Fixed-Parameter Algorithms for Graph Constraint Logic

Tatsuhiko Hatanaka
Graduate School of Information Sciences, Tohoku University, Sendai, Japan
hatanaka@ecei.tohoku.ac.jp

Felix Hommelsheim
Fakultät für Mathematik, TU Dortmund University, Germany
felix.hommelsheim@math.tu-dortmund.de

Takehiro Ito
Graduate School of Information Sciences, Tohoku University, Sendai, Japan
takehiro@tohoku.ac.jp

Yusuke Kobayashi
Research Institute for Mathematical Sciences, Kyoto University, Japan
yusuke@kurims.kyoto-u.ac.jp

Moritz Mühlenthaler
Laboratoire G-SCOP, Grenoble INP, Université Grenoble Alpes, France
moritz.muhlenthaler@grenoble-inp.fr

Akira Suzuki
Graduate School of Information Sciences, Tohoku University, Sendai, Japan
a.suzuki@ecei.tohoku.ac.jp

Abstract

Non-deterministic constraint logic (NCL) is a simple model of computation based on orientations of a constraint graph with edge weights and vertex demands. NCL captures PSPACE and has been a useful tool for proving algorithmic hardness of many puzzles, games, and reconfiguration problems. In particular, its usefulness stems from the fact that it remains PSPACE-complete even under severe restrictions of the weights (e.g., only edge-weights one and two are needed) and the structure of the constraint graph (e.g., planar AND/OR graphs of bounded bandwidth). While such restrictions on the structure of constraint graphs do not seem to limit the expressiveness of NCL, the building blocks of the constraint graphs cannot be limited without losing expressiveness: We consider as parameters the number of weight-one edges and the number of weight-two edges of a constraint graph, as well as the number of AND or OR vertices of an AND/OR constraint graph. We show that NCL is fixed-parameter tractable (FPT) for any of these parameters. In particular, for NCL parameterized by the number of weight-one edges or the number of AND vertices, we obtain a linear kernel. It follows that, in a sense, NCL as introduced by Hearn and Demaine is defined in the most economical way for the purpose of capturing PSPACE.

2012 ACM Subject Classification Theory of computation → Fixed parameter tractability; Mathematics of computing → Graph algorithms

Keywords and phrases Combinatorial Reconfiguration, Nondeterministic Constraint Logic, Fixed Parameter Tractability

Digital Object Identifier 10.4230/LIPIcs.IPEC.2020.15

Related Version A full version of this paper is available at `https://arxiv.org/abs/2011.10385`.

Funding *Tatsuhiko Hatanaka*: Partially supported by JSPS KAKENHI Grant Number JP16J02175, Japan.
Takehiro Ito: Partially supported by JSPS KAKENHI Grant Numbers JP18H04091 and JP19K11814, Japan.

15th International Symposium on Parameterized and Exact Computation (IPEC 2020).
Editors: Yixin Cao and Marcin Pilipczuk; Article No. 15; pp. 15:1–15:15

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Yusuke Kobayashi: Partially supported by JSPS KAKENHI Grant Numbers 17K19960, 18H05291, and JP20K11692, Japan.
Akira Suzuki: Partially supported by JSPS KAKENHI Grant Numbers JP18H04091 and JP20K11666, Japan.

1 Introduction

Non-deterministic constraint logic (NCL) has been introduced by Hearn and Demaine [7] as a model of computation in order to show that many puzzles and games are complete in their natural complexity classes. For instance, they showed that the 1-player games Sokoban and Rush Hour are PSPACE-complete [7] and there are many follow-up results showing hardness of a large number of puzzles, games, and reconfiguration problems. An NCL constraint graph is a graph with edge-weights one and two and a *configuration* is given by an orientation of the constraint graph, such that the in-weight at each vertex is at least two. Two configurations are *adjacent* if they differ with respect to the orientation of a single edge. The question whether two given configurations are connected by a path, i.e., a sequence of adjacent configurations, is known to be PSPACE-complete, even if the constraint graph is a planar graph of maximum degree three (in fact, a planar AND/OR graph, to be defined shortly) [7]. Similar hardness results are known for the question whether it is possible to reverse a single given edge, or whether there is a transformation between two configurations, such that each edge is reversed at most once.

