Minimum s-t Separator Reconfiguration on Chordal Graphs

Shira Zucker[1,2]

[1]Computer Science Department, Sapir Academic College, Israel
[2]Sapir Applied Science Institute, Faculty of Technology, Sapir Academic College, Israel

Abstract

We investigate the minimum s-t separator reconfiguration (MSR) problem on chordal graphs under the token jumping (TJ) and token sliding (TS) models. Our central structural result is that for any two non-adjacent vertices s, t, a chordal graph on n vertices has at most $n - 1$ minimum s-t separators, yielding a polynomial-size state space enumerable in $O(n + m)$ time. This enables an $O(n^2)$-time algorithm for shortest-path TJ-MSR, showing that the NP-hardness of the general case does not persist on chordal graphs. For TS-MSR, where polynomial-time solvability is already known on general graphs, our explicit framework provides an alternative $O(n^2k)$-time algorithm, where k is the (common) size of a minimum s-t separator. We also establish an $\Omega(n^2)$ lower bound on output size, showing that the TJ algorithm is output-optimal and the TS algorithm is output-sensitive up to a factor of $O(k)$.

1. Introduction

The field of combinatorial reconfiguration analyzes the solution space of a problem by turning it into a graph. For an instance of a problem I, a set of feasible solutions $\mathcal{V}$ forms the vertices of a *reconfiguration graph* $\mathcal{R}(I)$. An edge is placed between two solutions $S_A, S_B \in \mathcal{V}$ if S_B can be reached from S_A via a predefined reconfiguration step. This framework, systematically introduced by Ito *et al.* [12], allows us to ask questions about reachability (connectivity) and distance (shortest paths) within the solution space. Reconfiguration problems have been extensively studied for classical combinatorial problems such as independent set [13], graph coloring [5], and dominating set [10] (see [15] for a survey).

We investigate the reconfiguration of s-t separators. Given a graph $G = (V, E)$ with $n = |V|$ vertices and two non-adjacent vertices $s, t \in V$, a set $S \subseteq V \setminus \{s, t\}$ is an s-t separator if s and t are in different connected components of $G \setminus S$. A minimum s-t separator is an s-t separator with the smallest possible size; we denote this minimum size by k.

We assume throughout that s and t lie in the same connected component of G. Otherwise, the minimum s-t separator is the empty set ($k = 0$), and the reconfiguration problem becomes trivial.

In dynamic environments, an optimal solution is rarely static. System constraints often dictate that the selected set of critical nodes must be updated over time due to changing costs, node unavailability, or scheduled updates. What makes this process challenging is the strict requirement that the separation guarantee never be compromised during the transition, and the budget of blocked nodes never be exceeded. Reconfiguration formalizes this exact requirement. We must move from an initial solution to a target one through a sequence of intermediate states, each of which is itself a minimum s-t separator. The central questions are whether such a transformation exists at all, and how few moves it requires.

The *minimum s-t separator reconfiguration* (MSR) problem is defined on the state space $\mathcal{V}_k$, where each state is a minimum s-t separator of size k. We consider two standard, size-preserving adjacency relations.

- *Token jumping (TJ).* $S_A, S_B \in \mathcal{V}_k$ are adjacent if $S_B = (S_A \setminus \{u\}) \cup \{v\}$. This model, formalized by Kamiński *et al.* [13], corresponds to the standard adjacency relation of the Johnson graph applied to feasible solutions.

ICTCS 2026: Italian Conference on Theoretical Computer Science, September 7–9, 2026, Udine, Italy
shiraz@sapir.ac.il (S. Zucker)

- *Token sliding (TS).* $S_A, S_B \in \mathcal{V}_k$ are adjacent if they are TJ-adjacent, say $S_B = (S_A \setminus \{u\}) \cup \{v\}$, and in addition $(u, v) \in E(G)$. This model was introduced by Hearn and Demaine [11] and corresponds to sliding a token along an edge.

These two models capture distinct types of migration costs and transition constraints within a dynamic system. Token jumping is appropriate for scenarios where resources are globally reallocatable and can be freely reassigned across the entire network without structural limitations. In contrast, token sliding models situations with inherent locality constraints. It ensures that any reconfiguration strictly respects the underlying physical or logical topology of the network, limiting transitions to established structural connections.

Each reconfiguration step corresponds to a resource reallocation. Therefore, the length of the sequence serves as the natural cost measure. It represents the total number of system interventions, directly reflecting the downtime or operational effort required to migrate from the initial separator to the target.

We study the shortest path MSR problem. Given two minimum s-t separators $S_{\texttt{start}}, S_{\texttt{target}} \in \mathcal{V}_k$, the goal is to find the shortest sequence of reconfigurations between them under either the TJ or the TS model.

The MSR problem was recently studied in detail by Gomes *et al.* [9]. They established a fundamental complexity dichotomy on general graphs: by using a "forward-moving" lemma based on canonical s-t paths, they proved that the shortest path MSR problem for token sliding is polynomial-time solvable, whereas the corresponding problem for token jumping is NP-complete.

Gomes *et al.* [9] also provided fixed-parameter tractable algorithms for the TJ variant and highlighted the study of MSR on restricted graph classes as a key direction for further research.

The enumeration of minimal separators in chordal graphs has been studied by Kumar and Madhavan [14], who showed that all minimal vertex separators of a chordal graph can be listed in linear time using the clique tree structure. Their enumeration result is the starting point for our approach: on chordal graphs the entire state space of the reconfiguration problem, not merely a single solution, is small enough to be constructed explicitly.

The structural properties of separators in chordal graphs have been studied by Dirac [6] and Rose [16], who characterized chordal graphs via minimal vertex separators. Unlike general graphs, where the number of minimal separators can be exponential, chordal graphs admit a linear bound on their count, a property central to efficient enumeration algorithms [1].

In the context of combinatorial reconfiguration, chordal graphs have frequently allowed for tractable results where general cases are hard. For instance, while independent set reconfiguration is PSPACE-complete generally, it can be solved in linear time on chordal graphs [3]. Similarly, dominating set reconfiguration has been analyzed on subclasses of chordal graphs, such as split graphs [10].

In this paper, we address this direction by providing a complete polynomial-time solution for the class of chordal graphs.

