SHORTEST RECONFIGURATION PATHS IN THE SOLUTION SPACE OF BOOLEAN FORMULAS*

AMER E. MOUAWAD†, NAOMI NISHIMURA‡, VINAYAK PATHAK§, AND VENKATESH RAMAN¶

Abstract. Given a Boolean formula and a satisfying assignment, a flip is an operation that changes the value of a variable in the assignment so that the resulting assignment remains satisfying. We study the problem of computing the shortest sequence of flips (if one exists) that transforms a given satisfying assignment s to another satisfying assignment t of an input Boolean formula. Earlier work characterized the complexity of deciding the existence of a sequence of flips between two given satisfying assignments using Schaefer's framework for classification of Boolean formulas. We build on it to provide a trichotomy for the complexity of finding the shortest sequence of flips and show that it is either in P, NP-complete, or PSPACE-complete.

Our result adds to the growing set of complexity results known for shortest reconfiguration sequence problems by providing an example where the shortest sequence can be found in polynomial time even though the sequence flips variables that have the same value in both s and t. This is in contrast to most reconfiguration problems studied so far, where polynomial-time algorithms for computing the shortest path were known only for cases where the path modified no more than the symmetric difference of s and t. Our proof uses Birkhoff's representation theorem on a set system that we show to be a distributive lattice. The technique provides insights and can perhaps be used for other reconfiguration problems as well.

Key words. Reconfiguration, Solution space, Satisfiability, Boolean formulas.

AMS subject classifications. 03D15, 68Q15, 68Q17, 68Q25, 05C40.

1. Introduction.

1.1. Background and Motivation. Reconfiguration problems are motivated by practical situations where one wants to move from one solution of an optimization problem to another while maintaining feasibility in between [7, 14, 19, 24]. Each step of the move is dictated by a *reconfiguration step*, which specifies how one solution can be transformed into another (for example, in case of a graph problem, by adding or deleting a vertex or an edge). Hence, reconfiguration problems can be stated concisely in terms of a graph—the *reconfiguration graph*—that has a node for each feasible solution and an undirected edge between two solutions if one can be formed from the other by a single reconfiguration step. Reconfiguration problems typically study the complexity of finding a path, also known as a *reconfiguration sequence*, between two nodes in the reconfiguration graph [4, 11, 14, 15, 30]. Most reconfiguration versions of NP-complete decision problems are PSPACE-complete [14] (e.g. maximum independent set) and hence recent work addressed the problems under the framework of parameterized complexity [5, 16, 17, 18, 21, 23, 24, 25]. We refer the reader to the survey by van den Heuvel [29] for a detailed overview of reconfiguration problems and their applications.

For the problem of satisfiability of Boolean formulas, one defines a reconfiguration step to be a *flip* operation that changes the value of a variable in a satisfying assign-

*Research supported by the Natural Science and Engineering Research Council of Canada. A preliminary version of this paper appeared in the Proceedings of the 42^{nd} International Colloquium on Automata, Languages, and Programming (ICALP 2015) [22].

†University of Bergen, Bergen, Norway (a.mouawad@uib.no).

‡University of Waterloo, Ontario, Canada (nishi@uwaterloo.ca).

§University of Waterloo, Ontario, Canada (vpathak@uwaterloo.ca).

¶The Institute of Mathematical Sciences, Chennai, India (vraman@imsc.res.in).

ment so that the resulting assignment is also satisfying. Thus, in the reconfiguration graph for satisfiability [13], there is a node for each satisfying assignment and an edge whenever the *Hamming distance* between the two assignments, i.e., the number of variables for which the two assignments differ in value, is exactly one. In earlier results, Gopalan et al. [13] and Schwerdtfeger [28], using Schaefer's [27] framework to classify Boolean formulas, characterized the complexity of determining whether there exists a path between s and t (the *st-connectivity* problem) in the reconfiguration graph. They define a class of formulas called *tight* and show that st-connectivity is in P for tight formulas and is PSPACE-complete otherwise.

1.2. Our results and related work. We study the complexity of computing the *shortest* flip sequence between two satisfying assignments. Since st-connectivity is PSPACE-complete for non-tight formulas, finding the shortest reconfiguration sequence for such formulas is also PSPACE-complete. We show that the class of tight formulas can be further subdivided into *navigable* formulas, where finding the shortest reconfiguration sequence can be done in polynomial time, and tight but non-navigable formulas, where it is NP-complete.

Not many results are known for computing the shortest reconfiguration path except for the cases where the algorithm for st-connectivity returns the shortest path itself [13, 14, 19, 23]. Moreover, the only polynomial-time algorithms known for finding the shortest reconfiguration path have the property that they make no changes to parts of the solution common to s and t. For trees and cactus graphs, the shortest path between maximum independent sets s and t (under the so-called token addition/removal model [23]) never removes vertices in $s \cap t$ [23]. The same holds for the shortest path between independent sets of an even-hole-free graph under the token jumping model [19]. In the sequence of flips for 2CNF formulas (the only class for which a polynomial-time algorithm for shortest reconfiguration path of satisfiability was previously known), the only variables flipped are those whose values are different in s and t [13]. We refer to such variables, i.e., the set of variables assigned different values in s and t, as the *symmetric difference* of s and t.

The problem of computing the shortest reconfiguration sequence between triangulations of a convex polygon is an example where the complexity status has been open for more than 40 years [9]. On the other hand, the problem of determining the existence of a reconfiguration sequence, or equivalently the st-connectivity problem, is trivially solvable as it is known that one can always transform one triangulation of a polygon to another [20]; here the flip operation replaces one diagonal of the given convex polygon with another. There is a lot of work on determining the complexity of finding the shortest reconfiguration sequence for this problem, and it has been settled for some special cases of polygons and point sets [1, 6].

Interestingly, one distinction between the triangulations of convex polygons, for which the complexity of computing the shortest reconfiguration sequence is open, and of simple polygons, for which it is NP-complete, is that the former but not the latter has the property that the shortest flip sequence never flips a diagonal shared by s and t. Insights from our results may lead to a better understanding of the role of the symmetric difference in computing shortest reconfiguration paths.

Another consequence of our results is the identification of a new class of subgraphs of the hypercube (reconfiguration graphs of navigable formulas, as defined in Section 2) where shortest paths can be found efficiently. Partial cubes are subgraphs of the hypercube where the same property holds [10]; however, our class is fundamentally more general than partial cubes in the sense that the distance between two

vertices is not merely the Hamming distance between their labels. Such results are potentially interesting for the area of distance labeling schemes [12, 26] since they provide a new class of graphs that exhibit small distance labels.

2. Preliminaries and definitions. We use terminology originally introduced by Schaefer [27] and adapted for reconfiguration by Gopalan et al. [13] and Schwerdtfeger [28].

A *k-ary Boolean logical relation* (or *relation* for short) R is defined as a subset of $\{0,1\}^k$, where $k \geq 1$. Each $i \in \{1,\ldots,k\}$ can be interpreted as a variable of R so that R specifies exactly which assignments of values to the variables are to be considered satisfying.

For any k-ary relation R and positive integer $k' \leq k$, we define a k'-ary *restriction* of R to be any k'-ary relation R' that can be obtained from R by substitution with constants and identification of variables. More precisely, let $X : \{1,\ldots,k\} \to \{1,\ldots,k'\} \cup \{c_0,c_1\}$ be a bijection from the variables of R to the variables of R' and the constants 0 and 1. Any such X defines a mapping $f_X : \{0,1\}^{k'} \to \{0,1\}^k$ as follows. For $r \in \{0,1\}^{k'}$, let $f_X(r)$ be the k-bit vector whose i^{th} bit is 0 if $X(i) = c_0$, 1 if $X(i) = c_1$, and equal to the $X(i)^{th}$ bit of r otherwise. We say that a k'-ary relation R' is a restriction of R with respect to $X : \{1,\ldots,k\} \to \{1,\ldots,k'\} \cup \{c_0,c_1\}$ if $r \in R' \Leftrightarrow f_X(r) \in R$.

A Boolean formula ϕ over a set $\{x_1,\ldots,x_n\}$ of variables defines a relation R_ϕ as follows. For any n-bit vector $v \in \{0,1\}^n$, we interpret v as the assignment to the variables of ϕ where x_i is set to be equal to the i^{th} bit of v. We then say that $v \in R_\phi$ if and only if v is a satisfying assignment.

A *CNF formula* ϕ is a Boolean formula of the form $C_1 \wedge \ldots \wedge C_m$, where each C_i, $1 \leq i \leq m$, is a *clause* consisting of a finite disjunction of *literals* (variables or negated variables). A *kCNF formula*, $k \geq 1$, is a CNF formula where each clause has at most k literals. A CNF formula is *Horn* (*dual Horn*) if each clause has at most one positive (negative) literal.