One of the main advantages of NCL, apart from its simplicity, is its hardness on constraint graphs with a severely restricted structure, which entails strong hardness results for other problems. In particular, NCL is PSPACE-complete on *AND/OR graphs*, which are cubic graphs, where each vertex is either incident to three weight-two edges ("OR vertex") or exactly one weight-two edge ("AND vertex"), see Figure 1. It remains PSPACE-complete if in addition we assume that the constraint graphs are planar [7] and have bounded bandwidth [15]. We investigate the possibility of obtaining a further strengthening by restricting the *composition* of the constraint graph. In particular we consider constraint graphs with a bounded number of weight-one or weight-two edges, and AND/OR graphs with a bounded number of AND or OR vertices. Our main result is that NCL parameterized by any of the four quantities admits an FPT algorithm. That is, for the purpose of capturing PSPACE, the definition of NCL given by Hearn and Demaine is as economical as possible. We furthermore hope that based on our results, NCL may become of interest for investigating the parameterized complexity of puzzles, games, and reconfiguration problems.

In the following we adhere to the historical convention that an edge of weight one (resp., weight two) of a constraint graph is called *red* (resp., *blue*). We refer to the question whether a given configuration of a constraint graph is reachable from another given configuration as

(a) AND vertex. **(b)** OR vertex.

Figure 1 The two types of vertices that occur in AND/OR constraint graphs. Edges must be oriented such that the in-weight at each vertex is at least two. By convention, weight-one edges are red and weight-two edges are blue.

Table 1 Parameterized Complexity of NCL. For entries marked with † we obtain a linear kernel.

Parameter(s)	C2C	C2E
treewidth and max. degree [15]	PSPACE-c	PSPACE-c
transformation length [15]	W[1]-hard	W[1]-hard
transformation length and max. degree [15]	FPT	FPT
# of AND vertices (AND/OR graphs)	FPT† (Cor. 4)	FPT† (Cor. 7)
# of OR vertices (AND/OR graphs)	FPT (Thm. 1)	FPT (Thm. 1)
# of red edges	FPT† (Thm. 3)	FPT† (Cor. 7)
# of blue edges	FPT (Thm. 8)	FPT (Cor. 18)

configuration-to-configuration (*C2C*). Furthermore, by *configuration-to-edge* (C2E) we refer to the question whether, we can reach from a given configuration another one such that a given edge is reversed.

Our Contribution

We consider four natural parameterizations of NCL and show that the corresponding parameterized problems admit FPT algorithms. In particular we consider as parameters

1. the number of AND vertices of an AND/OR graph,
2. the number of OR vertices of an AND/OR graph,
3. the number of red edges of a constraint graph, and
4. the number of blue edges of a constraint graph.

Note that none of these parameterizations trivially leads to an XP algorithm that just enumerates all orientations for the constant number of red/blue edges according to the parameter. For an overview of the parameterized complexity results on NCL, including our results, please refer to Table 1.

NCL is known to be PSPACE-complete on AND/OR constraint graphs, which are undirected edge-weighted graphs where each vertex is either and AND vertex or an OR vertex as shown in Figure 1. We show that C2C and C2E parameterized by the number of AND vertices or the number of OR vertices admits an FPT algorithm. The algorithm first performs a preprocessing step followed by a reduction to the problem Binary Constraint Satisfiability Reconfiguration (BCSR for short). Hatanaka et al. have shown that BCSR can be solved in time $O^*(d^{O(p)})$, where d and p are the maximum size of a domain and the number of non-Boolean variables, respectively [5].

On general constraint graphs we obtain a linear kernel for C2C parameterized by the number of red edges. For this purpose we introduce three reduction rules, which, when applied exhaustively, yield a kernel of linear size. To the best of our knowledge, this is the first polynomial kernel for a parameterization of NCL. The first rule states that each component containing at least two blue cycles can be replaced by a gadget of constant size for each red edge that is attached to the component. The second rule states that vertices incident to a blue edge only can be deleted, since the orientation of this edge is the same for every orientation. The third rule is inverse to subdividing a blue edge: any vertex incident to precisely two blue edges can be deleted and replaced by a single blue edge connecting its former neighbors. Note that the number of red edges in an AND/OR graph is precisely the number of AND vertices in an AND/OR graph. Hence, a linear kernel for NCL parameterized by the number of red edges implies a linear kernel for NCL parameterized by the number of AND vertices of an AND/OR graph. Furthermore, we show that slightly modified reduction rules can be applied in order to obtain a linear kernel for C2E.