Why reconfiguration is not implied by tractability. Computing a single minimum s-t separator is polynomial on every graph via max-flow algorithms. This efficiency implies little about the reconfiguration problem. The number of minimum s-t separators may be exponential even on graphs where each separator is trivially computable. For example, let G consist of k internally disjoint s-t paths, each with ℓ internal vertices. Then $n = k\ell + 2$, every minimum s-t separator has size k, and the minimum separators are exactly the sets picking one internal vertex from each path. Consequently, there are ℓ^k such separators. Accordingly, Gomes *et al.* [9] proved that the shortest-path MSR under token jumping is NP-complete on general graphs. The contribution of restricting to chordal graphs is therefore not that the base optimization problem becomes easy, as it is already solvable in polynomial time, but that the solution space itself collapses. It reduces from exponentially many states with no useful structure to at most $n - 1$ states that are linearly ordered along a path of the clique tree. Note that the example graph above is far from chordal, since any two of the paths induce a cycle of length $2\ell + 2$ with no chord.

Our main contributions are as follows. In Section 3, we show that the number of minimum s-t separators in a chordal graph is at most $n - 1$, enabling explicit construction of the reconfiguration

graph. Combined with an interval representation of separator membership along the clique path, this yields an $O(n^2)$-time algorithm for shortest-path TJ-MSR on chordal graphs (Section 4). For TS-MSR, where polynomial-time solvability is already known on general graphs [9], Section 5 provides an alternative $O(n^2k)$-time algorithm. Finally, in Section 6, we establish an $\Omega(n^2)$ lower bound on the worst-case output size, showing that the TJ algorithm is output-optimal and the TS algorithm is output-sensitive up to a factor of $O(k)$.

2. Preliminaries: Chordal Graphs and Clique Trees

A graph $G = (V, E)$ is *chordal* if every cycle of length at least four has a *chord*, meaning an edge between two non-consecutive vertices of the cycle. A clique tree of G is a tree $T = (\mathcal{C}, E_T)$ whose node set $\mathcal{C}$ is the set of maximal cliques of G, and whose edge set E_T is any set of $|\mathcal{C}| - 1$ pairs of maximal cliques forming a tree that satisfies the following running intersection property. For each vertex$v \in V$, the cliques containing v induce a connected subtree of T. Equivalently, for every $K, K' \in \mathcal{C}$, the set $K \cap K'$ is contained in every clique on the simple path between K and K' in T. Each edge $(K_i, K_j) \in E_T$ is labelled by the set $K_i \cap K_j$. Each chordal graph admits a clique tree [8].

An s-t separator S is minimal if no proper subset of S is an s-t separator, and minimum if it has the smallest cardinality among all s-t separators. Every minimum s-t separator is minimal, but not conversely. A set S is a minimal separator of G if it is a minimal a-b separator for some pair of vertices $a, b \in V$.

The following is a classical correspondence between minimal separators and clique trees.

Theorem 2.1 ([4, 8]). *Let G be a chordal graph and let $T = (\mathcal{C}, E_T)$ be any clique tree of G. A set $S \subseteq V$ is a minimal separator of G if and only if $S = K_i \cap K_j$ for some edge $(K_i, K_j) \in E_T$.*

A perfect elimination ordering (PEO) of G is an ordering $v_1, \ldots, v_n$ of V such that for every i, the neighbours of v_i among $v_{i+1}, \ldots, v_n$ form a clique. A graph is chordal if and only if it admits a PEO [7]. Maximum cardinality search (MCS) is a linear-time traversal that repeatedly visits a vertex with the largest number of already-visited neighbors. Its reverse visiting order is a PEO whenever G is chordal [17].

For $X \in \{\mathrm{TJ}, \mathrm{TS}\}$, we write $\mathcal{R}_k^X(G)$ for the graph with vertex set $\mathcal{V}_k$ in which two states are joined by an edge if they are X-adjacent. Since both adjacency relations are symmetric, $\mathcal{R}_k^X(G)$ is undirected, and $\mathcal{R}_k^{\mathrm{TS}}(G)$ is a spanning subgraph of $\mathcal{R}_k^{\mathrm{TJ}}(G)$. When the model is clear from context, we simply write $\mathcal{R}_k(G)$.

Throughout the paper, $G = (V, E)$ denotes a chordal graph on $n = |V(G)|$ vertices and $m = |E(G)|$ edges. Furthermore, $s, t \in V$ are fixed non-adjacent vertices that lie in the same connected component of G.

Why chordality is needed. A graph admits a clique tree, defined as a tree decomposition whose bags are exactly its maximal cliques, if and only if it is chordal [4, 8]. For a non-chordal graph, the construction fails already on the cycle C_4. Its minimal separators are the two pairs of non-adjacent vertices, and neither is the intersection of two maximal cliques since the maximal cliques are the four edges. One can certainly triangulate a graph and build a clique tree of the result, but the fill edges create separators that do not exist in G and destroy separators that do, meaning Lemma 3.3 no longer holds. This is also where the linear bound is lost, as general graphs may have exponentially many minimal separators [1].

3. The Polynomial State Space on Chordal Graphs

Our approach relies on a single fundamental property. On a chordal graph, all minimum s-t separators reside on a single path of the clique tree. This section proves this property and derives two essential

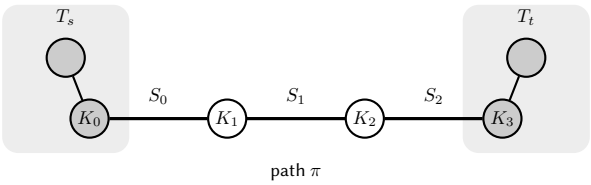

Figure 1: The clique tree T of a chordal graph G. Shaded regions are the subtrees T_s and T_t of cliques containing s and t; they are disjoint because s and t are non-adjacent. The bold edges form the minimum-length path $\pi = (K_0, \dots, K_r)$ between them, here with $r = 3$. Each edge of π is labelled by the separator $S_j = K_j \cap K_{j+1}$. By Lemma 3.3, every S_j is an s-t separator, and every minimal s-t separator of G arises this way; the minimum s-t separators are exactly the S_j of smallest size (Corollary 3.4).

consequences. The first is a linear bound on the number of states, and the second is an interval encoding of separator membership that allows us to test adjacency in $O(1)$ amortized time.

It is well known that chordal graphs have at most n maximal cliques and that their minimal separators correspond to edges of the clique tree [4, 8, 14]; the linear bound on $|\mathcal{V}_k|$ (Theorem 3.5) follows from these classical facts.

Theorem 3.1 ([2, 17]). *A clique tree of a chordal graph G can be constructed in time $O(n + m)$ using maximum cardinality search.*

The proof of this theorem is based on a maximum cardinality search (MCS), which produces a perfect elimination ordering (PEO) in $O(n + m)$ time [17]. From a PEO, the clique tree can be built by a single pass over vertices and edges in $O(n + m)$ time [2, 17].