For a finite set of relations $\mathcal{S}$, a *CNF($\mathcal{S}$) formula* ϕ over a set of n variables $\{x_1,\ldots,x_n\}$ is a finite collection $\{C_1,\ldots,C_m\}$ of clauses. Each C_i, $1 \leq i \leq m$, is defined by a tuple (R_i, X_i), where R_i is a k_i-ary relation in $\mathcal{S}$ and $X_i : \{1,\ldots,k_i\} \to \{1,\ldots,n\} \cup \{c_0,c_1\}$ is a function. Each X_i defines a mapping $f_{X_i} : \{0,1\}^n \to \{0,1\}^{k_i}$ and we say that an assignment v to the variables satisfies ϕ if and only if for all $i \in \{1,\ldots,m\}$, $f_{X_i}(v) \in R_i$. For any variable x_j, we say that x_j *appears in* clause C_i if $X_i(q) = j$ for some $q \in \{1,\ldots,k_i\}$ and for any assignment v to the variables of ϕ, we say that $f_{X_i}(v)$ is the assignment induced by v on R_i.

For example, to represent the class 3CNF in Schaefer's framework, we specify $\mathcal{S}$ as follows. Let $R^0 = \{0,1\}^3 \backslash \{000\}$, $R^1 = \{0,1\}^3 \backslash \{100\}$, $R^2 = \{0,1\}^3 \backslash \{110\}$, $R^3 = \{0,1\}^3 \backslash \{111\}$, and $\mathcal{S} = \{R^0, R^1, R^2, R^3\}$. Since R^i can be used to represent all 3-clauses with exactly i negative literals (regardless of the positions in which they appear in a clause), clearly CNF($\mathcal{S}$) is exactly the class of 3CNF formulas. Similarly, setting $\mathcal{S} = \{R_{1/3}\}$, where $R_{1/3} = \{100, 010, 001\}$, one can represent the class of formulas where exactly one variable must be set to true in each clause to obtain a satisfying assignment (also known as the Positive 1-in-3 Sat problem).

Below we define some classes of relations used in the literature and relevant to our work. Note that componentwise bijunctive, OR-free, and NAND-free relations were first defined by Gopalan et al. [13]. Schwerdtfeger [28] later modified them slightly and defined safely componentwise bijunctive, safely OR-free, and safely NAND-free relations. We reuse the names componentwise bijunctive, OR-free, and NAND-free for

Schwerdtfeger's safely componentwise bijunctive, safely OR-free, and safely NAND-free relations, respectively.

Definition 2.1. *For a k-ary relation R:*

- *R is* bijunctive *if it is the set of satisfying assignments of a 2CNF formula.*
- *R is* Horn *(*dual Horn*) if it is the set of satisfying assignments of a Horn (dual Horn) formula.*
- *R is* affine *if it is the set of satisfying assignments of a formula $x_{i_1} \oplus \ldots \oplus x_{i_h} \oplus c$, with $i_1, \ldots, i_h \in \{1, \ldots, k\}$ and $c \in \{0, 1\}$. Here $\oplus$ denotes the exclusive binary OR operation which evaluates to 1 when exactly one of the operands is 1 and evaluates to 0 otherwise.*
- *R is* componentwise bijunctive *if every connected component of the reconfiguration graph of R and of the reconfiguration graph of every restriction R' of R induces a bijunctive relation.*
- *R is* OR-free *(*NAND-free*) if there does not exist a restriction R' of R such that $R' = \{01, 10, 11\}$ ($R' = \{01, 10, 00\}$).*

Using his framework, Schaefer showed that SAT($\mathcal{S}$)—the problem of deciding if a given CNF($\mathcal{S}$) formula has a satisfying assignment—is in P if every relation in $\mathcal{S}$ is bijunctive, Horn, dual Horn, or affine, and is NP-complete otherwise. The result is remarkable because it divides a large set of problems into two equivalence classes based on their computational complexity, which is the opposite of what one might expect due to Ladner's theorem [2].

Since Schaefer's original paper, a myriad of problems about Boolean formulas have been analyzed, and similar divisions into equivalence classes obtained [8]. Gopalan et al.'s work [13], with corrections presented by Schwerdtfeger [28], shows a dichotomy for the problem of deciding whether a reconfiguration path exists between two satisfying assignments of a given CNF($\mathcal{S}$) formula. They showed that the st-connectivity problem on CNF($\mathcal{S}$) formulas is in P if $\mathcal{S}$ is tight (Definition 2.2) and PSPACE-complete otherwise.

Definition 2.2. *A set $\mathcal{S}$ of relations is called* tight *if at least one of the following holds:*

1. *All relations in $\mathcal{S}$ are OR-free.*
2. *All relations in $\mathcal{S}$ are NAND-free.*
3. *All relations in $\mathcal{S}$ are componentwise bijunctive.*

Our trichotomy relies on a new class of formulas that subdivides the set of tight relations into those for which computing the shortest reconfiguration path can be done in polynomial time and those for which it is NP-complete.

Definition 2.3. *For a k-ary relation R:*

- *R is* Horn-free *if there does not exist a restriction R' of R such that $R' = \{0, 1\}^3 \setminus \{011\}$, or equivalently, R' is the set of all satisfying assignments of the clause $(x \vee \overline{y} \vee \overline{z})$ for some three variables x, y, and z.*
- *R is* dual-Horn-free *if there does not exist a restriction R' of R such that $R' = \{0, 1\}^3 \setminus \{100\}$, or equivalently, R' is the set of all satisfying assignments of the clause $(\overline{x} \vee y \vee z)$ for some three variables x, y, and z.*

The following is a useful observation.

Observation 1. *For $k \geq 3$ and R a k-ary relation, if R is OR-free then it is dual-Horn-free. Similarly, if R is NAND-free then it is Horn-free.*

Proof. Assume that R is OR-free but not dual-Horn-free. Then there exists a restriction R' of R such that $R' = \{0, 1\}^3 \setminus \{100\}$. It is easy to see that, from R', one can obtain $R'' = \{01, 10, 11\}$ by setting one of the three variables in R' to 1, resulting

in a contradiction. A similar proof shows that NAND-free relations are Horn-free. □

DEFINITION 2.4. *We call a set $\mathcal{S}$ of relations* navigable *if at least one of the following holds:*

1. *All relations in $\mathcal{S}$ are OR-free and Horn-free.*
2. *All relations in $\mathcal{S}$ are NAND-free and dual-Horn-free.*
3. *All relations in $\mathcal{S}$ are componentwise bijunctive.*

It is clear that if $\mathcal{S}$ is navigable, then it is also tight. Moreover, it follows from Observation 1 that when all relations in $\mathcal{S}$ are OR-free and Horn-free (NAND-free and dual-Horn-free) then they are also dual-Horn-free (Horn-free). Our main result is the following trichotomy.

THEOREM 2.5. *For a CNF($\mathcal{S}$) formula ϕ and two satisfying assignments s and t, the problem of computing the shortest reconfiguration path between s and t is in* P *if $\mathcal{S}$ is navigable,* NP*-complete if $\mathcal{S}$ is tight but not navigable, and* PSPACE*-complete otherwise.*

In the next section, we establish the hardness results; the rest of the paper focuses on our polynomial-time algorithm for navigable formulas. Interestingly, unlike previous classification results, while the NP-completeness result in our case turns out to be relatively easy, the polynomial-time algorithm is quite involved.

3. The hard cases. Gopalan et al. [13] showed that if $\mathcal{S}$ is not tight, then st-connectivity is PSPACE-complete for CNF($\mathcal{S}$) formulas. This implies that finding the shortest reconfiguration path is also PSPACE-complete for such classes of formulas.

THEOREM 3.1. *If $\mathcal{S}$ is tight but not navigable, then finding the shortest reconfiguration path on CNF($\mathcal{S}$) formulas is* NP*-complete.*

Proof. The problem is in NP because the diameter of the reconfiguration graph is polynomial for all tight formulas (CNF($\mathcal{S}$) formulas where all clauses are associated with relations belonging to the set $\mathcal{S}$ of tight relations), as shown by Gopalan et al. [13]. We now prove that it is, in fact, NP-complete.