Finally, we consider C2C and C2E parameterized by the number k of blue edges and show that it admits an FPT algorithm. Our key idea is to partition the set of feasible orientations of the constraint graph into $2^{O(k)}$ classes, such that in each class, all blue edges are oriented in the same way and the red edges have the same indegree sequence. Denote the set of these classes by $\mathcal{F}$, and define a mapping ϕ from the set of orientations of the constraint graph to $\mathcal{F}$ (see Section 5.1 for the details). Then, in Section 5.2, we define an adjacency relation between elements in $\mathcal{F}$, which is consistent with the reachability of configurations of the constraint graph in some sense. In our algorithm, instead of the original reconfiguration problem, we first solve the reconfiguration problem in $\mathcal{F}$, which can be done in $2^{O(k)} \cdot \mathrm{poly}(|V|)$ time, where V is the set of vertices of the constraint graph. If it is impossible to reach the target configuration in $\mathcal{F}$, then we can conclude that it is also impossible with respect to the original constraint graph. Otherwise, we can reduce the original problem to the case that the blue edges agree in the initial and target configuration and the set of red edges in the initial configurations whose orientation differs from the target configuration consists of arc-disjoint dicycles (see Section 5.3). Finally, in Section 5.4, we test whether the direction of each dicycle can be reversed or not.

Related Work

A large number of puzzles, games, and reconfiguration problems have been shown to be hard using reductions from NCL and its variants. Examples include motion planning problems, where rectangular pieces have to be moved to certain final positions and sliding block puzzles such as Rush Hour [3, 7], Sokoban [7], Snowman [6] and other puzzle games such as Bloxors [16]. In the *bounded length* version of NCL, the orientation of each edge may be reversed at most once. This variant has been used to show NP-completeness of the games Klondike, Mahjong Solitaire and Nonogram [8]. Note that NCL gives a uniform view on games as computation and often allows for simpler proofs and strengthenings of known complexity results in this area. Furthermore, deciding proof equivalence in multiplicative linear logic has been shown to be PSPACE-complete by a reduction from NCL [9].

NCL is also very useful for showing hardness of reconfiguration problems. In a reconfiguration problem we are given two configurations and agree on some simple "move" that produces a new configuration from a given one. The question is whether we can reach the second configuration from the first by a sequence of moves. For surveys on reconfiguration problems, please refer to [12, 14]. For many reconfiguration problems, such as token sliding on graphs [7], a variant of independent set reconfiguration [11], as well as vertex cover reconfiguration [10], dominating set reconfiguration [4], reconfiguration of paths [2], and deciding Kempe-equivalence of 3-colorings [1], reductions from NCL establish PSPACE-hardness even on planar graphs of low maximum degree. Van der Zanden showed that there is some constant c, such that NCL is PSPACE-complete on planar subcubic graphs of bandwidth at most c [15]. Note that this property is often maintained in the reductions [1, 2, 4, 7, 10] and it implies that NCL remains hard on graphs of bounded treewidth.

Tractable special cases of NCL have received much less attention. Concerning parameterized complexity, NCL remains PSPACE-complete when parameterized by treewidth and maximum degree of the constraint graph. On the other hand, NCL parameterized by the length of the transformation is W[1]-hard and it becomes FPT when parameterized by the length of the transformation and the maximum degree [15]. If additionally each edge may be reversed at most once in a transformation, NCL is FPT when parameterized by treewidth and the maximum degree, or by the length of the transformation [15].

Organization

The paper is organized as follows. In the next section we give some preliminaries about NCL and introduce notation used throughout the paper. Section 3 contains our FPT algorithm for NCL parameterized by the number of OR vertices. The linear kernel for NCL parameterized by the number of red edges, which also implies the result for AND vertices, can be found in Section 4. Finally, in Section 5 we give an FPT algorithm for NCL parameterized by the number of blue edges. Section 6 concludes the paper and gives some open problems.