Lemma 3.2. *Let $T = (\mathcal{C}, E_T)$ be a clique tree. Let $(K_i, K_j) \in E_T$ be an edge whose removal partitions T into two connected components, T_i (containing K_i) and T_j (containing K_j). Let $V_i = \bigcup_{C \in T_i} C$ and $V_j = \bigcup_{C \in T_j} C$. Then $V_i \cap V_j = K_i \cap K_j$.*

Proof: This follows directly from the running intersection property of clique trees; see, e.g., Blair and Peyton [2]. □

Lemma 3.3. *Let $T = (\mathcal{C}, E_T)$ be a clique tree of G, and let T_s and T_t be the subtrees induced by cliques containing s and t, respectively. Since T_s and T_t are disjoint connected subtrees of the tree T, there is a unique path of minimum length between them. Let $\pi = (K_0, K_1, \dots, K_r)$ denote this path, where $K_0 \in T_s$ and $K_r \in T_t$; see Figure 1. Then the following properties hold.*

(a) *For every edge (K_j, K_{j+1}) on π, the set $K_j \cap K_{j+1}$ is an s-t separator.*
(b) *Every minimal s-t separator equals $K_j \cap K_{j+1}$ for some edge (K_j, K_{j+1}) on π.*

Proof: Since s and t are non-adjacent, no maximal clique contains both, therefore T_s and T_t are disjoint; as they are connected subtrees of a tree, the path π is well defined, and by minimality no internal vertex of π lies in T_s or T_t.

Proof of (a). Let (K_j, K_{j+1}) be an edge on π, and let $S = K_j \cap K_{j+1}$. Removing (K_j, K_{j+1}) from T partitions the tree into two connected components, $\mathcal{T}_s$ (containing T_s) and $\mathcal{T}_t$ (containing T_t). Let $V_s = \bigcup_{C \in \mathcal{T}_s} C$ and $V_t = \bigcup_{C \in \mathcal{T}_t} C$. By Lemma 3.2, $V_s \cap V_t = K_j \cap K_{j+1} = S$.

We claim that no edge of G connects $V_s \setminus S$ to $V_t \setminus S$. Suppose in contradiction that $(u, v) \in E(G)$ with $u \in V_s \setminus S$ and $v \in V_t \setminus S$. Since G is chordal, (u, v) is contained in some maximal clique K. The clique K belongs to either $\mathcal{T}_s$ or $\mathcal{T}_t$. If $K \in \mathcal{T}_s$, then $K \subseteq V_s$, so $v \in K \subseteq V_s$, giving $v \in V_s \cap V_t = S$, which contradicts $v \in V_t \setminus S$. The case $K \in \mathcal{T}_t$ is symmetric and yields $u \in S$, a contradiction.

Since $s \in V_s \setminus S$ and $t \in V_t \setminus S$, and no edge of G connects $V_s \setminus S$ to $V_t \setminus S$, the set S separates s from t.

Proof of (b). Let S be a minimal s-t separator. We show that $S = K_j \cap K_{j+1}$ for some edge (K_j, K_{j+1}) on π.

Let R_s denote the connected component of s in $G \setminus S$. Note that $R_s \subseteq V \setminus S$ and $t \notin R_s$. Define

$$j^* = \max\{\, j \mid K_j \cap R_s \neq \emptyset \,\}$$

Claim. j^* is well-defined and $0 \leq j^* < r$. Since $s \in K_0$ and $s \in R_s$, we have $K_0 \cap R_s \neq \emptyset$, and therefore $j^* \geq 0$. For the upper bound, suppose in contradiction that $K_r \cap R_s \neq \emptyset$, and let $v \in K_r \cap R_s$. Since v and t both belong to the clique K_r, they are adjacent in G. Moreover, $v \notin S$ (as $R_s \cap S = \emptyset$) and $t \notin S$, so the edge (v, t) exists in $G \setminus S$. Since $v \in R_s$, vertex t is reachable from s in $G \setminus S$, contradicting the assumption that S separates s from t. Thus, $K_r \cap R_s = \emptyset$ and $j^* < r$.

Claim. $K_{j^*} \cap K_{j^*+1} \subseteq S$. By the definition of j^*, there exists a vertex $u \in K_{j^*} \cap R_s$, and $K_{j^*+1} \cap R_s = \emptyset$. Let $v \in K_{j^*} \cap K_{j^*+1}$ and suppose $v \notin S$. Since u and v belong to the clique K_{j^*}, they are adjacent in G. Since $u \in R_s$ and $v \notin S$, the edge (u, v) exists in $G \setminus S$, and therefore $v \in R_s$. But then $v \in K_{j^*+1} \cap R_s$, contradicting the maximality of j^*. Hence $K_{j^*} \cap K_{j^*+1} \subseteq S$.

Conclusion. By part (a), $K_{j^*} \cap K_{j^*+1}$ is an s-t separator. Since $K_{j^*} \cap K_{j^*+1} \subseteq S$ and S is a minimal s-t separator, no proper subset of S is an s-t separator, forcing $S = K_{j^*} \cap K_{j^*+1}$. □

We write $E(\pi) = \{(K_j, K_{j+1}) \mid 0 \leq j < r\}$ for the edge set of π, and $S_j = K_j \cap K_{j+1}$ for the separator labeling the j-th edge.

Corollary 3.4. *Let $k = \min\{\, |S_j| \mid 0 \leq j < r \,\}$. Then k is the size of a minimum s-t separator of G, and*

$$\mathcal{V}_k \;=\; \{\, S_j \mid 0 \leq j < r \text{ and } |S_j| = k \,\}.$$

Proof: By Lemma 3.3(a) each S_j is an s-t separator, therefore the minimum size is at most k; conversely, every minimum s-t separator is minimal and hence equals some S_j by Lemma 3.3(b), therefore its size is at least k. Thus, the minimum size is exactly k, and the minimum s-t separators are precisely the s-t separators of size k, i.e., S_j with $|S_j| = k$. □

Lemma 3.3 directly yields a bound on the number of minimal, and hence minimum, s-t separators.

Theorem 3.5. *Let $G = (V, E)$ be a chordal graph, and let $s, t \in V$ be non-adjacent vertices. Then the number of minimum s-t separators is at most $n - 1$.*

Proof: By Lemma 3.3(b), every minimal s-t separator belongs to $\mathcal{S}_{s,t} = \{K_i \cap K_j \mid (K_i, K_j) \in E(\pi)\}$. Since T is a tree on $|\mathcal{C}|$ nodes, it has $|\mathcal{C}| - 1$ edges, therefore $|E(\pi)| \leq |E_T| = |\mathcal{C}| - 1$. Every maximal clique of a chordal graph is uniquely identified by the vertex with the smallest index in any perfect elimination ordering [7, 2], which gives an injection from maximal cliques to vertices and hence $|\mathcal{C}| \leq n$.