As $\mathcal{S}$ is tight but not navigable, it does not hold that all relations in $\mathcal{S}$ are componentwise bijunctive. Hence, either all relations in $\mathcal{S}$ are OR-free or all relations in $\mathcal{S}$ are NAND-free. Let us first assume that all relations in $\mathcal{S}$ are NAND-free. Then, as $\mathcal{S}$ is not navigable, there exists a relation R in $\mathcal{S}$ that is not dual-Horn-free. We show a reduction from VERTEX COVER to such a CNF($\mathcal{S}$) formula. If the arity of R is greater than three then, by Definition 2.3, there must exist a restriction R' of R such that $R' = \{0,1\}^3 \setminus \{100\}$, or equivalently, R' is the set of all satisfying assignments of the clause $(\overline{x} \vee y \vee z)$ for some three variables x, y, and z. Since substitution with constants and identification of variables is allowed, we can assume, without loss of generality, that $R' \in \mathcal{S}$.

Given an instance $(G = (V, E), k)$ of VERTEX COVER, we create a variable x_v for each $v \in V$. For each edge $e = (u, v) \in E$, we create two new variables y_e and z_e and the clauses $(\overline{z_e} \vee y_e \vee x_u)$ and $(\overline{y_e} \vee z_e \vee x_v)$. The resulting formula $F(G)$ has $|V| + 2|E|$ variables and $2|E|$ clauses.

It is easy to see that all clauses of $F(G)$ are associated with R'. Hence the formula $F(G)$ is a CNF($\mathcal{S}$) formula, where $\mathcal{S}$ is tight but not navigable.

Let s be the satisfying assignment for the formula with all variables set to 0, and let t be the satisfying assignment with all the variables x_v, $v \in V$, set to 0 and the rest set to 1. If G has a vertex cover S of size at most k, then we can form a reconfiguration sequence of length at most $2|E| + 2k$ from s to t by flipping each x_v, $v \in S$, from 0 to 1, flipping the y_e and z_e variables, and then flipping each x_v, $v \in S$, back from 1 to 0. To show that such a reconfiguration sequence exists only if there exists such a vertex

cover, we observe that if neither x_u nor x_v has been flipped to 1, neither y_e nor z_e can be flipped to 1 while keeping the formula satisfied at the intermediate steps.

To show hardness when all relations in $\mathcal{S}$ are OR-free but not Horn-free, we again give a reduction from Vertex Cover. Given $G = (V, E)$ and an integer k, we create a variable x_v for each $v \in V$ and two variables y_e and z_e for each $e \in E$. For each edge $e = (u, v) \in E$, we create the clauses $(y_e \vee \overline{z_e} \vee \overline{x_u})$ and $(\overline{y_e} \vee z_e \vee \overline{x_v})$. Clearly, all the relations of the formula are OR-free, and none of them is Horn-free.

We let s be the satisfying assignment that sets all the variables to 1, and t be the satisfying assignment that sets all the variables to 0 except the variables x_v, $v \in V$, which are set to 1. If G has a vertex cover S of size at most k, we can form a reconfiguration sequence of length at most $2|E| + 2k$ from s to t by flipping each x_v, $v \in S$, from 1 to 0, flipping the y_e and z_e variables, and then flipping each x_v, $v \in S$, back from 0 to 1. To show that such a reconfiguration sequence exists only if there exists such a vertex cover (of size k), we observe that if neither x_u nor x_v has been flipped to 0, neither y_e nor z_e can be flipped to 0 while keeping the formula satisfied at the intermediate steps. □

4. The polynomial-time algorithm for navigable formulas. In this section, we give a polynomial-time algorithm to find a shortest reconfiguration sequence between two satisfying assignments of a navigable formula. Gopalan et al. gave a polynomial-time algorithm for finding a shortest reconfiguration path in componentwise bijunctive formulas. The path, in this case, flips only variables that have different values in s and t. The NP-completeness proof from the previous section crucially relies on the fact that we need to flip variables with common values; in fact, the hardness lies in deciding precisely which common variables need to be flipped. Thus it is tempting to conjecture that hardness for shortest reconfiguration path is caused by relations where the shortest distance is not always equal to the Hamming distance. This is not the case. The reconfiguration graph for the relation $P_5 = \{000, 001, 101, 111, 110\}$ is a 5-vertex path, where for 000 and 110 the shortest path (measured in number of edges) is of length four but the Hamming distance is two. However, we can find the shortest reconfiguration paths in formulas built out of P_5 in polynomial time, the exact reason for which will become clear in our general description of the algorithm. The intuitive reason is that there are very few candidates for shortest paths; if we restrict our attention to a single clause built out of P_5, then there exists a unique path to follow. It then suffices to determine whether there exist two clauses for which the prescribed paths are in conflict. In general, our proof relies on showing that even if there does not exist a unique path, the set of all possible paths between two satisfying assignments of a navigable formula is not diverse enough to make the problem computationally hard. We show that the set of all possible paths can be characterized using a partial order on the set of flips.

4.1. Notation. Our results make use of two different views of the problem (graph-theoretic and algebraic), and hence two sets of notation.

The graph-theoretic view consists of the reconfiguration graph G_R that has a node for each Boolean string $s \in R$ and an edge whenever the Hamming distance between the two strings is exactly one. We call a path from s to t *monotonically increasing* if the Hamming weights of the vertices on the path increase monotonically as we go from s to t, and define a *monotonically decreasing* path similarly. A path is *canonical* if it consists of a monotonically increasing path followed by a monotonically decreasing path.

The algebraic view consists of a *token system* [10] consisting of a set $\mathcal{S}$ of states and a set τ of tokens. The tokens specify the rules of transition between states. Each token $t \in \tau$ is a function that maps $\mathcal{S}$ to itself. Given a k-ary relation R, we define a token system as follows. The set $\mathcal{S}$ of states consists of all the elements of R and a special state s^* called the *invalid state* that captures all the unsatisfying assignments of the formula. The set τ of tokens is the set $\{x_1^+, \ldots, x_k^+\} \cup \{x_1^-, \ldots, x_k^-\}$, where x_i^+ denotes a flip of variable x_i from 0 to 1, which we call a *positive flip*, and consider the sign of the flip as positive, and x_i^- denotes a flip of variable x_i from 1 to 0, which we call a *negative flip* and consider the sign of the flip as negative.

To complete the description of the token system, we need to specify the function to which each token corresponds. For $x_i^+ \in \tau$ and $s \in \mathcal{S} \setminus \{s^*\}$, $x_i^+(s^*) = s^*$, $x_i^+(s) = s'$ if the value of variable x_i in s is 0 and the bit string s' obtained by flipping x_i to 1 lies in R, and $x_i^+(s) = s^*$ if the value of variable x_i in s is 1 or the value of variable x_i in s is 0 and the bit string s' obtained by flipping x_i to 1 does not lie in R. The function x_i^- is defined analogously. In the rest of this article, we will use the word "flip" instead of "token", and we will use the words "state", "vertex", and "satisfying assignment" interchangeably.

A sequence of flips also defines a function, that is, the composition of all the functions in the sequence. We call a flip sequence *invalid* at a given state s if the sequence applied to s results in invalid state s^*, and *valid* otherwise. Two flip sequences are *equivalent* if they result in the same final state when applied to the same starting state. Finally, we call a flip sequence *canonical* if all positive flips in it occur before all the negative flips. That is, the path from its first state (node) to the last is a canonical path. Note that in any canonical flip sequence, each flip occurs at most once. Given two states $s, t \in \mathcal{S}$, we say that a (multi) set $\mathcal{C}$ of flips (hereafter called *flip (multi) set*) *transforms s to t* if the elements of $\mathcal{C}$ can be arranged in some order such that the resulting flip sequence transforms s to t. For a given state s and flip (multi) set $\mathcal{C}$, we say $\mathcal{C}$ is *valid* if the elements of $\mathcal{C}$ can be arranged in some order such that the resulting flip sequence applied to s results in a valid state.

We describe a flip sequence simply by listing the flips in order. The flip sequence formed by removing flip f from $\mathcal{F}$ is denoted $\mathcal{F} \setminus f$. The flip sequence obtained by reversing $\mathcal{F}$ is $\mathcal{F}^{-1}$, and by performing $\mathcal{F}_1$ followed by $\mathcal{F}_2$ is $\mathcal{F}_1 \cdot \mathcal{F}_2$. We use $\mathcal{C}(\mathcal{F})$ to denote the (multi) set of flips that appear in $\mathcal{F}$. A flip sequence (set) consisting of only positive flips will be called a *positive flip sequence (set)*. We use $\mathcal{F}_0$ to denote an empty flip sequence and, by convention, define it to be valid. For a flip sequence $\mathcal{F}$, if $f \in \mathcal{F}$ appears before $f' \in \mathcal{F}$ in the sequence, then we say $f <_{\mathcal{F}} f'$. For a tuple $t = (x_{i_1}, \ldots, x_{i_d})$ of variables and a state s, we use s^t to denote the string of values of s restricted to $x_{i_1}, \ldots, x_{i_d}$. Note that for any valid positive flip sequence or canonical sequence $\mathcal{F}$, $\mathcal{C}(\mathcal{F})$ is a set (and not just a multiset). Similarly, if $\mathcal{C}$ consists of only positive flips then in order for $\mathcal{C}$ to be valid it must be a set (and not just a multiset).