2 Preliminaries

Let $G = (V, E)$ be an undirected graph, which may have multiple edges and (self) loops. We denote by $V(G)$ (resp., $E(G)$) the set of vertices (resp., set of edges) G. Each edge in an undirected graph which joins two vertices x and y is represented as an unordered pair xy (or equivalently yx). On the other hand, each arc in a digraph which leaves x and enters y is written as an ordered pair (x, y). Let (V, E, w) be a constraint graph, that is, an undirected graph (V, E) with edge weights $w : E \to \{1, 2\}$. We denote by E^{red} and E^{blue} the sets of red (weight one) and blue (weight two) edges in E, respectively, and have that $E = E^{\mathrm{red}} \cup E^{\mathrm{blue}}$. We denote by $V_{\mathrm{AND}}(G)$ and $V_{\mathrm{OR}}(G)$ the sets of AND and OR vertices in a graph G, respectively; we sometimes drop G, and simply write V_{AND} and V_{OR} if it is clear from the context. A constraint graph is called *AND/OR graph* if each vertex is an AND or OR vertex; thus, an AND/OR graph is 3-regular.

An *orientation* A of E is a multi-set of arcs obtained by replacing each edge in E with a single arc having the same end vertices. We refer to G as the underlying graph of the digraph (V, A). For an orientation A of E, we always denote by A^{red} and A^{blue} the subsets of A corresponding to E^{red} and E^{blue}, respectively. For any arc subset $B \subseteq A$ and a vertex $v \in V$, let $\rho_B(v)$ denote the number of arcs in B that enter v. Then, ρ_B can be regarded as a vector in $\mathbb{Z}_{\geq 0}^{V}$, where $\mathbb{Z}_{\geq 0}$ is the set of all nonnegative integers. An orientation A of E is *feasible* if $\rho_{A^{\mathrm{red}}}(v) + 2 \cdot \rho_{A^{\mathrm{blue}}}(v) \geq 2$ for every $v \in V$; a feasible orientation is synonymously referred to as *configuration*.

For two orientations B and B' of an edge subset $F \subseteq E$, we write $B \leftrightarrow B'$ if $B = B'$ or there exists an arc $(x, y) \in B$ such that $B' = (B \setminus \{(x, y)\}) \cup \{(y, x)\}$. For notational convenience, we simply write $B' = B - (x, y) + (y, x)$ in the latter case. For an orientation B of F, *reversing* the direction of an edge $xy \in F$ is the operation which yields from B an orientation B' of F, such that $B' = B - (x, y) + (y, x)$ if $(x, y) \in B$ and $B - (y, x) + (x, y)$ otherwise. For two feasible orientations A and A' of E, a sequence $\langle A_0, A_1, \ldots, A_\ell \rangle$ of feasible orientations of E is called a *reconfiguration sequence* between A and A' if $A_0 = A$, $A_\ell = A'$, and $A_{i-1} \leftrightarrow A_i$ for all $i \in \{1, 2, \ldots, \ell\}$. We write $A \leftrightsquigarrow A'$ if there exists a reconfiguration sequence between A and A' (or $A \not\leftrightsquigarrow A'$ if not). Given a constraint graph G and two feasible orientations A_{ini} and A_{tar} of $E(G)$, the problem C2C asks whether $A_{\mathrm{ini}} \leftrightsquigarrow A_{\mathrm{tar}}$ or not. Similarly, given a constraint graph (G, w), a feasible orientation A_{ini} of $E(G)$, and an edge $e \in E(G)$, the problem C2E asks whether there is a feasible orientation A_{tar}, such that $A_{\mathrm{ini}} \leftrightsquigarrow A_{\mathrm{tar}}$ and the direction of e is different in A_{ini} and A_{tar}. We denote by a triple $(G, A_{\mathrm{ini}}, A_{\mathrm{tar}})$ an instance of C2C and by a triple $(G, A_{\mathrm{ini}}, vw)$ an instance of C2E.