Let $\mathcal{V}_k$ be the set of all minimum s-t separators of size k. Since $\mathcal{V}_k \subseteq \mathcal{S}_{s,t}$, we obtain $|\mathcal{V}_k| \leq |\mathcal{S}_{s,t}| \leq n - 1$, as claimed. □

Observation 3.6. *Let S_i and S_j be separators along π with $j > i$ and $|S_i| = |S_j|$. If $S_i = S_j$, then every intermediate separator S_ℓ ($i < \ell < j$) with $|S_\ell| = |S_i|$ also satisfies $S_\ell = S_i$. Consequently, among the separators of any fixed size k, a separator is a duplicate if and only if it equals the most recently encountered separator of size k.*

Proof: Let $v \in S_i = K_i \cap K_{i+1}$. Since $S_i = S_j$, we also have $v \in S_j = K_j \cap K_{j+1}$, therefore, $v \in K_{i+1}$ and $v \in K_j$. By the running intersection property of the clique tree, the cliques containing v form a connected subtree. Since this subtree contains both K_{i+1} and K_j, it contains every clique K_ℓ with $i + 1 \leq \ell \leq j$. In particular, $v \in K_\ell \cap K_{\ell+1} = S_\ell$ for all $i \leq \ell < j$, and therefore $S_i \subseteq S_\ell$.

Now suppose $|S_\ell| = |S_i| = k$. Since $S_i \subseteq S_\ell$ and both sets have cardinality k, we conclude $S_\ell = S_i$. □

3.1. Interval Representation Along the Clique Path

The structure of the clique path yields a compact representation of the separator membership that is exploited in Algorithm 2.

Definition 3.7 (Vertex intervals). *Let $\pi = (K_0, K_1, \ldots, K_r)$ be the clique path from T_s to T_t. For each vertex v appearing in some clique on π, define*

$$\text{first}(v) = \min\{i \mid v \in K_i\}, \qquad \text{last}(v) = \max\{i \mid v \in K_i\}.$$

The cliques containing v form a subtree of T, and the intersection of a subtree with the path π is a (possibly empty) subpath. Thus, $v \in K_i$ for all $\text{first}(v) \le i \le \text{last}(v)$.

Lemma 3.8. *$v \in S_i = K_i \cap K_{i+1}$ if and only if $\text{first}(v) \le i$ and $\text{last}(v) \ge i+1$.*

Proof: $v \in S_i \Leftrightarrow v \in K_i$ and $v \in K_{i+1} \Leftrightarrow \text{first}(v) \le i$ and $\text{last}(v) \ge i+1$, where the second equivalence uses the fact that the intersection of the subtree of cliques containing v with the path π is a contiguous subpath. □

Proposition 3.9. *For separators S_i and S_j along π with $i < j$, the following relations hold.*

$$S_i \cap S_j = \{\, v \mid \text{first}(v) \le i \text{ and } \text{last}(v) \ge j+1 \,\}, \tag{1}$$

$$S_i \setminus S_j = \{\, v \mid \text{first}(v) \le i \text{ and } i+1 \le \text{last}(v) \le j \,\}, \tag{2}$$

$$S_j \setminus S_i = \{\, v \mid i+1 \le \text{first}(v) \le j \text{ and } \text{last}(v) \ge j+1 \,\}. \tag{3}$$

Proof: By Lemma 3.8, $v \in S_i$ if and only if $\text{first}(v) \le i$ and $\text{last}(v) \ge i+1$. Since $i < j$, the conjunction of conditions for $v \in S_i$ and $v \in S_j$ simplifies directly to the binding constraints $\text{first}(v) \le i$ and $\text{last}(v) \ge j+1$, proving (1). For (2), the condition $v \in S_i \setminus S_j$ means $\text{first}(v) \le i < j$ is already satisfied, so $v \notin S_j$ strictly requires the negation $\text{last}(v) \le j$. Symmetrically for (3), $v \in S_j \setminus S_i$ guarantees $\text{last}(v) \ge j+1 > i$, leaving $\text{first}(v) \ge i+1$ as the sole requirement for $v \notin S_i$. □

Corollary 3.10. *For any fixed separator index i, the intersection size $|S_i \cap S_j|$ is non-increasing as $j > i$ ranges over the separator indices along π.*

Proof: Increasing j tightens the constraint $\text{last}(v) \ge j+1$ in (1), therefore, the set $\{\, v \mid \text{first}(v) \le i \text{ and } \text{last}(v) \ge j+1 \,\}$ can only shrink. □

Corollary 3.11. *Two minimum s-t separators S_i and S_j (with $i < j$, $|S_i| = |S_j| = k$) are TJ-adjacent if and only if $|S_i \cap S_j| = k-1$. Equivalently, by (2) and (3), exactly one vertex u satisfies $\text{first}(u) \le i$ and $i+1 \le \text{last}(u) \le j$ (the vertex that "exits"), and exactly one vertex w satisfies $i+1 \le \text{first}(w) \le j$ and $\text{last}(w) \ge j+1$ (the vertex that "enters").*

Proof: The first equivalence holds because $|S_i| = |S_j| = k$ implies $|S_i \cap S_j| = k-1$ if and only if $|S_i \triangle S_j| = 2$, which is the definition of TJ-adjacency. The interval characterization then follows directly from (2) and (3). □

3.2. Enumeration of Minimum s-t Separators

Algorithm 1 enumerates all minimal s-t separators corresponding to edges in the unique path of the clique tree T connecting the subtree of cliques containing s to the subtree of cliques containing t, and then extracts those of minimum size. For each edge (K_i, K_{i+1}) on this path, we compute the separator $S_i = K_i \cap K_{i+1}$ by scanning the vertices of K_i and retaining only those that also appear in K_{i+1}. To handle possible repetitions among separators of the same size, we exploit Observation 3.6. Among separators of any fixed size, duplicates can only be consecutive in path order. The algorithm therefore tracks the separator most recently encountered of the current minimum size k and skips duplicates by a single comparison.