4.2. Overview of the algorithm. Consider, once again, the set $\{P_5\}$, where $P_5 = \{000, 001, 101, 111, 110\}$, from Section 4, which we claimed to be navigable. A satisfying assignment to a CNF($\{P_5\}$) formula induces, on each clause, a Boolean string that consists of the values of the variables appearing in that clause. Similarly, a flip sequence $\mathcal{F}$ induces a flip sequence for each clause C, which is the subsequence of $\mathcal{F}$ that flips a variable that appears in C. Note that $\mathcal{F}$ is valid if and only if the sequence induced on each clause is valid. The relation P_5 satisfies two nice properties:

1. Any valid flip sequence of P_5 is canonical.
2. Let x, y, z be the three variables that represent the three bits of P_5. Then

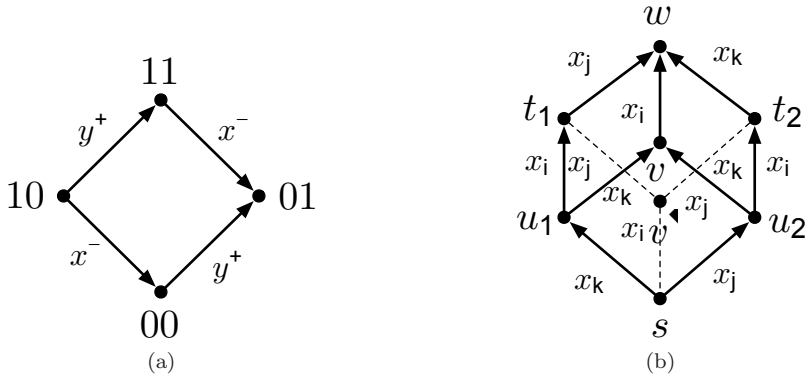

FIG. 1. *(a) Example for Lemma 4.1 (b) Example for Lemma 4.5*

there is a total order, namely, $z^+ < x^+ < y^+$, such that any valid positive flip sequence must satisfy this order.

With this observation, formulating an algorithm is easy. Given two satisfying assignments s and t of the formula, find the Boolean string induced on each clause. The shortest flip sequence inside each clause can be computed in constant time. Each clause prescribes a unique order in which the flips of its corresponding flip sequence must be performed. If no two clauses prescribe conflicting orders, then their sequences can be combined into a sequence for the entire formula. If there is a conflict, then we know that no path exists. In general, for navigable formulas, we show, in Lemma 4.2, that any flip sequence can be transformed into an equivalent canonical flip sequence by rearranging the flips, and, in Lemma 4.9, that each clause prescribes a partial order instead of a total order. The task of combining the partial orders prescribed by each clause becomes more involved, but can still be done efficiently, as shown in Lemma 4.11.

We will consider properties only of NAND-free and dual-Horn-free relations, as our algorithm for NAND-free and dual-Horn-free relations can easily be modified to handle OR-free and Horn-free relations (by "reversing" the roles of positive and negative flips).

4.3. The token system of NAND-free relations. We begin by proving some useful properties of the token system formed by NAND-free relations.

LEMMA 4.1. *For a NAND-free relation R and $\mathcal{F} = f_1 \dots f_q$ a valid flip sequence at $s \in R$, if there exists $i \in \{1, \dots, q-1\}$ such that $f_i = x^-$ is a negative flip and $f_{i+1} = y^+$ is a positive flip, with $x \neq y$, then the sequence $\mathcal{F}' = f_1 \dots f_{i-1} f_{i+1} f_i \dots f_q$ is also valid at s and is equivalent to $\mathcal{F}$, i.e., swapping f_i and f_{i+1} results in an equivalent flip sequence.*

Proof. Let u be the state just before applying f_i in $\mathcal{F}$, $v = f_i(u)$ be the state after applying f_i but before applying f_{i+1}, and $w = f_{i+1}(v)$ be the one after applying f_{i+1}. It is clear that $u^{(x,y)} = 10$, $v^{(x,y)} = 00$, and $w^{(x,y)} = 01$. Also, since no other variables are flipped between u, v, and w, the values of all variables other than x and y remain the same in the states u, v, and w.

Let t be the Boolean string whose value is the same as u, v, and w on all variables except x and y and $t^{(x,y)} = 11$. If $t \notin R$, then the substitution described above gives us the relation $\{10, 00, 01\}$ on x and y, which is precisely the NAND relation. Since R is NAND-free, $t \in R$ (Figure 1(a)) and thus we can replace the path $u \to v \to w$

with the path $u \to t \to w$. This is equivalent to swapping the flips f_{i+1} and f_i. □

Lemma 4.2 (first proved by Gopalan et al. [13]) shows that any valid flip sequence can be made canonical. Lemma 4.3 shows that the union of two valid positive flips sets is also a valid flip set.

LEMMA 4.2. *For R a NAND-free relation, if $\mathcal{F}$ is a valid sequence at $s \in R$, then there exists a valid canonical sequence $\mathcal{F}'$ equivalent to $\mathcal{F}$ such that $\mathcal{C}(\mathcal{F}') \subseteq \mathcal{C}(\mathcal{F})$ and, for any two flips $f_1, f_2 \in \mathcal{F}'$ of the same sign, if $f_1 <_{\mathcal{F}'} f_2$ then $f_1 <_{\mathcal{F}} f_2$, i.e., the relative order among flips of the same sign is preserved.*

Proof. If $\mathcal{F}$ is not canonical, it must have a negative flip followed by a positive flip somewhere. If both flips act on the same variable, we cancel them out; otherwise, we swap them using the proof of Lemma 4.1. Doing this repeatedly gives us the required canonical sequence $\mathcal{F}'$. The order among the flips of the same sign is preserved since we never swap two flips of the same sign. □

LEMMA 4.3. *For R a NAND-free relation, if $\mathcal{C}_1$ and $\mathcal{C}_2$ are two positive flip sets that are valid at $s \in R$, then $\mathcal{C}_1 \cup \mathcal{C}_2$ is also a valid flip set at s.*

Proof. Let $u = \mathcal{F}_1(s)$ and $v = \mathcal{F}_2(s)$, where $\mathcal{F}_1$ and $\mathcal{F}_2$ are valid flip sequences such that $\mathcal{C}(\mathcal{F}_1) = \mathcal{C}_1$ and $\mathcal{C}(\mathcal{F}_2) = \mathcal{C}_2$. Clearly, $\mathcal{F}_1^{-1} \cdot \mathcal{F}_2$ is a valid flip sequence from u to v. Thus, we can apply Lemma 4.2 to the sequence $\mathcal{F}_1^{-1} \cdot \mathcal{F}_2$ to transform it into a canonical sequence $\mathcal{F}$. Let $\mathcal{F}^+$ denote the prefix of $\mathcal{F}$ that contains all the positive flips. It is clear that $\mathcal{F}_1 \cdot \mathcal{F}^+$ is a valid flip sequence at s and $\mathcal{C}(\mathcal{F}_1 \cdot \mathcal{F}^+) \subseteq \mathcal{C}_1 \cup \mathcal{C}_2$. To see why $\mathcal{C}_1 \cup \mathcal{C}_2 \subseteq \mathcal{C}(\mathcal{F}_1 \cdot \mathcal{F}^+)$, it is enough to note that any flip in $\mathcal{C}_2$ which does not appear in $\mathcal{C}(\mathcal{F}^+)$ must belong to $\mathcal{C}_1 \cap \mathcal{C}_2 \subseteq \mathcal{C}(\mathcal{F}_1)$ (as the canonical sequence $\mathcal{F}$ only cancels out flips acting on the same variable). Therefore, $\mathcal{C}(\mathcal{F}_1 \cdot \mathcal{F}^+) = \mathcal{C}_1 \cup \mathcal{C}_2$, as needed. □

Later, we prove a similar lemma for the intersection of two flip sets, but for dual-Horn-free relations. We conclude this subsection with a lemma that shows that if two disjoint flips sets are valid at a state, we can, in some sense, perform the two sets of flips one after the other in either order.

