subseteq \Gamma[w]$, so $u \in \Gamma[w]$. Since $u \notin I$, we have $u \neq w$, and hence $\{u, w\} \in E(G)$. But we then have $u, w \in I'$, which is a contradiction given that I' is an independent set. □

Corollary 2. *Let I, I' be adjacent configurations in $\mathcal{R}_k(G)$, and write $I' \setminus I = \{u\}$. If $u \in X$, where $X \in \{A, B\}$, then $u = \max_{<_X}(I' \cap X)$.*

Proof. Simply apply Lemma 4 to the reverse slide from I' to I. □

As previously mentioned, the above observation reveals an elegant invariant: At any point in the reconfiguration, only the rightmost token of each chain can be moved. Furthermore, after a token is transferred into a chain, that token becomes the unique movable token in its new chain. Thus, the set of possible token slides is highly constrained. In particular, the vertices admit a natural linear order given by $a_1, a_2, \ldots, a_{|A|}, b_{|B|}, b_{|B|-1}, \ldots, b_1$, under which at most two tokens are movable at any time and no token can overtake another token within this order.[7] Observe that, beyond threshold signed graphs, this property is true for any arbitrary graph H: for any configuration of H, only at most $\mathcal{D}(H)$ tokens are movable, and these tokens are always the rightmost of their respective chain. We will come back to this observation in the conclusion.

In the proofs that follow, we show that moving a token to the right within its chain cannot enable a slide of another token that was previously impossible.

Lemma 5. *Let I_1 be an independent set, and let $u \in I_1$ and $v \notin I_1$ belong to the same chain with $u <_X v$. Suppose that $I_2 = (I_1 \setminus \{u\}) \cup \{v\}$ is independent. Then*

$$M_{I_2}(x) \subseteq M_{I_1}(x) \qquad \text{for every } x \in I_1 \setminus \{u\}.$$

Proof. Fix $x \in I_1 \setminus \{u\}$ ($x \in I_2$ since $I_1 \triangle I_2 = \{u, v\}$) and $y \in M_{I_2}(x)$. By definition, there exists a path $x = v_0, v_1, \ldots, v_t = y$ such that for all $i \in [t]$, $\Gamma[v_i] \cap (I_2 \setminus \{x\}) = \emptyset$. Call that path P. Since $v \in I_2$, we have $w \notin P$ for all $w \in \Gamma[v]$. Furthermore, $\Gamma(u) \subseteq \Gamma[v]$ by $u <_X v$, hence $w \notin P$ for all $w \in \Gamma(u)$. We also have $u \notin P$ since $u \neq \max_{<_X}(I_2 \cap X)$ by Corollary 2; otherwise, since $x \neq u$, some slide along P would enter u while v remains occupied. As $u <_X v$, this contradicts Corollary 2. Therefore, since $I_1 \triangle I_2 = \{u, v\}$, $\Gamma[w] \cap (I_1 \setminus \{x\}) = \emptyset$ for all $w \in P$, thus $y \in M_{I_1}(x)$ and the lemma follows. □

Lemma 6. *Let I be an independent set, let $X, Y \in \{A, B\}$ be distinct chains, and let $v = \max_{<_X}(I \cap X)$. Suppose that $M_I(v) \cap Y \neq \emptyset$, and put $u = \min_{<_Y}(M_I(v) \cap Y)$. For any $w \in M_I(v) \cap Y$, define $I_u = (I \setminus \{v\}) \cup \{u\}$ and $I_w = (I \setminus \{v\}) \cup \{w\}$. Then*

$$M_{I_w}(x) \subseteq M_{I_u}(x) \qquad \text{for every } x \in I \setminus \{v\}.$$

Proof. Both I_u and I_w are independent by the definition of $M_I(v)$. If $u = w$, there is nothing to prove. Otherwise, $u <_Y w$, and the claim follows from Lemma 5, since $I_w = (I_u \setminus \{u\}) \cup \{w\}$. □

The previous claims motivate the two operations used by our algorithm: left-normalisation and transfers between chains. Moving a token further to the left within its chain cannot reduce the reachable set of any other token while the remaining tokens are fixed. Similarly, when moving a token from one chain to another while keeping all other tokens fixed, choosing the leftmost reachable destination preserves every single-token movement available to the other tokens after any alternative choice of destination on that same chain.

[7] More generally, for any incomparability graph, sliding a token preserves the relative order of tokens in any fixed linear extension of a poset representing the graph.