Algorithm 1: BuildStateSpace(G, s, t)

Input : A chordal graph $G = (V, E)$ and non-adjacent $s, t \in V$

Output: The state space $\mathcal{V}_k$ (minimum s-t separators, each stored with its edge index along π), size k, and the path π

```
Compute clique tree T = (C, E_T);
T_s ← {C ∈ C | s ∈ C}; T_t ← {C ∈ C | t ∈ C};
π ← the simple path from T_s to T_t via BFS on T;
Let (K_0, K_1, ..., K_r) be the cliques along π;
k ← ∞; V_k ← ∅; S_prev ← nil ;                    // Last seen separator of size k
for i = 0 to r − 1 do
    S_i ← K_i ∩ K_{i+1};
    if |S_i| < k then
        k ← |S_i|; V_k ← {S_i}; S_prev ← S_i;
    else if |S_i| = k then
        if S_i ≠ S_prev then
            add S_i to V_k;
            S_prev ← S_i ;                                   // Obs. 3.6
return (V_k, k, π);
```

Theorem 3.12. *Algorithm 1 correctly computes $\mathcal{V}_k$, k, and π in time $O(n + m)$.*

Proof: **Correctness.** By Corollary 3.4, the minimum s-t separators are exactly the size-k intersections $K_i \cap K_{i+1}$ along π. The algorithm enumerates these in path order and maintains $\mathcal{V}_k$ as the set of separators achieving the current minimum size k. When a strictly smaller separator is found, k and $\mathcal{V}_k$ are reset. Duplicates between the size-k separators are detected by comparing with the most recently added separator of size k. By Observation 3.6, this comparison suffices to catch all duplicates. At termination, $\mathcal{V}_k$ contains exactly the minimum-size separators.

Runtime analysis. The clique tree is constructed in $O(n + m)$ time (Theorem 3.1). Identifying the path π through BFS takes $O(n)$ time. Computing all intersections $S_i = K_i \cap K_{i+1}$ costs $\sum_i (|K_i| + |K_{i+1}|) = O(\sum |K_i|) = O(n + m)$, since the sum of all maximal clique sizes in a chordal graph is $O(n + m)$ [2]. Deduplication adds $O(n + m)$ time by Observation 3.6. The total running time is $O(n + m)$. □

Remark 3.13. *Algorithm 1 uses $O(n+m)$ space. The clique tree and the cliques along π occupy $O(n+m)$ space. For the output, each separator $S_i = K_i \cap K_{i+1}$ satisfies $S_i \subseteq K_i$, and since the cliques K_i are distinct maximal cliques, we have $\sum_{S \in \mathcal{V}_k} |S| \leq \sum_i |K_i| = O(n + m)$ [2]. The state space therefore never exceeds the size of the input.*

4. Shortest Path for Token Jumping (TJ)

We now address the shortest path MSR problem for token jumping on chordal graphs. While this problem is NP-complete on general graphs [9], the linear bound on $|\mathcal{V}_k|$ established in Theorem 3.5 allows for an explicit construction of the TJ reconfiguration graph and a polynomial-time shortest-path computation. With the state space explicitly available, the problem reduces to a standard breadth-first search. The only remaining obstacle is building the reconfiguration graph efficiently without paying $O(k)$ time per pair. The interval encoding developed in the previous section provides exactly the mechanism needed to overcome this bottleneck.

A direct approach would compare all $\binom{|\mathcal{V}_k|}{2} = O(n^2)$ pairs of separators, computing each intersection in $O(k)$ time, for a total of $O(n^2 k)$. Using the interval representation of Section 3.1, we avoid the

per-pair $O(k)$ cost by sweeping along the clique path and maintaining intersection sizes incrementally. By Corollary 3.10, for a fixed separator S_{i_a}, the intersection size $|S_{i_a} \cap S_{i_b}|$ is non-increasing in b, which allows a single pointer to track the decreasing count as b advances. Algorithm 2 details this approach.

Algorithm 2: Solve_MSR_TJ($G, s, t, S_{\texttt{start}}, S_{\texttt{target}}$)

Input : Chordal graph $G = (V, E)$, non-adjacent $s, t \in V$, and minimum s-t separators $S_{\texttt{start}}, S_{\texttt{target}}$

Output: The length of a shortest TJ-reconfiguration sequence and, if requested, the sequence itself. Returns ∞ if no sequence exists.

```
// Phase 1: State space and intervals
```

$(\mathcal{V}_k, k, \pi) \leftarrow$ BuildStateSpace(G, s, t);

if $S_{\texttt{start}} \notin \mathcal{V}_k$ ***or*** $S_{\texttt{target}} \notin \mathcal{V}_k$ **then**

  return ∞;

Compute $\mathrm{first}(v)$ and $\mathrm{last}(v)$ for every vertex v on π;

```
// Phase 2: Build R_k(G) via sweep
```

Let $S_{i_1}, S_{i_2}, \ldots, S_{i_q}$ be the elements of $\mathcal{V}_k$ in path order;

$E_J \leftarrow \emptyset$;

for $a = 1$ **to** q **do**

  Sort the k vertices of S_{i_a} by $\mathrm{last}(v)$ in non-decreasing order; call them $v_1, v_2, \ldots, v_k$;

  $count \leftarrow k$;

  $ptr \leftarrow 1$;

  for $b = a + 1$ **to** q **do**

    // Remove vertices no longer in S_{i_b}

    while $ptr \leq k$ ***and*** $\mathrm{last}(v_{ptr}) \leq i_b$ **do**

      $count \leftarrow count - 1$;

      $ptr \leftarrow ptr + 1$;

    if $count = k - 1$ **then**

      Add (S_{i_a}, S_{i_b}) to E_J;

```
// Phase 3: BFS
```

Run BFS on $\mathcal{R}_k(G) = (\mathcal{V}_k, E_J)$ from $S_{\texttt{start}}$;

Extract the reconfiguration sequence by following BFS parent pointers;

return dist$(S_{\texttt{start}}, S_{\texttt{target}})$ and the sequence;

Theorem 4.1. *Algorithm 2 solves the shortest path MSR problem under token jumping on chordal graphs in time $O(n^2)$.*

Proof: **Correctness.** We show that the sweep correctly determines, for every pair (S_{i_a}, S_{i_b}) with $a < b$, whether $|S_{i_a} \cap S_{i_b}| = k - 1$.

Consider a fixed index a. At initialization, $count = |S_{i_a}| = k$ and $ptr = 1$, with the vertices of S_{i_a} sorted as $v_1, v_2, \ldots, v_k$ by non-decreasing $\mathrm{last}(v)$.

By Proposition 3.9 (1), a vertex $v \in S_{i_a}$ belongs to $S_{i_a} \cap S_{i_b}$ if and only if $\mathrm{last}(v) \geq i_b + 1$. Since $v \in S_{i_a}$ already implies $\mathrm{first}(v) \leq i_a < i_b$, the only binding condition is $\mathrm{last}(v) \geq i_b + 1$, which is equivalent to $\neg(\mathrm{last}(v) \leq i_b)$.