3 NCL for AND/OR graphs

In this section, we consider NCL when restricted to AND/OR constraint graphs. Recall that NCL remains PSPACE-complete on AND/OR graphs [7]. We thus prove that C2C and C2E on AND/OR constraint graphs is fixed-parameter tractable when parameterized by the

number of OR vertices. An analogous result for C2C and C2E parameterized by the number of AND vertices follows from our FPT result for NCL parameterized by the number of red edges in the next section (see Theorem 3). Therefore, the main result here is the following theorem.

▶ **Theorem 1.** *C2C and C2E on AND/OR constraint graphs with n vertices parameterized by the number k of OR vertices admits a $2^{O(k)} \cdot \mathrm{poly}(n)$-time algorithm.*

In the reminder of this section, we give an overview of the proof of Theorem 1. Our strategy is to give an FPT-reduction from C2C on AND/OR constraint graphs to the BINARY CONSTRAINT SATISFIABILITY RECONFIGURATION problem (BCSR, for short) [5], which will be defined in Section 3.2. To do so, we first apply some preprocessing to a given instance of C2C on an AND/OR graph (in Section 3.1), and then give our FPT-reduction to BCSR (in Section 3.2). By similar arguments we obtain the result for C2E.

3.1 Preprocessing

The preprocessing subdivides each blue edge that is not a loop into two blue edges. It is not hard to see that a single subdivision yields an equivalent instance: Let uv be a blue edge of a constraint graph $\hat{G}$ and consider the constraint graph G obtained by subdividing uv into two blue edges uz and zv, where z is a new vertex we call *middle vertex*. Let G be the resulting constraint graph and observe that from any feasible orientation $\hat{A}$ of $\hat{G}$ we may obtain a feasible orientation A of G by letting $A = \hat{A} - (u,v) + (u,z) + (z,v)$ if $(u,v) \in \hat{A}$ and $A = \hat{A} - (v,u) + (v,z) + (z,u)$ otherwise.

Furthermore, in any feasible orientation of $\hat{G}$, we can transfer in-weight from, say, u to v by reversing the arc (v,u) iff the in-weight at u is at least four. Furthermore, due to the orientation of uv, the corresponding arc contributes to the in-weight of precisely one of u and v. Conversely, in an orientation of G, we can transfer in-weight from, say, u to v by reversing the directions of the arcs corresponding to uz and zv iff the in-weight at u is at least four. Furthermore, in any orientation of G, the arcs corresponding to uz and zv contribute in-weight to at most one of u and v. Hence, by subdividing a blue edge of $\hat{G}$ from an instance $(\hat{G}, \hat{A}_{\mathrm{ini}}, \hat{A}_{\mathrm{tar}})$ of C2C, we obtain an equivalent instance. Let $(G, A_{\mathrm{ini}}, A_{\mathrm{tar}})$ be the instance of C2C obtained from $(\hat{G}, \hat{A}_{\mathrm{ini}}, \hat{A}_{\mathrm{tar}})$ by subdividing each blue edge of $\hat{G}$ that is not a loop. By repetition of the above argument we obtain the following result.

▶ **Lemma 2.** $(G, A_{\mathrm{ini}}, A_{\mathrm{tar}})$ *is a* yes*-instance if and only if* $(\hat{G}, \hat{A}_{\mathrm{ini}}, \hat{A}_{\mathrm{tar}})$ *is.*