Section-specific terminology II. To analyse the algorithm, we introduce the following terminology. We write $\pi_i(S)$ for the vertex occupied by token i in S. An $A \to B$ *half-pass* repeatedly performs a transfer from A to B, followed immediately by left-normalisation, until no further such transfer is possible. Thus, each successful step replaces the current configuration S by $L(T_{A\to B}(S))$. A $B \to A$ half-pass is defined symmetrically. A *pass* first left-normalises the current configuration, then performs an $A \to B$ half-pass followed by a $B \to A$ half-pass. Define $P(I)$ and $H_{X\to Y}(I)$ to be the configurations resulting from a pass and an $X \to Y$ half-pass starting from I, respectively. An *execution sequence* of a pass or half-pass from I is the sequence $I = I_0, I_1, \ldots, I_r$ obtained by recording the initial configuration and the configuration after each completed left-normalisation or transfer. Intermediate configurations along the slide paths which implement these operations are omitted; that is, we have $I_i = T_{X\to Y}(I_{i-1})$ or $I_i = L(I_{i-1})$ for $X \in \{A, B\}$ and all $i \in [r]$. The final configuration is $P(I)$ or $H_{X\to Y}(I)$, respectively. A token occupying $v \in I$ is *I-exposed* if $v = \max_{<_X}(I \cap X)$ for $X \in \{A, B\}$. A pass or half-pass *exposes* a token if that token is exposed in at least one configuration of its execution sequence.

Fix a reference configuration I and label each token by the vertex it occupies in I. A token initially at $x \in I$ is *improved relative to I* in a configuration I' if its position y in I' satisfies $w(y) > w(x)$. Equivalently, either x, y belong to the same chain X and $y <_X x$, or $x \in B$ and $y \in A$. A configuration I' is *I-improving* if at least one token is improved relative to I.

In the next lemma, we formally establish the easy observation that moving a token as far to the left in another chain always maximises the weight of the current configuration.

Lemma 7. *Let I be an independent set and let $X, Y \in \{A, B\}$ be distinct. Suppose $I \cap X \neq \emptyset$, and put $v = \max_{<_X}(I \cap X)$. If $M_I(v) \cap Y \neq \emptyset$, then, for every $u \in M_I(v) \cap Y$,*

$$W\big(T_{X\to Y}(I)\big) \geq W\big((I \setminus \{v\}) \cup \{u\}\big),$$

with equality if and only if $u = \min_{<_Y}(M_I(v) \cap Y)$.

Proof. Let $z = \min_{<_Y}(M_I(v) \cap Y)$. Since vertex weights strictly decrease along Y, $w(z) \geq w(u)$, with equality if and only if $z = u$. Consequently, $W\big(T_{X\to Y}(I)\big) = W(I) - w(v) + w(z) \geq W(I) - w(v) + w(u)$, so the lemma holds. □

Our next lemma shows that the set of vertices reachable by an exposed token, with all other tokens fixed, is actually only determined by at most two other tokens: the token immediately to its left in its chain and the rightmost token of the other chain, when they exist. Moreover, if these two neighbouring tokens occupy vertices no farther right in their respective chains in another independent set, while keeping the exposed token at the same vertex, its reachable set still cannot decrease. This property will come in handy in the proof of Theorem 5 to show that the successive slides of a token in a hypothetical reconfiguration sequence remain possible with the other tokens fixed at their positions in a particular configuration of the pass.

Lemma 8. *Let I, I' be independent sets and let $X, Y \in \{A, B\}$ be distinct. Suppose that, for each $S \in \{I, I'\}$, $|S \cap X| \geq 2$, $|S \cap Y| \geq 1$, and $v = \max_{<_X}(S \cap X)$. Let $l = \max_{<_X}((I \setminus \{v\}) \cap X)$, $r = \max_{<_Y}(I \cap Y)$, $l' = \max_{<_X}((I' \setminus \{v\}) \cap X)$, and $r' = \max_{<_Y}(I' \cap Y)$. If $l' \leq_X l$ and $r' \leq_Y r$, then $M_I(v) \subseteq M_{I'}(v)$.*