The while loop removes from the count every vertex with $\mathrm{last}(v) \leq i_b$. Since the vertices are sorted by $\mathrm{last}(v)$ and the sequence $i_{a+1} < i_{a+2} < \cdots < i_q$ is strictly increasing, the pointer ptr advances monotonically and each vertex is removed at most once. After the while loop, $count$ equals the number of vertices in S_{i_a} with $\mathrm{last}(v) > i_b$, which is $|S_{i_a} \cap S_{i_b}|$.

An edge (S_{i_a}, S_{i_b}) is added to E_J precisely when $count = k - 1$, which is the TJ-adjacency condition (since $|S_{i_a}| = |S_{i_b}| = k$ and $|S_{i_a} \triangle S_{i_b}| = 2k - 2|S_{i_a} \cap S_{i_b}| = 2$).

The algorithm iterates over all ordered pairs (a, b) with $a < b$, and by the symmetry of TJ-adjacency, the graph E_J is constructed correctly. Phase 3 is a standard BFS.

Runtime analysis.

Phase 1. By Theorem 3.12, computing $\mathcal{V}_k$, k and π takes $O(n+m)$ time. Testing whether $S_{\texttt{start}}$ and $S_{\texttt{target}}$ belong to $\mathcal{V}_k$ takes $O(k)$ time per candidate, using a hash table of the sorted vertex lists of the states. Building this table costs $O\Big(\sum_{S\in\mathcal{V}_k}|S|\Big) = O(n+m)$, since each state is contained in a distinct maximal clique. Computing $\mathrm{first}(v)$ and $\mathrm{last}(v)$ requires a single scan of the cliques on π. For each clique K_i, we update the interval of every $v \in K_i$. The total work is $\sum_i |K_i| = O(n+m)$. Phase 1 thus costs $O(n+m)$.

Phase 2. Let $q = |\mathcal{V}_k| \le n-1$. For each fixed index a, sorting the k vertices of S_{i_a} by $\mathrm{last}(v)$ via counting sort with range $\{0,\dots,r\} \subseteq \{0,\dots,n\}$ takes $O(k+n)$ time. The inner loop over $b = a+1,\dots,q$ executes at most $q-a$ iterations, during which the pointer ptr advances at most k times overall. The total work for a fixed a is therefore $O(k+n+q-a)$. Summing this over all a yields

$$\sum_{a=1}^{q} O(k+n+q) \;=\; O(q(k+n+q)) \;=\; O(n^2),$$

where the last equality uses $k \le n$ and $q \le n$.

Phase 3. The BFS on $\mathcal{R}_k(G)$ takes $O(|\mathcal{V}_k| + |E_J|) = O(n+n^2) = O(n^2)$ time.

Total. The overall complexity is $O(n+m) + O(n^2) + O(n^2) = O(n^2)$. □

Remark 4.2. *Algorithm 2 requires $O(n^2)$ space, a bound dominated by the explicit storage of the edge set E_J. However, the breadth-first search can alternatively generate neighbors on the fly directly from the interval representation. This modification avoids the explicit construction of E_J and reduces the space complexity to $O(n+m)$, at a minor operational cost of recomputing adjacency tests for each layer of the BFS traversal.*

5. Shortest Path for Token Sliding (TS)

On general graphs, Gomes *et al.* [9] showed that the shortest path MSR problem under token sliding is solvable in polynomial time using a forward-moving lemma on canonical s-t paths. Their algorithm works without enumerating all minimum separators explicitly; instead, each TS step is validated by checking that the new set remains a minimum separator, which requires graph searches of cost $O(n+m)$ per move. In contrast, our approach on chordal graphs exploits the explicit polynomial bound on the state space (Theorem 3.5) to construct the entire TS reconfiguration graph on $\mathcal{V}_k$ and to perform shortest-path search directly on this explicit graph. This yields a unified explicit framework for both TS and TJ on chordal graphs. An implicit TS algorithm that is faster on sparse, bounded-degree graphs is presented in the full version.

An Explicit $O(n^2k)$ TS Algorithm We first present an explicit algorithm that reuses the state space $\mathcal{V}_k$, computed in Section 3, and mirrors the structure of the TJ algorithm.

Theorem 5.1. *Algorithm 3 solves the shortest path MSR problem under token sliding on chordal graphs in time $O(n^2k)$.*

Proof: **Correctness.** By Theorem 3.12, Phase 1 returns exactly $\mathcal{V}_k$. Phase 2 adds the edge $\{S_A, S_B\}$ precisely when $|S_A \cap S_B| = k-1$ (equivalently $|S_A \triangle S_B| = 2$, as both states have size k) and the two differing vertices are adjacent in G, which is the definition of TS-adjacency; hence $\mathcal{R}_k(G)$ contains all valid transitions and no others. Phase 3 is a standard BFS, which returns a shortest path from $S_{\texttt{start}}$ to $S_{\texttt{target}}$, or ∞ if none exists.

Runtime analysis. By Theorem 3.12, Phase 1 takes $O(n+m)$ time. In chordal graphs, the sum of all maximal clique sizes is $O(n+m)$, which bounds the total work for constructing $\mathcal{V}_k$ and testing

Algorithm 3: Solve_MSR_TS($G, s, t, S_{\texttt{start}}, S_{\texttt{target}}$)

Input : Chordal graph $G = (V, E)$ on n vertices, non-adjacent $s, t \in V$, and sets $S_{\texttt{start}}, S_{\texttt{target}} \subseteq V$

Output: The length of a shortest TS-reconfiguration sequence and, if requested, the sequence itself. Returns ∞ if no sequence exists.

```
// Phase 1: Find state space
```
$(\mathcal{V}_k, k, \pi) \leftarrow$ BuildStateSpace(G, s, t) ; // π not used here

if $S_{\texttt{start}} \notin \mathcal{V}_k$ ***or*** $S_{\texttt{target}} \notin \mathcal{V}_k$ **then**

  return ∞;

```
// Phase 2: Build TS reconfiguration graph R_k(G)
```
Let $\mathcal{R}_k(G) = (\mathcal{V}_k, E_S)$ be an empty undirected graph;

for *each unordered pair* $\{S_A, S_B\} \subseteq \mathcal{V}_k$ *with* $S_A \neq S_B$ **do**

  if $|S_A \cap S_B| = k - 1$ **then**

    $\{u\} \leftarrow S_A \setminus S_B$;

    $\{v\} \leftarrow S_B \setminus S_A$;

    if $(u, v) \in E(G)$ **then**

      Add edge (S_A, S_B) to E_S;

```
// Phase 3: Shortest path in R_k(G)
```
Run BFS on $\mathcal{R}_k(G)$ starting from $S_{\texttt{start}}$;