LEMMA 4.4. *For R a NAND-free relation and $\mathcal{F}_1$ and $\mathcal{F}_2$ two positive flip sequences that are valid at $s \in R$, if $\mathcal{C}(\mathcal{F}_1) \cap \mathcal{C}(\mathcal{F}_2) = \emptyset$, then $\mathcal{F}_1$ is valid at $\mathcal{F}_2(s)$ and $\mathcal{F}_2$ is valid at $\mathcal{F}_1(s)$.*

Proof. Consider the sequence $\mathcal{F}_2^{-1} \cdot \mathcal{F}_1$ that transforms $\mathcal{F}_2(s)$ to $\mathcal{F}_1(s)$. Applying Lemma 4.2 to it, we obtain the canonical flip sequence $\mathcal{F}_1 \cdot \mathcal{F}_2^{-1}$. Thus $\mathcal{F}_1$ is valid at $\mathcal{F}_2(s)$. Using the same argument on the sequence $\mathcal{F}_1^{-1} \cdot \mathcal{F}_2$ proves the other claim. □

4.4. The token system of NAND-free and dual-Horn-free relations. In this section, we establish stronger properties with the assumption that R is not only NAND-free, but is also dual-Horn-free. We begin by establishing a simple property of relations that are NAND-free and dual-Horn-free.

LEMMA 4.5. *Let R be a NAND-free and dual-Horn-free relation and $s, t_1, t_2 \in R$ be three distinct states such that the flip sequence $\mathcal{F}_1 = x_k^+ x_i^+$ transforms s to t_1, the flip sequence $\mathcal{F}_2 = x_j^+ x_i^+$ transforms s to t_2, and $x_k \neq x_j$. Then the sequence $\mathcal{F}_1' = x_i^+ x_k^+$ also transforms s to t_1 and the sequence $\mathcal{F}_2' = x_i^+ x_j^+$ also transforms s to t_2, i.e., we can swap the flips in both $\mathcal{F}_1$ and $\mathcal{F}_2$.*

Proof. For $u_1 = x_k^+(s)$ and $u_2 = x_j^+(s)$, the sequence $x_j^- x_k^+$ transforms u_2 to u_1. We can reorder the sequence to obtain $x_k^+ x_j^-$, using Lemma 4.1. For $v = x_k^+(u_2)$, we can use a similar argument to show that x_i^+ is a valid flip at v; we let $w = x_i^+(v)$. The values of variables x_i, x_j, and x_k at states s, u_1, u_2, t_1, t_2, v, and w form exactly the seven satisfying assignments $\{000, 001, 010, 101, 110, 011, 111\}$ of the dual-Horn clause $(\overline{x_i} \vee x_j \vee x_k)$ (Figure 1(b)). But since R is dual-Horn-free, there must also

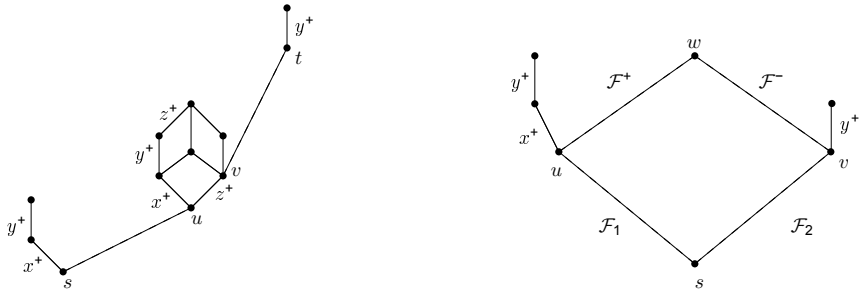

FIG. 2. *Dotted lines denote paths and solid lines denote edges. Hamming weight increases in the upward direction. (a) Proof of Lemma 4.6 (b) Proof of Lemma 4.7.*

exist the state v' for which $x_i = 1, x_j = 0, x_k = 0$. The path $s \to v' \to t_1$ gives the sequence $x_i^+ x_k^+$ and the path $s \to v' \to t_2$ gives the sequence $x_i^+ x_j^+$. □

The seemingly innocuous lemma above turns out to be very powerful. In the following sequence of lemmas, we build on it to eventually prove that the set of all positive valid flip sets starting from an assignment s forms a distributive lattice. The lattice structure then helps us formulate a polynomial-time algorithm for computing the shortest reconfiguration path.

LEMMA 4.6. *Let R be a NAND-free and dual-Horn-free relation and $s, t \in R$ be two satisfying assignments such that x^+y^+ is a valid flip sequence at s and y^+ is a valid flip at t. Furthermore, let $\mathcal{F}$ be a positive flip sequence such that $\mathcal{F}(s) = t$ and $x^+ \notin \mathcal{C}(\mathcal{F})$. Then, the sequence y^+x^+ must also be valid at s.*

Proof. Let v be the vertex with smallest Hamming weight on the path corresponding to $\mathcal{F}$ from s to t (including s and t) at which y^+ is a valid flip. Let $\mathcal{F}_1 = x^+y^+$ and let $\mathcal{F}_2$ be the positive flip sequence that transforms s to v, i.e., $v = \mathcal{F}_2(s)$. Note that $\mathcal{C}(\mathcal{F}_1) \cap \mathcal{C}(\mathcal{F}_2) = \emptyset$, as neither x^+ nor y^+ can appear in $\mathcal{C}(\mathcal{F}_2)$ (Figure 2(a)).

If $v = s$, we are done; then let us assume this not to be the case. Let u be the vertex immediately before v on the path from s to t and let $z^+(u) = v$. Since $\mathcal{C}(\mathcal{F}_1) \cap \mathcal{C}(\mathcal{F}_2) = \emptyset$ and $\mathcal{C}(\mathcal{F}_1) \cap (\mathcal{C}(\mathcal{F}_2) \setminus \{z^+\}) = \emptyset$, we can apply Lemma 4.4 at s, which implies that x^+y^+ must be valid at both u and v. Now we use Lemma 4.5 at u. Since both x^+y^+ and z^+y^+ are valid sequences at u, y^+x^+ must also be a valid sequence at u. This contradicts the assumption that v was the vertex with smallest Hamming weight on the path where y^+ was a valid flip. □

LEMMA 4.7. *For a NAND-free and dual-Horn-free relation R, if $\mathcal{F}_1 \cdot x^+ \cdot y^+$ and $\mathcal{F}_2 \cdot y^+$ are both valid positive flip sequences at $s \in R$ such that $x^+ \notin \mathcal{C}(\mathcal{F}_2)$ then $\mathcal{F}_1 \cdot y^+ \cdot x^+$ is also valid at s.*

Proof. Let $u = \mathcal{F}_1(s)$ and $v = \mathcal{F}_2(s)$. We apply Lemma 4.2 to the sequence $\mathcal{F}_1^{-1} \cdot \mathcal{F}_2$ that transforms u to v to obtain a canonical sequence $\mathcal{F} = \mathcal{F}^+ \cdot \mathcal{F}^-$. Let w be the vertex with maximum Hamming weight on this canonical path (Figure 2(b)).

Hence, we have $w = \mathcal{F}^+(u)$ and $v = \mathcal{F}^-(w)$. Note that $\mathcal{F}$ does not involve flips of the variables x or y. Since y^+ is a valid flip at v, $y^+ \notin \mathcal{C}(\mathcal{F}^-)$, and the path from v to w is monotonically increasing, from Lemma 4.4, y^+ is also valid at w. Now using Lemma 4.6, since x^+y^+ is valid at u, $x^+ \notin \mathcal{F}^+$, and y^+ is valid at w, y^+x^+ is also valid at u. □

Next, we show that the set of valid flip sets is closed under intersection.

LEMMA 4.8. *For a NAND-free and dual-Horn-free relation R, if $\mathcal{C}_1$ and $\mathcal{C}_2$ are*

two positive flip sets that are valid at $s \in R$, then $\mathcal{C}_1 \cap \mathcal{C}_2$ is also a valid flip set at s.

Proof. If $\mathcal{C}_1 \subseteq \mathcal{C}_2$ or $\mathcal{C}_2 \subseteq \mathcal{C}_1$, then the statement is trivial. Otherwise, consider any valid ordering $\mathcal{F}_1$ of $\mathcal{C}_1$. We show that if x^+ and y^+ are two consecutive elements of $\mathcal{F}_1$ such that $x^+ \in \mathcal{C}_1 \backslash \mathcal{C}_2$, $y^+ \in \mathcal{C}_1 \cap \mathcal{C}_2$, and $x^+ <_{\mathcal{F}_1} y^+$, then swapping x^+ and y^+ also gives a valid ordering of $\mathcal{C}_1$. Applying such swaps repeatedly, we get an ordering where all elements of $\mathcal{C}_1 \cap \mathcal{C}_2$ appear before all elements of $\mathcal{C}_1 \backslash \mathcal{C}_2$, thus proving that $\mathcal{C}_1 \cap \mathcal{C}_2$ is a valid set at s.