Proof. Let $J = \{v, l, r\}$ and $J' = \{v, l', r'\}$. For every $z \in I \setminus J$, either $\Gamma(z) \subseteq \Gamma[l]$ or $\Gamma(z) \subseteq \Gamma[r]$. Thus $M_J(v)$ contains no neighbour of z and cannot contain z itself, since $z \neq v$, $z <_X w$ for some $X \in \{A, B\}$ and $w \in \{l, r\}$, which contradicts Corollary 2. Hence $M_I(v) = M_J(v)$. Similarly, $M_{I'}(v) = M_{J'}(v)$. Applying Lemma 5 first to $l' \leq_X l$ and then to $r' \leq_Y r$ gives $M_J(v) \subseteq M_{J'}(v)$, and the claim holds. □

Lemma 8 also holds when $I \cap X = I' \cap X = \{v\}$ or $I \cap Y = I' \cap Y = \emptyset$. In these cases, omit l, l' or r, r', respectively, together with the corresponding inequality. This does not affect the proof.

Lemma 9. *Let $I' = P(I)$. For each $u \in I$, let $v \in I'$ be the vertex occupied by the token initially at u. Then:*

- *If $u \in A$, then $v \in A$ and $v \leq_A u$.*
- *If $u \in B$, then either $v \in A$, or $v \in B$ and $v \leq_B u$.*

Here, $v \leq_X u$ means $v = u$ or $v <_X u$.

Proof. Label the tokens $1, \dots, k$ according to their order on $a_1, \dots, a_{|A|}, b_{|B|}, \dots, b_1$. By Lemma 4 and Corollary 2, this order is invariant. Let $m = |I \cap A|$, and suppose that $H_{A\to B}(L(I))$ transfers tokens $m, \dots, \ell + 1$. For $\ell < i \leq m$, let d_i be the vertex occupied by token i immediately before its transfer; thus $d_i \leq_A \pi_i(I)$. Set $D_0 = H_{A\to B}(L(I))$ and $D_q = (L \circ T_{B\to A})^q(D_0)$.[8] Let $S_0, \dots, S_r$ be the execution sequence of $H_{A\to B}(L(I))$. For $\ell < t \leq m$, define $h_t = \max\{h \in \{0, \dots, r\} : \pi_t(S_h) \in A\}$ and $C_t = S_{h_t}$. In other words, C_t is the last configuration of the sequence where token t occupies a vertex in A; thus, we have $\pi_t(C_t) = d_t$. We prove by induction on $q = 0, \dots, m - \ell$ that $|D_q \cap A| = \ell + q$ and that $\pi_j(D_q) \in A$ and $\pi_j(D_q) \leq_A d_j$ for every $\ell < j \leq \ell + q$. For $q = 0$, there is nothing to prove since the equality follows by definition and the condition on j is vacuous anyway. If $m = \ell$, we are done. So consider $q = 1$ and let $E = T_{A\to B}(C_{\ell+1})$.

By reversibility, we get $d_{\ell+1} \in M_E(\pi_{\ell+1}(E))$. Since this is the last $A \to B$ transfer, $D_0 = L(E)$. During this left-normalisation, only tokens ℓ (if $\ell > 0$) and $\ell + 1$ can change position. If $\ell > 0$, a leftward move of token ℓ onto $z \leq_A \pi_\ell(E)$ preserves $d_{\ell+1} \in M_S(\pi_{\ell+1}(S))$ for $S = (E \setminus \{\pi_\ell(E)\}) \cup \{z\}$ by Lemma 5. A leftward move of token $\ell + 1$ also preserves it: for arbitrary $S \subseteq V(G)$, if $S' = (S \setminus \{x\}) \cup \{y\}$ with $x = \pi_{\ell+1}(S)$ and $y \in M_S(x)$, then clearly $M_{S'}(y) = M_S(x)$ by Observation 2. Consequently, $d_{\ell+1} \in M_{D_0}(\pi_{\ell+1}(D_0))$. Thus $T_{B\to A}(D_0) \neq D_0$, $|D_1 \cap A| = \ell + 1$, and $\pi_{\ell+1}(D_1) \leq_A d_{\ell+1}$.