Extract the reconfiguration sequence by following BFS parent pointers;

return dist$(S_{\texttt{start}}, S_{\texttt{target}})$ and the sequence;

membership for $S_{\texttt{start}}$ and $S_{\texttt{target}}$. Theorem 3.5 bounds $|\mathcal{V}_k| \leq n - 1$. Phase 2 iterates over all $O(|\mathcal{V}_k|^2) = O(n^2)$ unordered pairs (S_A, S_B). Inside the loop, computing $|S_A \cap S_B|$ takes $O(k)$. Since the intersection size is $k - 1$, finding u, v takes $O(k)$. Checking if $(u, v) \in E(G)$ takes $O(1)$ (assuming adjacency matrix or hash set for edges). Together, we get that Phase 2 runs in $O(n^2 k)$. Phase 3 (BFS) takes $O(|\mathcal{V}_k| + |E_S|) = O(n^2)$ time, and the overall complexity is $O(n^2 k)$. □

Comparison with Gomes et al. [9]. Let L denote the length of the shortest reconfiguration sequence. The algorithm of Gomes *et al.* achieves a runtime of $O(L(n+m))$. Since $L \leq |\mathcal{V}_k| \leq n-1$, our $O(n^2 k)$ algorithm outperforms their approach on dense instances ($m = \Theta(n^2)$) in the regime where $k = o(L)$. For example, when $k = O(1)$ and $L = \Theta(n)$, our algorithm runs in $O(n^2)$ time, whereas their algorithm takes $O(n^3)$ time. Furthermore, because our explicit construction builds the entire reconfiguration graph $\mathcal{R}_k(G)$, it naturally supports multiple queries on the same graph without necessitating any recomputation.

6. Lower Bound on Output Size

In this section, we demonstrate that the running times of our algorithms are justified by the worst-case size of the output. For TJ, Algorithm 2 runs in $O(n^2)$ time, matching the $\Omega(n^2)$ lower bound and making it output-optimal under explicit representation. For TS, Algorithm 3 has a factor $O(k)$ gap and is therefore output-sensitive up to $O(k)$.

Proposition 6.1. *There exists a family of chordal graphs on n vertices such that the shortest reconfiguration path between two minimum s-t separators has length $\Omega(n)$, and each separator in the sequence has size $\Omega(n)$, resulting in a total explicit output size of $\Omega(n^2)$.*

Proof: We construct such a graph using an interval graph model, which forms a subclass of chordal graphs. Let $G = (V, E)$ be a graph on n vertices, where $V = \{1, 2, \ldots, n\}$. Assume $n \geq 6$. We set a parameter $q = \lfloor n/3 \rfloor$ and define $p = n - 2q$.

Graph construction and maximal cliques. We define a sequence of overlapping intervals. For $i = 0, \ldots, p-1$, let

$$K_i = \{i+1, i+2, \ldots, i+2q+1\}.$$

Each set K_i is an interval of $2q+1$ consecutive integers. The graph G is the intersection graph of these intervals, meaning two vertices are adjacent if and only if they appear together in some K_i. This makes G an interval graph and, consequently, chordal.

By the construction of the graph, two vertices u and v are adjacent if and only if $|u-v| \leq 2q$. Since the elements of K_i are consecutive integers spanning a range of exactly $2q$, every pair of vertices within K_i satisfies this adjacency condition. Therefore, each K_i forms a clique. It is also maximal, because any $j \leq i$ satisfies $(i+2q+1)-j \geq 2q+1 > 2q$, and any $j > i+2q+1$ satisfies $j-(i+1) > 2q$, meaning no outside vertex can be adjacent to all elements of K_i. Furthermore, since any arbitrary clique C in the graph must also satisfy $\max C - \min C \leq 2q$, it is guaranteed to be contained within $K_{\min\{\min C-1,\, p-1\}}$. This proves that $\mathcal{C} = \{K_0, \ldots, K_{p-1}\}$ is exactly the set of maximal cliques of G. Finally, since the intersections $K_i \cap K_{i+1}$ naturally satisfy the running intersection property along their index order, the simple path $K_0 - K_1 - \cdots - K_{p-1}$ constitutes a valid clique tree T of G.

Choice of s and t. Let $s = 1$ and $t = n$. We first note that s and t are non-adjacent, since $n-1 > 2q$ holds for all $n \geq 6$. We now verify that s and t each appear in exactly one maximal clique.

Vertex $s = 1$ appears in K_i if and only if $i+1 \leq 1 \leq i+2q+1$, which holds if and only if $i = 0$. Thus, s appears only in K_0.

Vertex $t = n$ appears in K_i if and only if $i+1 \leq n \leq i+2q+1$, which simplifies to $n-2q-1 \leq i \leq n-1$. Since $i \leq p-1 = n-2q-1$ by definition, we see that t appears only in K_{p-1}, where $i = p-1 = n-2q-1$.

Therefore, $T_s = \{K_0\}$ and $T_t = \{K_{p-1}\}$ are singleton subtrees located at opposite ends of the path T.

Minimum separators. By Lemma 3.3, the minimum s-t separators correspond to intersections of adjacent cliques along the unique path from K_0 to K_{p-1}.

For $i = 0, \ldots, p-2$, define

$$S_i = K_i \cap K_{i+1} = \{i+2, i+3, \ldots, i+2q+1\}.$$

All edges of the clique path yield separators of the same size $2q$, therefore by Corollary 3.4 we have $k = 2q$ and $\mathcal{V}_k = \{S_0, \ldots, S_{p-2}\}$; these sets are pairwise distinct, since $S_i \neq S_{i+1}$ for every i.

TJ-adjacency of consecutive separators. Observe that $S_i = \{i+2, \ldots, i+2q+1\}$ and $S_{i+1} = \{i+3, \ldots, i+2q+2\}$, therefore $S_i \setminus S_{i+1} = \{i+2\}$ and $S_{i+1} \setminus S_i = \{i+2q+2\}$. Hence $|S_i \triangle S_{i+1}| = 2$, confirming that consecutive separators are TJ-adjacent. They are also TS-adjacent, since vertices $i+2$ and $i+2q+2$ are both contained in K_{i+1} and are therefore adjacent in G.

No shortcuts exist. Note that $j - i \leq p-2 = n-2q-2 < 2q$, since $q = \lfloor n/3 \rfloor \geq (n-2)/3$ implies $4q \geq 4(n-2)/3 > n-2$. This means $S_i \cap S_j \neq \emptyset$ for all valid $i < j$. For $j \geq i+2$, we have $S_i \setminus S_j = \{i+2, \ldots, j+1\}$ and $S_j \setminus S_i = \{i+2q+2, \ldots, j+2q+1\}$, giving $|S_i \triangle S_j| = 2(j-i) \geq 4$. Therefore, S_i and S_j are not TJ-adjacent. The TJ reconfiguration graph on $\{S_0, \ldots, S_{p-2}\}$ is a simple path, and the shortest reconfiguration sequence from S_0 to S_{p-2} has length exactly $p-2$.