To see how to swap x^+ and y^+ in $\mathcal{F}_1$, suppose u is the vertex on the path corresponding to $\mathcal{F}_1$ on which the sequence $x^+ \cdot y^+$ is performed, and consider an arbitrary valid ordering $\mathcal{F}_2$ of $\mathcal{C}_2$. Let v be the vertex on the path corresponding to $\mathcal{F}_2$ on which y^+ is performed. Such a vertex exists since $y^+ \in \mathcal{C}_1 \cap \mathcal{C}_2$. Now, since $x^+ \cdot y^+$ is valid at u, y^+ is valid at v and the monotonically increasing path from s to v does not contain the flip x^+ (since $x^+ \in \mathcal{C}_1 \backslash \mathcal{C}_2$), by applying Lemma 4.7 we can swap y^+ and x^+ in $\mathcal{F}_1$. □

The above lemma, combined with Lemma 4.3, shows that the set of valid positive flip sets starting at s forms a distributive lattice [3]. Using Birkhoff's representation theorem [3] on it directly implies the next lemma. However, for clarity, we also provide an independent proof. Let $\prec$ be a transitive and asymmetric binary relation defined on a set $\mathcal{C}$ of flips. We call $\prec$ a *strict partial order* defined on $\mathcal{C}$. We say a set $\mathcal{C}' \subseteq \mathcal{C}$ is *downward closed* if for every $x, y \in \mathcal{C}$ $(x \neq y)$, $(y \in \mathcal{C}') \wedge (x \prec y) \implies x \in \mathcal{C}'$. We say that an ordering $\mathcal{F}$ of a subset of elements in $\mathcal{C}$ *obeys* the strict partial order $\prec$ if (i) $\mathcal{C}(\mathcal{F})$ is downward closed and (ii) for every $x, y \in \mathcal{C}(\mathcal{F})$ $(x \neq y)$, $x \prec y \implies x <_{\mathcal{F}} y$.

LEMMA 4.9. *Let R be a NAND-free and dual-Horn-free relation and s be an element of R. Let $\mathcal{P} = \{x^+ \mid x^+ \in \mathcal{C}$ for a positive valid flip set $\mathcal{C}$ at $s\}$. Then there exists a strict partial order $\prec$ on $\mathcal{P}$ such that any positive flip sequence $\mathcal{F}$, where $\mathcal{C}(\mathcal{F}) \subseteq \mathcal{P}$, is a valid flip sequence at s if and only if it obeys $\prec$.*

Proof. Our proof proceeds by providing an explicit strict partial order $\prec$ on the flips in $\mathcal{P}$. For $x^+, y^+ \in \mathcal{P}$ and $x^+ \neq y^+$, let $x^+ \prec y^+$ if and only if all valid positive flip sequences $\mathcal{F}$ starting at s that contain y^+ also contain x^+. Note that this implies that for each such sequence $\mathcal{F}$ we have $x^+ <_{\mathcal{F}} y^+$; if $y^+ <_{\mathcal{F}} x^+$, then stopping this sequence at y^+ would produce a valid positive flip sequence which contains y^+ but not x^+ (and hence $x^+ \not\prec y^+$). This order is clearly a strict partial order since if $x^+ \prec y^+$ and $y^+ \prec z^+$, then $x^+ \prec z^+$ (asymmetry holds trivially).

From the definition of the strict partial order, it is clear that every valid flip sequence must obey the strict partial order. For the other direction, consider a flip sequence $\mathcal{F}^*$ that obeys the strict partial order. We will show that $\mathcal{F}^*$ is valid by induction on the length of the flip sequence. Before doing so, we need the following claim.

CLAIM 1. *Let $\mathcal{F}_{i-1} = (f_1, \ldots, f_{i-1})$ be a valid positive flip sequence at s and let $\mathcal{C} = \mathcal{C}(\mathcal{F}_{i-1}) \cup \{f_i\}$, $f_i \notin \mathcal{C}(\mathcal{F}_{i-1})$, be a valid positive flip set at s. Then $\mathcal{F}_i = \mathcal{F}_{i-1} \cdot f_i$ is a valid positive flip sequence at s.*

Proof. Let $u = \mathcal{F}_{i-1}(s)$. Given that $\mathcal{C}$ is a valid positive flip set at s, there exists a valid ordering of $\mathcal{C}$ resulting in some valid state v. Since f_i is a positive flip and $f_i \notin \mathcal{C}(\mathcal{F}_{i-1})$, $f_i(u) = v$ and hence $\mathcal{F}_{i-1} \cdot f_i$ is a valid positive flip sequence at s. This completes the proof of the claim. □

For the base case, $\mathcal{F}^*$ is trivially valid when $|\mathcal{F}^*| = 0$. As the induction hypothesis, suppose that any flip sequence of length $i-1$ obeying the strict partial order is valid. Consider a flip sequence $\mathcal{F}^* = (f_1, \ldots, f_i)$ that obeys the strict partial order, i.e., $\mathcal{C}(\mathcal{F}^*)$ is downward closed and for every $x, y \in \mathcal{C}(\mathcal{F}^*)$ $(x \neq y)$, $x \prec y \implies x <_{\mathcal{F}^*}$

y. Let $\mathcal{F}_{i-1} = (f_1, \ldots, f_{i-1})$, which is a valid flip sequence at s by the induction hypothesis. Let $\mathcal{X}$ be the set of all positive flip sequences valid at s whose last element is f_i (since $f_i \in \mathcal{P}$ we have $\mathcal{X} \neq \emptyset$). Consider the set $\mathcal{C} = \bigcap_{\mathcal{F} \in \mathcal{X}} \mathcal{C}(\mathcal{F})$ (note that $f_i \in \mathcal{C}$). We first show that since $\mathcal{F}^*$ obeys the strict partial order, we must have $\mathcal{C} \subseteq \mathcal{C}(\mathcal{F}^*)$. Consider an arbitrary element $x^+ \in \mathcal{C}$. If $x^+ = f_i$, then $x^+ \in \mathcal{C}(\mathcal{F}^*)$. If $x^+ \neq f_i$, then the definition of $\mathcal{C}$ implies that x^+ appears before f_i in all valid sequences starting at s and ending in f_i. But then $x^+ \prec f_i$. Since $\mathcal{C}(\mathcal{F}^*)$ is a downward closed set containing f_i, we infer that $x^+ \in \mathcal{C}(\mathcal{F}^*)$. Consequently, we have shown that if $x^+ \in \mathcal{C}$ then $x^+ \in \mathcal{C}(\mathcal{F}^*)$, that is, $\mathcal{C} \subseteq \mathcal{C}(\mathcal{F}^*)$. Now using Lemma 4.8, we conclude that $\mathcal{C}$ is a valid flip set. Since $\mathcal{C}(\mathcal{F}_{i-1})$ is also a valid flip set (from the induction hypothesis), from Lemma 4.3 we know that $\mathcal{C} \cup \mathcal{C}(\mathcal{F}_{i-1}) = \mathcal{C}(\mathcal{F}_{i-1}) \cup \{f_i\} = \mathcal{C}(\mathcal{F}^*)$ (since $\{f_i\} \subseteq \mathcal{C} \subseteq \mathcal{C}(\mathcal{F}^*)$) is a valid flip set. Finally, since $\mathcal{C}(\mathcal{F}^*)$ is a valid positive flip set, $\mathcal{F}_{i-1}$ is a valid positive flip sequence, and $\mathcal{C}(\mathcal{F}^*) \setminus \mathcal{C}(\mathcal{F}_{i-1}) = \{f_i\}$, $\mathcal{F}^*$ must be a valid positive flip sequence by Claim 1. □

4.5. The polynomial-time algorithm. We are now ready to present the algorithm for finding shortest reconfiguration paths in CNF($\mathcal{S}$) formulas where $\mathcal{S}$ is navigable. If every relation in $\mathcal{S}$ is componentwise bijunctive, we use Gopalan et al.'s algorithm. Otherwise, we assume that every relation in $\mathcal{S}$ is NAND-free and dual-Horn-free.

Let ϕ be a CNF($\mathcal{S}$) formula where every relation in $\mathcal{S}$ is NAND-free and dual-Horn-free, $\{x_1, \ldots, x_n\}$ be the set of variables, and $\{C_1, \ldots, C_m\}$ be the set of clauses in ϕ. We wish to compute the shortest reconfiguration path between s and t in G_ϕ for $s, t \in R_\phi$. Let $\mathcal{P}_s$ and $\mathcal{P}_t$ be the sets of positive flips that occur in any positive flip set valid at s and t, respectively.