Now let $1 < q \leq m - \ell$, assume the claim for $q-1$, and write $i = \ell + q$. Let $E = T_{A\to B}(C_i)$, $F = C_{i-1}$, and $R = T_{B\to A}(D_{q-2})$; thus $D_{q-1} = L(R)$. By reversibility, $d_i \in M_E(\pi_i(E))$. Between E and F, only left-normalisation is performed. As in the case $q = 1$, Lemma 5 and Observation 2 therefore give $d_i \in M_F(\pi_i(F))$. Between F and R, token i is unchanged: after the transfer of token $i - 1$ to B, token i is not exposed until token $i - 1$ occupies a vertex in A. Hence $\pi_i(R) = \pi_i(F)$. Moreover, $\pi_{i-1}(F) = d_{i-1}$ and, by the previous induction step, we have $\pi_{i-1}(R) \leq_A d_{i-1}$. If $i < k$, token $i + 1$ is unchanged between F and R, so $\pi_{i+1}(R) = \pi_{i+1}(F)$. In both F and R, token $i-1$ is rightmost in A and token $i+1$, if present, is the rightmost token of B other than token i. Lemma 8 then gives $M_F(\pi_i(F)) \subseteq M_R(\pi_i(R))$. Thus $d_i \in M_R(\pi_i(R))$. Applying Lemma 5 and Observation 2 for $D_{q-1} = L(R)$ gives $d_i \in M_{D_{q-1}}(\pi_i(D_{q-1}))$. Therefore $T_{B\to A}(D_{q-1}) \neq D_{q-1}$ and $\pi_i(T_{B\to A}(D_{q-1})) \leq_A d_i$. After left-normalisation, $|D_q \cap A| = \ell + q$ and $\pi_i(D_q) \leq_A d_i$. For $\ell < j < i$, we also have $\pi_j(D_q) \leq_A \pi_j(D_{q-1}) \leq_A d_j$, which completes our induction.

Finally, now that we have established that a token occupying a vertex $v \in A$ at I occupies a vertex $u \in A$ such that $u \leq_A v$ at $P(I)$, it suffices to show that a token occupying a vertex $v \in B$ at I which occupies a vertex $u \in B$ at $P(I)$ satisfies $u \leq_B v$. In fact, observe that such a token (call it t) can only be moved by left-normalisation. Indeed, otherwise a $B \to A$ half-pass would have placed token t onto A, and $\pi_t(P(I)) \in A$, a contradiction. By definition of left-normalisation, the claim holds, which completes the proof. □

[8] We write $f^q(x) = (\underbrace{f \circ f \circ \cdots \circ f}_{q \text{ times}})(x)$.

Notice that the proof of Lemma 9 establishes the following stronger property: For every token initially in A, the subsequence of vertices of A occupied by that token during the execution sequence of the pass is non-increasing with respect to $\leq_A$. Likewise, for every token initially in B, the subsequence of vertices of B occupied by that token is non-increasing with respect to $\leq_B$. And lastly, once such a token is transferred to A, it does not return to B during the same pass. More formally:

Corollary 3. *Let $I = \Delta_0, \Delta_1, \ldots, \Delta_r = P(I)$ be the execution sequence of a pass. If $P(I) = I$, then $L(I) = I$ and, for every $i \in [k]$, $q \in \{0, \ldots, r\}$, and $X \in \{A, B\}$, $\pi_i(I), \pi_i(\Delta_q) \in X$ implies $\pi_i(\Delta_q) = \pi_i(I)$.*

Corollary 4. *Let $I' = P(I)$. Then $W(I') \geq W(I)$, with equality if and only if $I' = I$.*

The corollary above and the invariant relative order of the tokens further lead us to the following crucial lemma.

Lemma 10. *Let I, J be independent sets of size k, with tokens labelled in their relative order. If $W(J) > W(I)$, then at least one of the following holds:*

1. *There exist $i \in [k]$ and $X \in \{A, B\}$ such that $\pi_i(I), \pi_i(J) \in X$ and $\pi_i(J) <_X \pi_i(I)$.*
2. *There exists $i \in [k]$ such that $\pi_i(I) \in B$ and $\pi_i(J) \in A$.*

Proof. If neither holds, then $w(\pi_i(J)) \leq w(\pi_i(I))$ all $i \in [k]$, which contradicts $W(J) > W(I)$. □

We are now finally ready to prove that our algorithm correctly computes the canonical configuration of an arbitrary configuration.

Lemma 11. *Let $I \neq \text{can}(I)$, and let $I = S_0, S_1, \ldots, S_r = \text{can}(I)$ be any reconfiguration sequence. Let $h = \min\{a : S_a \text{ is } I\text{-improving}\}$, and let t be the token moved by $S_{h-1} \to S_h$. If $\pi_t(I) \in X$, where $X \in \{A, B\}$, then the execution sequence of the pass from I contains a configuration D such that $\pi_t(D) = \max_{<_X}(D \cap X)$, that is, token t is D-exposed in X.*

Proof. As usual, label the tokens $1, \ldots, k$ according to their order on $a_1, \ldots, a_{|A|}, b_{|B|}, \ldots, b_1$. If $\pi_t(I) \in B$, no token initially in B enters A before $S_{h-1} \to S_h$, since such a move would be I-improving by definition. Therefore, token t is I-exposed in B and the claim trivially follows. So suppose we have $\pi_t(I) \in A$ instead and consider the following claim.