3.2 FPT-reduction to BCSR

In this subsection, we sketch our FPT-reduction to BCSR. We start by formally defining the problem BCSR. Let $H = (X, F)$ be an undirected graph. We call each vertex $x \in X$ a *variable*. Each $x \in X$ has a finite set $D(x)$, called a *domain of* x. A variable x is called a *Boolean variable* if $|D(x)| \leq 2$, and otherwise called a *non-Boolean variable*. Each edge $xy \in F$ has a subset $\mathcal{C}(xy) \subseteq D(x) \times D(y)$, called a (*binary*) *constraint of* xy. A mapping $\Gamma\colon X \to \bigcup_{x \in X} D(x)$ is a *solution* of H if $\Gamma(x) \in D(x)$ for every $x \in X$. In addition, a solution Γ of H is *proper* if $\Gamma(x)\Gamma(y) \in \mathcal{C}(xy)$ for every $xy \in F$. For two solutions Γ and Γ', we write $\Gamma \leftrightarrow \Gamma'$ if $|\{x \in X : \Gamma(x) \neq \Gamma'(x)\}| = 1$. Given an undirected graph H, a domain $D(x)$ for each $x \in X$, a constraint $\mathcal{C}(xy)$ for each $xy \in F$, and two proper solutions Γ_{ini} and Γ_{tar} of H, the BINARY CONSTRAINT SATISFIABILITY RECONFIGURATION problem (BCSR) asks whether there exists a sequence $\langle \Gamma_0, \Gamma_1, \ldots, \Gamma_\ell \rangle$ of proper solutions of H such that $\Gamma_0 = \Gamma_{\mathrm{ini}}$, $\Gamma_\ell = \Gamma_{\mathrm{tar}}$, and $\Gamma_{i-1} \leftrightarrow \Gamma_i$ for each $i \in \{1, 2, \ldots, \ell\}$. Let $(H, D, \mathcal{C}, \Gamma_{\mathrm{ini}}, \Gamma_{\mathrm{tar}})$ an instance of BCSR.

It is known that BCSR can be solved in time $O^*(d^{O(p)})$, where $d := \max_{x \in X} |D(x)|$ and p is the number of non-Boolean variables in X [5, Theorem 18]. To prove Theorem 1, given an instance $(\hat{G}, \hat{A}_{\text{ini}}, \hat{A}_{\text{tar}})$ of C2C on an AND/OR constraint graph with at most k AND/OR vertices, we first perform the preprocessing from Section 3.1 to obtain an instance $(G, A_{\text{ini}}, A_{\text{tar}})$ of C2C. Note that G is not an AND/OR graph, and V can be partitioned into $V_{\text{AND}}(G)$, $V_{\text{OR}}(G)$ and $V_{\text{MID}}(G)$, where $V_{\text{MID}}(G)$ (or simply V_{MID}) is the set of middle vertices in G. By Lemma 2, we have that $(G, A_{\text{ini}}, A_{\text{tar}})$ is a yes-instance if and only if $(\hat{G}, \hat{A}_{\text{ini}}, \hat{A}_{\text{tar}})$. Hence, to conclude the proof of Theorem 1, we provide an FPT-reduction from a preprocessed instance $(G, A_{\text{ini}}, A_{\text{tar}})$ of C2C with the parameter $|V_{\text{OR}}(G)| = |V_{\text{OR}}(\hat{G})| \leq k$ to an instance $(H, D, \mathcal{C}, \Gamma_{\text{ini}}, \Gamma_{\text{tar}})$ of BCSR such that both d and p are bounded by some computable functions depending only on k.

Due to the preprocessing, observe that the constraint graph G has no two parallel blue edges. In addition, no edge in G joins an AND vertex and an OR vertex, and hence we can partition E into two sets E_{AND} and E_{OR}, defined as follows: E_{AND} is the set of edges of G that are incident to an AND vertex; E_{OR} is the set of edges of G that are incident to an OR vertex. The high-level idea of the reduction to BCSR is the following. For each OR vertex v, we create an OR variable x_v. Observe that the in-weight requirement at v is violated only if each arc is pointing away from v. We forbid such orientations by giving each OR variable x_v a domain of size seven corresponding to the seven legal orientations of the incident edges of v.

The remaining in-weight requirements and consistency requirements are modelled by adding constraints, which also define the set of edges in H. For each edge e of G, we create a Boolean edge-variable x_e, whose domain represents the two possible orientations of an edge. The construction of domains above ensures that in-weight requirement is satisfied for each AND vertex. To ensure the same property for all other vertices, we add three types of constraints for middle vertices and AND vertices, to enforce the following constraints:

Type 1: Constraints for middle vertices.

Let v be a middle vertex between two vertices v_1 and v_2. Since both v_1v and vv_2 are blue edges, the in-weight requirement at v is satisfied if and only if v_1v or vv_2 points to v.

Type 2-1: Constraints for AND vertices having loops.

Let v be an AND vertex having a loop vv. So vv must be red and the remaining edge $vv_3 \in E_{\text{AND}}$ is blue where v_3 is a middle vertex. Then, the in-weight requirement at v is satisfied if and only if vv_3 is oriented towards v.