In Lemma 4.10, we generalize Lemma 4.2 from a single relation to R_ϕ and show that if there exists a valid sequence that transforms s to t, then it can be made canonical. Similarly, Lemma 4.11 shows that the property of any valid flip sequence for a NAND-free and dual-Horn-free relation being describable by a strict partial order, as proved in Lemma 4.9, also applies to CNF($\mathcal{S}$) formulas where every relation in $\mathcal{S}$ is NAND-free and dual-Horn-free.

LEMMA 4.10. *Let ϕ be a CNF($\mathcal{S}$) formula where every relation in $\mathcal{S}$ is NAND-free. For any $s, t \in R_\phi$, if $\mathcal{F}$ is a valid sequence that transforms s to t, then there exists a valid canonical sequence $\mathcal{F}'$ equivalent to $\mathcal{F}$ such that $\mathcal{C}(\mathcal{F}') \subseteq \mathcal{C}(\mathcal{F})$.*

Proof. If $\mathcal{F}$ cannot be made canonical, then it must have a negative flip followed by a positive flip such that the two flips act on two different variables, say x_1 and x_2, and cannot be swapped. In other words, setting either both x_1 and x_2 to 0 or at most one of the two to 1 results in a satisfying assignment, whereas setting $x_1 = x_2 = 1$ results in an assignment which does not satisfy ϕ. Hence, there exists a clause in ϕ that is satisfied by $\{01, 10, 00\}$ but is violated by 11. So the relation corresponding to that clause is not NAND-free, a contradiction. □

LEMMA 4.11. *Let ϕ be a CNF($\mathcal{S}$) formula where every relation in $\mathcal{S}$ is NAND-free and dual-Horn-free. For any $s \in R_\phi$, there exists a strict partial order $\prec_s$ on $\mathcal{P}_s$ such that any positive flip sequence $\mathcal{F}_s$ consisting of a subset of $\mathcal{P}_s$ is a valid flip sequence at s if and only if it obeys the strict partial order $\prec_s$. Moreover, $\mathcal{P}_s$ and $\prec_s$ can be computed in polynomial time.*

Proof. We compute $\mathcal{P}_s$ and $\prec_s$ using a directed graph G_s which we generate in two phases; a construction phase and a deletion phase.

We define $\mathcal{P} = \{x^+ \mid x^+ \in \mathcal{C}$ for a positive valid flip set $\mathcal{C}$ at s for some relation in $\mathcal{S}\}$ and let G_s contain a node for each flip in $\mathcal{P}$. The assignment s in-

duces an assignment $f_{X_j}(s)$ on clause $C_j = (R_j, X_j)$ and Lemma 4.9 defines a strict partial order $\prec_s^j$ that characterizes the valid positive sequences in R_j starting at $f_{X_j}(s)$. For all $p, q \in \{1, \ldots, k_j\}$ (where k_j is the arity of R_j) such that $p^+ \prec_s^j q^+$, if $X_j(p) \notin \{c_0, c_1\}, X_j(q) \notin \{c_0, c_1\}$, and $X_j(p) \neq X_j(q)$, we add the directed edge $(x^+_{X_j(p)}, x^+_{X_j(q)})$ to G_s. We do this for each clause C_j for $j \in \{1, \ldots, m\}$. This completes the construction phase of G_s.

Now, in this graph, the flip corresponding to a vertex f which lies on a cycle and the flip corresponding to any vertex reachable from f by an outgoing directed path (starting from f) are never going to be performed (as these flips do not satisfy the order relation on the edges). Hence we delete the corresponding vertices from G_s as follows. First, any vertex that appears on a directed cycle is marked to be deleted. Then, we iteratively mark every vertex that has an incoming edge from a marked vertex. Once the set of marked vertices stops changing, we delete all marked vertices. Note that G_s is now acyclic and we can detect if a vertex belongs to a cycle in polynomial time using standard depth-first search starting at that vertex.

We claim that $\mathcal{P}_s = V(G_s)$. This follows from Lemma 4.9 and the fact that any vertex that was removed from G_s cannot be a part of any valid flip sequence at s. We define a strict partial order $\prec_s$ as follows: $f_1 \prec_s f_2$ if and only if there is a directed path from f_1 to f_2 in G_s. To see that $\prec_s$ is the required strict partial order, it is enough to note that any flip sequence is valid for ϕ if and only if it is valid for each clause.

Computing the strict partial orders defined by Lemma 4.9 can be accomplished in constant time for each relation in $\mathcal{S}$ (recall that $\mathcal{S}$ is assumed to be finite). Then, the construction and deletion phases for G_s can be accomplished in polynomial time as described above. □

For a set $\mathcal{P}$, a strict partial order $\prec$ on $\mathcal{P}$, and a subset $A \subseteq \mathcal{P}$, the *smallest lower set* of A is the smallest superset of A that is downward closed. Such a lower set can be constructed in polynomial time by starting with A and including any element f' not in A such that $f' \prec f$ for some $f \in A$. It is clear that any valid flip set that contains A must also contain the smallest lower set of A.

Now the algorithm for finding the shortest reconfiguration path is clear. We start from s and let S be the set of positive flips on the variables that are set to 1 in t and to 0 in s. Then we compute the smallest lower set S' containing S and perform the flips in S' as prescribed by the strict partial order $\prec_s$ (on $\mathcal{P}_s$) to reach $s' \in R_\phi$. We perform a similar set of flips starting from t to reach $t' \in R_\phi$. If $s' = t'$, we are done. Otherwise, we recursively find the shortest path between s' and t'. The complete algorithm is described in Algorithm 1.

We are now ready to prove the following theorem.

THEOREM 4.12. *Let $\mathcal{S}$ be a navigable set of relations, ϕ be a CNF($\mathcal{S}$) formula, and s and t be two satisfying assignments of ϕ. We can compute the shortest reconfiguration path between s and t in polynomial time.*

Proof. We show that Algorithm 1 finds the shortest path between s and t, and runs in polynomial time. For any Boolean vector x, let $\eta(x)$ denote the number of 0's in x and let $\eta = \eta(s) + \eta(t)$. It is clear that Steps 1 to 15 take time polynomial in the input size N, where $N = |\phi| + |\mathcal{S}| + |s| + |t|$ and $|x|$ denotes the number of bits needed to represent x. Since $\mathcal{F}_s$ and $\mathcal{F}_t$ are both positive flip sequences, $\eta(s') + \eta(t') \leq \eta(s) + \eta(t) - 2$. Thus the running time $T(\eta)$ of the algorithm satisfies the recursive inequality $T(\eta) \leq T(\eta - 2) + P(N)$ where $P(N)$ is some polynomial in N. Since $\eta < N$, this results in a polynomial in N.

Algorithm 1 ShortestPath(s,t)

Require: A CNF($\mathcal{S}$) formula ϕ where all relations in $\mathcal{S}$ are NAND-free and dual-Horn-free; two satisfying assignments s and t
Ensure: Shortest reconfiguration path between s and t
1: **if** $(s = t)$
2: **return** $\mathcal{F}_0$ {the empty flip sequence}
3: **end if**
4: Let S be the set of positive flips that flip variables assigned 0 in s and 1 in t
5: Let T be the set of positive flips that flip variables assigned 0 in t and 1 in s
6: Compute $\mathcal{P}_s$ and $\prec_s$
7: Compute $\mathcal{P}_t$ and $\prec_t$
8: **if** S contains an element not in $\mathcal{P}_s$ or if T contains an element not in $\mathcal{P}_t$
9: **return** not connected
10: **end if**
11: Compute the smallest lower set S' of S in $\mathcal{P}_s$ with respect to $\prec_s$
12: Compute the smallest lower set T' of T in $\mathcal{P}_t$ with respect to $\prec_t$
13: Let $\mathcal{F}_s$ be an ordering of S' that obeys $\prec_s$
14: Let $\mathcal{F}_t$ be an ordering of T' that obeys $\prec_t$
15: Let $s' = \mathcal{F}_s(s)$ and $t' = \mathcal{F}_t(t)$
16: Let $\mathcal{F} =$ ShortestPath(s',t')
17: **return** $\mathcal{F}_s \cdot \mathcal{F} \cdot \mathcal{F}_t^{-1}$