Claim 1. *Let $n_A = |I \cap A|$ and $D_s = (L \circ T_{A \to B})^s(L(I))$. For $0 \leq s \leq n_A - t$, we have $|D_s \cap A| = n_A - s$ and, for every $n_A - s < p \leq k$ and $0 \leq a < h$, $\pi_p(S_a) \in B$ implies $\pi_p(D_s) \leq_B \pi_p(S_a)$.*

Proof. We prove this by induction on $s = 0, 1, \ldots, n_A - t$. Write $a_p = \pi_p(I)$. For $s = 0$, $|D_0 \cap A| = n_A$ and $\pi_p(D_0) \leq_B a_p$ for $p > n_A$. Since no configuration S_a with $a < h$ is I-improving, $\pi_p(S_a) \in B$ implies $a_p \leq_B \pi_p(S_a)$. Thus the invariant holds for $s = 0$. Now, suppose it holds for $s < n_A - t$, and let $j = n_A - s$. Token j is D_s-exposed in A. Before first becoming exposed during the half-pass, it stays at a_j; subsequent leftward moves preserve $a_j \in M_{D_s}(\pi_j(D_s))$ by Lemma 5 and Observation 2. Whenever token j slides in $S_0, \ldots, S_{h-1}$, token $j - 1$ belongs to A and satisfies $\pi_{j-1}(D_s) = a_{j-1} \leq_A \pi_{j-1}(S_a)$. Token $j + 1$ belongs to B and satisfies $\pi_{j+1}(D_s) \leq_B \pi_{j+1}(S_a)$ by the induction hypothesis.

Starting from $a_j \in M_{D_s}(\pi_j(D_s))$, let us step aside for a moment and prove by induction on $0 \leq a < h$ that $\pi_j(S_a) \in M_{D_s}(\pi_j(D_s))$. The base case is trivial since we already have $\pi_j(S_0) = a_j$ by definition. For the induction step, suppose $a + 1 < h$. If the slide $S_a \to S_{a+1}$

does not move token j, the claim is unchanged; otherwise, let $Q_a = (D_s \setminus \{\pi_j(D_s)\}) \cup \{\pi_j(S_a)\}$. By the induction hypothesis, Q_a is an independent set, and Observation 2 gives $M_{Q_a}(\pi_j(S_a)) = M_{D_s}(\pi_j(D_s))$. Since token j slides, it is S_a-exposed. The inequalities we established above imply that it is also Q_a-exposed and that Lemma 8 applies, which gives us $M_{S_a}(\pi_j(S_a)) \subseteq M_{Q_a}(\pi_j(S_a))$. Since $S_a \to S_{a+1}$ slides token j, we have $\pi_j(S_{a+1}) \in M_{S_a}(\pi_j(S_a))$. Thus $\pi_j(S_{a+1}) \in M_{D_s}(\pi_j(D_s))$, which completes the nested induction.

Now back to our original claim, note that absent neighbouring tokens (i.e. $j-1=0$ or $j+1=k+1$) do not affect the argument in any way, so they can be safely omitted. Since $\pi_t(I) \in A$ and token t is improved in S_h relative to I, we have $\pi_t(S_h) \in A$. By Corollary 2, token t is S_h-exposed in A. Hence, for every $j > t$, $\pi_j(S_{h-1}) = \pi_j(S_h) \in B$. Consequently, $T_{A\to B}(D_s) \neq D_s$, which gives, by its definition (as well as the left-normalisation) $\pi_j(D_{s+1}) \leq_B \pi_j(S_a)$ whenever $a < h$ and $\pi_j(S_a) \in B$. Tokens $p > j$ remain fixed during this transfer and the subsequent left-normalisation, since they are not $T_{A\to B}(D_s)$-exposed. Hence we have $|D_{s+1} \cap A| = n_A - s - 1$ and all required inequalities hold, which completes our induction. □

Finally, the lemma follows from Claim 1 by taking $s = n_A - t$, which completes the proof. □