Reconfiguration sequence length and output size. The length of the shortest reconfiguration path is $L = p-2 = n-2q-2 = \Theta(n)$. The sequence visits $L+1 = p-1 = \Theta(n)$ separators in total, each having size $k = 2q = \Theta(n)$. Therefore, explicitly outputting every intermediate separator yields a total output size of $(p-1) \cdot 2q = \Theta(n) \cdot \Theta(n) = \Omega(n^2)$. □

Remark 6.2. *The bound of Proposition 6.1 refers to the explicit representation of the reconfiguration sequence, in which every intermediate separator is listed in full. Under a differential encoding, where a sequence is instead given by its L moves (u_i, v_i), the output has size $\Theta(L) = O(n)$ and the lower bound does not apply. Whether $O(n^2)$ time is strictly necessary for producing the differential output remains an open problem.*

7. Discussion and Open Problems

Connectivity of $\mathcal{R}_k(G)$. Unlike the state space itself, the reconfiguration graph need not be connected. Consider a graph G with maximal cliques $K_0 = \{s, a, b\}$, $K_1 = \{a, b, c, d\}$, and $K_2 = \{c, d, t\}$ forming the clique path $K_0 - K_1 - K_2$. The graph G is chordal, s and t are non-adjacent, and the two s-t paths s, a, c, t and s, b, d, t are internally disjoint. Consequently, the minimum separator size is $k = 2$, and the state space is $\mathcal{V}_k = \{\{a, b\}, \{c, d\}\}$. Since these two states are disjoint, they are not TJ-adjacent, leaving $\mathcal{R}_k(G)$ as a graph with two isolated vertices. Characterizing the exact class of chordal graphs for which $\mathcal{R}_k(G)$ is connected remains an open problem. However, a sufficient condition is that every pair of consecutive states along π satisfies $|S_{i_a} \cap S_{i_{a+1}}| = k - 1$, as this guarantees the clique path itself induces a connected path in $\mathcal{R}_k(G)$.

Greedy strategies along the clique path. When $\mathcal{R}_k(G)$ is connected, a natural heuristic is to greedily select, at each step, the farthest minimum separator along π reachable in a single TJ move. If this approach always produces a shortest reconfiguration sequence, it would yield a faster $O(nk)$-time algorithm. We formalize this as follows.

Conjecture 7.1. *If the TJ reconfiguration graph $\mathcal{R}_k(G)$ is connected, the greedy strategy of transitioning to the furthest TJ-adjacent minimum separator along the clique path computes a shortest reconfiguration sequence.*

Differential output encoding. As discussed in Section 6, our $\Omega(n^2)$ lower bound applies only when the output sequence is represented explicitly. If the sequence is instead given by a differential encoding of the moved vertices, the output size drops to $O(n)$. It remains open whether an algorithm can compute and output this differential sequence in strictly less than $O(n^2)$ time.

Sparse TJ algorithms. Developing a faster neighbor generation for TJ on sparse graphs remains open. This would be analogous to the $O(nk^2\Delta)$ TS improvement presented in the full version of this work.

Extensions. We suggest investigating extensions to weakly chordal graphs, graphs of bounded treewidth, and the token addition and removal (TAR) model [12].

Declaration of generative AI and AI-assisted technologies in the manuscript preparation process

During the preparation of this work the author(s) used Claude and Overleaf for grammar and spelling checking, paraphrasing, and rewording. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the content of the published article.

References

[1] A. Berry, J. P. Bordat, and O. Cogis. Generating all the minimal separators of a graph. *Graph-Theoretic Concepts in Computer Science (WG 1999)*, volume 1665 of Lecture Notes in Computer Science. Springer, 1999.

[2] J. R. S. Blair and B. Peyton. An introduction to chordal graphs and clique trees. In *Graph Theory and Sparse Matrix Computation*, pages 1–29. Springer, 1993.
[3] M. Bonamy and N. Bousquet. Reconfiguring independent sets in chordal graphs. *arXiv preprint arXiv:1406.1433*, 2014.
[4] P. Buneman. A characterization of rigid circuit graphs. *Discrete Mathematics*, 9(3):205–212, 1974.
[5] L. Cereceda, J. van den Heuvel, and M. Johnson. Finding paths between 3-colorings. *Journal of Graph Theory*, 67(1):69–82, 2011.
[6] G. A. Dirac. On rigid circuit graphs. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 25:71–76, 1961.
[7] D. R. Fulkerson and O. A. Gross. Incidence matrices and interval graphs. *Pacific Journal of Mathematics*, 15(3):835–855, 1965.
[8] F. Gavril. The intersection graphs of subtrees in trees are exactly the chordal graphs. *Journal of Combinatorial Theory, Series B*, 16(1):47–56, 1974.
[9] G. C. M. Gomes, C. Legrand-Duchesne, R. Mahmoud, A. E. Mouawad, Y. Okamoto, V. F. dos Santos, and T. C. van der Zanden. Minimum Separator Reconfiguration. In *18th International Symposium on Parameterized and Exact Computation (IPEC 2023)*, volume 285 of *LIPIcs*, pages 9:1–9:12. Schloss Dagstuhl, 2023.
[10] A. Haddadan, T. Ito, A. E. Mouawad, N. Nishimura, H. Ono, A. Suzuki, and Y. Tebbal. The complexity of dominating set reconfiguration. *Theoretical Computer Science*, 651:37–49, 2016.
[11] R. A. Hearn and E. 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.
[12] T. Ito, E. D. Demaine, N. J. A. Harvey, C. H. Papadimitriou, M. Sideri, R. Uehara, and Y. Uno. On the complexity of reconfiguration problems. *Theoretical Computer Science*, 412(12-14):1054–1065, 2011.
[13] M. Kamiński, P. Medvedev, and M. Milanič. Complexity of independent set reconfigurability problems. *Theoretical Computer Science*, 439:9–15, 2012.
[14] P. Sreenivasa Kumar and C. E. Veni Madhavan. Minimal vertex separators of chordal graphs. *Discrete Applied Mathematics*, 89(1–3):155–168, 1998.
[15] N. Nishimura. Introduction to reconfiguration. *Algorithms*, 11(4):52, 2018.
[16] D. J. Rose. Triangulated graphs and the elimination process. *Journal of Mathematical Analysis and Applications*, 32(3):597–609, 1970.
[17] R. E. Tarjan and M. Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM Journal on Computing*, 13(3):566–579, 1984.