Finally, we prove the correctness of the algorithm. We use induction on η. If $\eta = 0$, then $s = t$ and the algorithm is trivially correct. If the algorithm returns "not connected", then it is because of either Step 9 or Step 16. If it is because of Step 16, then by the induction hypothesis s' and t' are not connected, and thus s and t are also not connected. Any flip sequence that transforms s to t must perform each flip in S. Thus it is also clear that if Step 9 returns "not connected", then s and t are not connected. If the algorithm returns a flip sequence, then we claim that it is a shortest sequence. From induction, we know that $\mathcal{F}$ is a shortest flip sequence from s' to t'. The claim follows from the observation that if s and t are connected, then there must exist a shortest path from s to t that passes through both s' and t'. We know from Lemma 4.10 that any flip sequence from s to t can be made canonical. Hence, we let $\mathcal{F}_1 \cdot \mathcal{F}_2^{-1}$ be a shortest flip sequence from s to t such that $\mathcal{F}_1$ and $\mathcal{F}_2$ are both positive. It is clear that $S' \subseteq \mathcal{C}(\mathcal{F}_1)$. Since S' is the smallest lower set of S in $\mathcal{P}_s$ with respect to $\prec_s$, from Lemma 4.11, there must exist a valid ordering of $\mathcal{C}(\mathcal{F}_1)$ that first performs all flips of S' (by the construction of the smallest lower set there exists no $f' \in S'$ and $f \in \mathcal{C}(\mathcal{F}_1) \setminus S'$ such that $f \prec_s f'$). In this ordering, the vertex reached after performing all flips of S' is exactly s'. Using a similar argument on $\mathcal{F}_2$, we get a shortest path that goes through both s' and t'. □

5. Final remarks. Many computational problems can be modeled as finding shortest paths in large graphs. Our result provides new insights into the kinds of structures a graph will need to possess to be amenable to an efficient shortest path algorithm. The fact that the shortest path in navigable formulas flips variables that are not in the symmetric difference is evidence that our algorithm exploits a property of the reconfiguration graph that is fundamentally new. Any previously known properties that were used to find shortest paths efficiently also rendered the graph too simple, in that any shortest path only flipped the symmetric difference. It will be

interesting to see if our results help us understand other large graphs, in particular, the flip graph of triangulations of a convex polygon where the complexity of finding the shortest path is still open.

REFERENCES

[1] Oswin Aichholzer, Wolfgang Mulzer, and Alexander Pilz, *Flip distance between triangulations of a simple polygon is NP-complete*, in Proceedings of the 21st Annual European Symposium on Algorithms, vol. 8125, 2013, pp. 13–24.

[2] Sanjeev Arora and Boaz Barak, *Computational Complexity: A Modern Approach*, Cambridge University Press, 1st ed., 2009.

[3] Garrett Birkhoff, *Rings of sets*, Duke Mathematical Journal, 3 (1937), pp. 443–454.

[4] Marthe Bonamy and Nicolas Bousquet, *Recoloring bounded treewidth graphs*, Electronic Notes in Discrete Mathematics, 44 (2013), pp. 257–262.

[5] Paul Bonsma, Amer E. Mouawad, Naomi Nishimura, and Venkatesh Raman, *The complexity of bounded length graph recoloring and CSP reconfiguration*, in Proceedings of the 9th International Symposium on Parameterized and Exact Computation, 2014, pp. 110–121.

[6] Prosenjit Bose, Anna Lubiw, Vinayak Pathak, and Sander Verdonschot, *Flipping edge-labelled triangulations*, CoRR, (2013). arXiv:1310.1166.

[7] Luis Cereceda, Jan van den Heuvel, and Matthew Johnson, *Connectedness of the graph of vertex-colourings*, Discrete Mathematics, 308 (2008), pp. 913–919.

[8] Nadia Creignou, Sanjeev Khanna, and Madhu Sudan, *Complexity classifications of Boolean constraint satisfaction problems*, Society for Industrial and Applied Mathematics, 2001.

[9] Karel Culik II and Derick Wood, *A note on some tree similarity measures*, Information Processing Letters, 15 (1982), pp. 39–42.

[10] David Eppstein, Jean-Claude Falmagne, and Sergei Ovchinnikov, *Media Theory - Interdisciplinary Applied Mathematics*, Springer, 2008.

[11] Gerd Fricke, Sandra Mitchell Hedetniemi, Stephen T. Hedetniemi, and Kevin R. Hutson, *γ-Graphs of Graphs*, Discussiones Mathematicae Graph Theory, 31 (2011), pp. 517–531.

[12] Cyril Gavoille, David Peleg, Stéphane Pérennes, and Ran Raz, *Distance labeling in graphs*, in Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms, 2001, pp. 210–219.

[13] Parikshit Gopalan, Phokion G. Kolaitis, Elitza N. Maneva, and Christos H. Papadimitriou, *The connectivity of Boolean satisfiability: computational and structural dichotomies*, SIAM Journal on Computing, 38 (2009), pp. 2330–2355.

[14] Takehiro Ito, Erik D. Demaine, Nicholas J. A. Harvey, Christos H. Papadimitriou, Martha Sideri, Ryuhei Uehara, and Yushi Uno, *On the complexity of reconfiguration problems*, Theoretical Computer Science, 412 (2011), pp. 1054–1065.

[15] Takehiro Ito, Marcin Kamiński, and Erik D. Demaine, *Reconfiguration of list edge-colorings in a graph*, Discrete Applied Mathematics, 160 (2012), pp. 2199–2207.

[16] Takehiro Ito, Marcin Kamiński, and Hirotaka Ono, *Fixed-parameter tractability of token jumping on planar graphs*, in Algorithms and Computation, Lecture Notes in Computer Science, Springer International Publishing, 2014, pp. 208–219.

[17] Takehiro Ito, Marcin Kamiński, Hirotaka Ono, Akira Suzuki, Ryuhei Uehara, and Katsuhisa Yamanaka, *On the parameterized complexity for token jumping on graphs*, in Proceedings of the 11th Annual Conference on Theory and Applications of Models of Computation, vol. 8402 of Lecture Notes in Computer Science, Springer International Publishing, 2014, pp. 341–351.

[18] Matthew Johnson, Dieter Kratsch, Stefan Kratsch, Viresh Patel, and Daniel Paulusma, *Finding shortest paths between graph colourings*, in Proceedings of the 9th International Symposium on Parameterized and Exact Computation, 2014, pp. 221–233.

[19] Marcin Kamiński, Paul Medvedev, and Martin Milanič, *Complexity of independent set reconfigurability problems*, Theoretical Computer Science, 439 (2012), pp. 9–15.

[20] Charles L. Lawson, *Transforming triangulations*, Discrete Mathematics, 3 (1972), pp. 365–372.

[21] Daniel Lokshtanov, Amer E. Mouawad, Fahad Panolan, M.S. Ramanujan, and Saket Saurabh, *Reconfiguration on sparse graphs*, in Proceedings of the 14th International Symposium on Algorithms and Data Structures Symposium, 2015, pp. 506–517.

[22] Amer E. Mouawad, Naomi Nishimura, Vinayak Pathak, and Venkatesh Raman, *Shortest*

reconfiguration paths in the solution space of Boolean formulas, in Proceedings of the 42nd International Colloquium on Automata, Languages, and Programming, vol. 9134, 2015, pp. 985–996.

[23] Amer E. Mouawad, Naomi Nishimura, and Venkatesh Raman, *Vertex cover reconfiguration and beyond*, in Proceedings of the 25th International Symposium on Algorithms and Computation, 2014, pp. 452–463.

[24] Amer E. Mouawad, Naomi Nishimura, Venkatesh Raman, Narges Simjour, and Akira Suzuki, *On the parameterized complexity of reconfiguration problems*, in Proceedings of the 8th International Symposium on Parameterized and Exact Computation, 2013, pp. 281–294.

[25] Amer E. Mouawad, Naomi Nishimura, Venkatesh Raman, and Marcin Wrochna, *Reconfiguration over tree decompositions*, in Proceedings of the 9th International Symposium on Parameterized and Exact Computation, 2014, pp. 246–257.

[26] David Peleg, *Proximity-preserving labeling schemes*, Journal of Graph Theory, 33 (2000), pp. 167–176.

[27] Thomas J. Schaefer, *The complexity of satisfiability problems*, in Proceedings of the 10th Annual ACM Symposium on Theory of Computing, 1978, pp. 216–226.

[28] Konrad W. Schwerdtfeger, *A computational trichotomy for connectivity of Boolean satisfiability*, CoRR, (2013). arXiv:1312.4524.

[29] Jan van den Heuvel, *The complexity of change*, Surveys in Combinatorics 2013, 409 (2013), pp. 127–160.

[30] Marcin Wrochna, *Reconfiguration in bounded bandwidth and treedepth*, CoRR, (2014). arXiv:1405.0847.