Theorem 5. *If $I = P(I)$, then $I = \mathrm{can}(I)$.*

Proof. For the sake of contradiction, suppose that $I = P(I)$ but $I \neq \mathrm{can}(I)$. Consider a reconfiguration sequence $I = S_0, S_1, \ldots, S_r = \mathrm{can}(I)$, with tokens labelled according to their order on $a_1, \ldots, a_{|A|}, b_{|B|}, \ldots, b_1$. Since $W(\mathrm{can}(I)) > W(I)$, Lemma 10 implies that $h = \min\{j : S_j \text{ is } I\text{-improving}\}$ exists. Let $E = S_{h-1}$ and $F = S_h$, and write $E \setminus F = \{u\}$ and $F \setminus E = \{v\}$. Let t be the token moved by $E \to F$. Then $\pi_t(E) = u$, $\pi_t(F) = v$, and $w(v) > w(\pi_t(I))$, whereas $w(\pi_j(S_s)) \leq w(\pi_j(I))$ for every token j and every $0 \leq s < h$. Let $a = \pi_t(I)$ and $n_A = |I \cap A|$. We now distinguish three cases.

Case 1: $a, v \in A$ and $v <_A a$. By Lemma 11 and Claim 1, token t is $D = (L \circ T_{A\to B})^{n_A - t}(I)$-exposed in A. By $P(I) = I$, Corollary 3 and Lemma 9, $\pi_t(D) = a$; moreover, $L(D) = D$. We prove $\pi_t(S_s) \in M_D(a)$ by induction on $s = 0, \ldots, h$. This is obviously true for $s = 0$ by not moving the token at all. Now suppose it holds for $s-1$. If the slide moves a token other than t, then $\pi_t(S_s) = \pi_t(S_{s-1}) \in M_D(a)$ by induction. Otherwise, let $x = \pi_t(S_{s-1})$, $y = \pi_t(S_s)$, and $Q = (D \setminus \{a\}) \cup \{x\}$. By induction and Observation 2, Q is an independent set and $M_Q(x) = M_D(a)$. Since token t slides, token $t-1$, if $1 \leq t-1$, occupies a vertex in A at S_{s-1} and satisfies $\pi_{t-1}(D) = \pi_{t-1}(I) \leq_A \pi_{t-1}(S_{s-1})$. Token $t+1$, if $t+1 \leq k$, occupies a vertex in B and satisfies $\pi_{t+1}(D) \leq_B \pi_{t+1}(S_{s-1})$ by Claim 1. Lemma 4 and those inequalities imply that token t is Q-exposed. Lemma 8 (with absent neighbouring tokens omitted if it applies) therefore gives us $M_{S_{s-1}}(x) \subseteq M_Q(x)$. Since $y \in M_{S_{s-1}}(x)$, we obtain $y \in M_Q(x) = M_D(a)$, which completes our induction. We hence have $v = \pi_t(S_h) \in M_D(a)$, but since we also have $v <_A a$, this contradicts $L(D) = D$.

Case 2: $a \in B$ and $v \in A$. Since no token initially in B enters A before $E \to F$, the invariant token order gives $t = n_A + 1$. Whenever token t slides before or at $E \to F$, every token $j < t$ belongs to A and satisfies $\pi_j(I) \leq_A \pi_j(S_{s-1})$, while every token $j > t$ belongs to B and satisfies $\pi_j(I) \leq_B \pi_j(S_{s-1})$. The induction from Case 1, with $D = I$ instead, therefore gives $v \in M_I(a)$. Since $P(I) = I$, Lemma 9 and Corollary 3 imply that after all tokens transferred from A have returned and left-normalisation is complete, the configuration is I. But $v \in M_I(a) \cap A$ implies $T_{B\to A}(I) \neq I$. Thus token t is transferred to A and remains there until the end of the pass, which contradicts $P(I) = I$.

Case 3: $a, v \in B$ and $v <_B a$. As in Case 2, $t = n_A + 1$ and $v \in M_I(a)$. Thus token t is I-exposed in B and $v \in M_I(a)$, contradicting $L(I) = I$. □

The correctness of the algorithm follows effortlessly. Let us now discuss the time complexity.

Complexity. For a configuration S and $v \in S$, the set $M_S(v)$ can be computed in $O(n^2)$ time by deleting $\bigcup_{u \in S \setminus \{v\}} \Gamma[u]$ and computing the connected component containing v. During a left-normalisation, only the rightmost token of each chain can change position, and these two tokens remain rightmost before and after each operation. Each completed leftward move strictly decreases the position of one of them. Hence a left-normalisation requires $O(n)$ reachable-set computations and takes $O(n^3)$ time. Each half-pass contains at most k transfers, so a full pass takes $O(kn^3)$ time.

To bound the number of passes, define $\rho(a_i) = n - i + 1$, $\rho(b_j) = |B| - j + 1$, and $\Phi(S) = \sum_{v \in S} \rho(v)$. By Lemma 9, $\Phi(P(S)) \geq \Phi(S) + 1$ whenever $P(S) \neq S$. Since Φ is an integer between k and kn, there are $O(kn)$ passes that change the configuration, followed by one pass that leaves it unchanged, which is canonical by Theorem 5. So `Canonical` runs in $O(k^2n^4)$ time, assuming, of course, that we are given the ordered chain partition (which can be computed in polynomial time regardless [Ful56]). We then get the result below.

Theorem 3. *There exists a polynomial-time algorithm to decide* Sliding Token *on threshold signed graphs.*

Parameterised complexity. It remains elusive for now whether Sliding Token has an XP algorithm parameterised by the Dilworth number. The vector of token counts in chains we used for Token Jumping does not extend directly because, under Sliding Token, left-normalising two configurations with the same number of tokens in each chain need not produce the same configuration. So chain counts alone do not determine a normalised configuration. We leave this question for future work.

5 Concluding Remarks

We introduced a new perspective on Independent Set Reconfiguration by exploiting the neighbourhood-inclusion chains of a graph. This led us to polynomial-time algorithms for both Token Jumping and Sliding Token, which resolves these problems for a non-trivial subclass of permutation graphs.

Our results raise several natural questions. The most immediate one is whether the chain decomposition used here can be turned into a more general algorithmic framework.

Question 1. *Are* Token Jumping *and* Sliding Token *fixed-parameter tractable parameterised by the Dilworth number $\mathcal{D}(G)$? That is, does each problem admit an algorithm running in $f(\mathcal{D}(G))n^{O(1)}$ time for some computable function f?*

On the other hand, it is conceivable that Sliding Token turns out to be PSPACE-complete already on graphs with Dilworth number three. For broader classes of permutation graphs, it is unclear whether similar ordering-based arguments can capture the structure of their reconfiguration graphs. For Sliding Token, we observed in Section 4 that only the rightmost token of each chain can move at any time (which, by the way, extends to any number of chains). This suggests viewing the problem as a reconfiguration "puzzle" on stacks, where only the top element of each stack is accessible.

Another natural question concerns the length of the sequences produced by our algorithm. Although canonicalisation allows us to decide reachability, a sequence passing through the canonical configuration may not be shortest.

Question 2. *Given two independent sets of a threshold signed graph that are reachable from one another under* Sliding Token*, can a shortest reconfiguration sequence between them be computed in polynomial time?*

It would also be interesting to substantially improve the daunting $O(k^2n^4)$ running time of our Sliding Token algorithm. But more generally, we ask whether decompositions of permutation graphs into neighbourhood-inclusion chains can reveal nice algorithmic structure. Such decompositions may provide a route towards resolving the conjecture of Briański et al. [BFHM21] that Sliding Token is tractable on permutation graphs. Our work suggests that understanding which properties of these decompositions allow efficient reconfiguration algorithms is a promising direction for further research.

Acknowledgements. We thank the anonymous reviewers of an earlier version of this paper for their helpful feedback. This work was supported by EPSRC grant EP/X039447/1.

References

[AKKO24] Yuuki Aoike, Masashi Kiyomi, Yasuaki Kobayashi, and Yota Otachi. Finding a reconfiguration sequence between longest increasing subsequences. *IEICE TRANSACTIONS on Information and Systems*, 107(4):559–563, 2024.

[AU16] Hoang Duc Anh and Ryuhei Uehara. Sliding tokens on a cactus. In *The 27th International Symposium on Algorithms and Computation (ISAAC 2016)*. Schloss Dagstuhl, 2016.

[BB17] Marthe Bonamy and Nicolas Bousquet. Token sliding on chordal graphs. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 127–139. Springer, 2017.

[BBM23] Valentin Bartier, Nicolas Bousquet, and Amer E Mouawad. Galactic token sliding. *Journal of Computer and System Sciences*, 136:220–248, 2023.

[BFHM21] Marcin Briański, Stefan Felsner, Jędrzej Hodor, and Piotr Micek. Reconfiguring independent sets on interval graphs. In *46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021)*, pages 23–1. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2021.

[BHdW85a] Claude Benzaken, Peter L Hammer, and Dominique de Werra. Split graphs of dilworth number 2. *Discrete mathematics*, 55(2):123–127, 1985.

[BHdW85b] Claude Benzaken, Peter L Hammer, and Dominique de Werra. Threshold characterization of graphs with dilworth number two. *Journal of graph theory*, 9(2):245–267, 1985.

[BKL+21] Rémy Belmonte, Eun Jung Kim, Michael Lampis, Valia Mitsou, Yota Otachi, and Florian Sikora. Token sliding on split graphs. *Theory of Computing Systems*, 65(4):662–686, 2021.

[BKW14] Paul Bonsma, Marcin Kamiński, and Marcin Wrochna. Reconfiguring independent sets in claw-free graphs. In *Scandinavian Workshop on Algorithm Theory*, pages 86–97. Springer, 2014.

[Bon16] Paul Bonsma. Independent set reconfiguration in cographs and their generalizations. *Journal of Graph Theory*, 83(2):164–195, 2016.

[BVP26] Lucia Busolini and Mario Valencia-Pabon. Token sliding independent set reconfiguration on graphs with few P_4's. *arXiv preprint arXiv:2606.31815*, 2026.

[CP14] Tiziana Calamoneri and Rossella Petreschi. On pairwise compatibility graphs having dilworth number two. *Theoretical Computer Science*, 524:34–40, 2014.

[DDFE+14] Erik D Demaine, Martin L Demaine, Eli Fox-Epstein, Duc A Hoang, Takehiro Ito, Hirotaka Ono, Yota Otachi, Ryuhei Uehara, and Takeshi Yamada. Polynomial-time algorithm for sliding tokens on trees. In *International Symposium on Algorithms and Computation*, pages 389–400. Springer, 2014.

[FEHOU15] Eli Fox-Epstein, Duc A Hoang, Yota Otachi, and Ryuhei Uehara. Sliding token on bipartite permutation graphs. In *International Symposium on Algorithms and Computation*, pages 237–247. Springer, 2015.

[FH77] Stéphane Foldes and Peter L Hammer. Split graphs having dilworth number two. *Canadian Journal of Mathematics*, 29(3):666–672, 1977.

[FP25] Mathew C Francis and Veena Prabhakaran. Token sliding independent set reconfiguration on block graphs. In *45th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2025)*, pages 31–1. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2025.

[Ful56] Delbert R Fulkerson. Note on dilworth's decomposition theorem for partially ordered sets. In *Proc. Amer. Math. Soc*, volume 7, pages 701–702, 1956.

[HD05] Robert A Hearn and Erik D Demaine. Pspace-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. *Theoretical Computer Science*, 343(1-2):72–96, 2005.

[Hoa] Duc A Hoang. A note regarding "sliding tokens on a cactus". https://hoanganhduc.github.io/events/ISAAC2016/note.pdf. Accessed: 11/07/2026.

[IDH+11] Takehiro Ito, Erik D Demaine, Nicholas JA Harvey, Christos H Papadimitriou, Martha Sideri, Ryuhei Uehara, and Yushi Uno. On the complexity of reconfiguration problems. *Theoretical Computer Science*, 412(12-14):1054–1065, 2011.

[KMM12] Marcin Kamiński, Paul Medvedev, and Martin Milanič. Complexity of independent set reconfigurability problems. *Theoretical computer science*, 439:9–15, 2012.

[LM18] Daniel Lokshtanov and Amer E Mouawad. The complexity of independent set reconfiguration on bipartite graphs. *ACM Transactions on Algorithms (TALG)*, 15(1):1–19, 2018.

[Nis18] Naomi Nishimura. Introduction to reconfiguration. *Algorithms*, 11(4):52, 2018.

[Wro18] Marcin Wrochna. Reconfiguration in bounded bandwidth and tree-depth. *Journal of Computer and System Sciences*, 93:1–10, 2018.

[ZBA26] Shira Zucker and Yuval Ben-Abu. Efficient reconfiguration of multi-robot formations on ptolemaic topologies: Linear-time feasibility and exact optimization. In *2026 23rd International Conference on Ubiquitous Robots (UR)*, pages 128–133, 2026.