Type 2-2: Constraints for AND vertices without loops.

Let v be an AND vertex, and let vv_1, vv_2, vv_3 be three (distinct) edges incident to v such that vv_1 and vv_2 are red, and vv_3 is blue; it may hold that $v_1 = v_2$. Then, the in-degree requirement at v is satisfied if and only if i) vv_1 or vv_3 are oriented towards v and ii) vv_2 or vv_3 are oriented towards v.

By the construction of constraints above, we know that a solution Γ of H is proper if and only if the corresponding orientation A_Γ of E is feasible. Therefore, we can define proper solutions Γ_{ini} and Γ_{tar} of H which correspond to feasible orientations A_{ini} and A_{tar} of E, respectively. In this way, from a preprocessed instance $(G, A_{\text{ini}}, A_{\text{tar}})$ of C2C with the parameter $|V_{\text{OR}}(G)| \leq k$, we have constructed in polynomial time a corresponding equivalent instance $(H, D, \mathcal{C}, \Gamma_{\text{ini}}, \Gamma_{\text{tar}})$ of BCSR such that $d = \max_{x \in X} |D(x)| = 7$ and $p \leq |V_{\text{OR}}(G)| \leq k$.

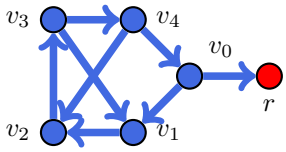

Figure 2 The gadget used in reduction rule 2.

4 NCL parameterized by the number of red edges

Our main result in this section is a linear kernel for C2C parameterized by the number of red edges of the constraint graph.

▶ **Theorem 3.** *There is a polynomial-time algorithm that, given an instance of C2C on a constraint graph with k red edges, outputs an equivalent instance of C2C of size $O(k)$.*

In particular, Theorem 3 implies that C2C parameterized by the number of red edges admits a $O^*(2^{O(k)})$-time algorithm. By observing that in any AND/OR constraint graph, the number of red edges is equal to the number of AND vertices, we immediately obtain the following result.

▶ **Corollary 4.** *C2C on AND/OR graphs parameterized by the number k' of AND vertices admits a kernel of size $O(k')$.*

It can be shown by similar arguments that there is also a linear kernel for C2E parameterized by the number of red edges of the constraint graph. In the remainder of this section, we prove Theorem 3. Let $I = (G, A_{\text{ini}}, A_{\text{tar}})$ be an instance of C2C, where G is any constraint graph with k red edges. We give four reduction rules, and show that applying them repeatedly preserves the answer. Furthermore, we show that applying them exhaustively yields an instance of size $O(k)$, where $k = |E^{\text{red}}|$. To conclude the proof, we show that the reduction can be applied in polynomial time.

We say that a vertex is *blue* if all its incident edges are blue. Otherwise, if at least one incident edge is red, we call the vertex *red*. A subset $V' \subseteq V$ is called a *blue component* if it is a connected component in the graph (V, E^{blue}). Note that a blue component may contain red vertices of G. The first reduction rule removes blue components of G that are directed cycles. Observe that no arc in such a component can be reversed. The second reduction rule removes blue components that contain at least two cycles and attaches to each red vertex v of the component a copy of the gadget shown in Figure 2. The gadget consists of a cycle on five new vertices $\{v_0, v_1, v_2, v_3, v_4\}$ with two chords $\{v_1, v_3\}$ and $\{v_2, v_4\}$. Additionally we add an edge joining v and v_0. All edges of the gadget have weight two. The third reduction rule removes blue vertices of degree one and the last rule removes the center vertex of a blue path on three vertices.

While modifying G we also modify A_{ini} and A_{tar} accordingly. That is, if we delete edges of G, these edges are also deleted in A_{ini} and A_{tar}. If we add a gadget to G, then the arcs in A_{ini} and A_{tar} have the same orientation on the gadget. Note that the number k of red vertices is not altered by an application of any of the rules. Here is a more formal description of the four rules:

▶ **Reduction rule 1.** *Let C be a component of G that is a blue chordless cycle. If the orientations A_{ini} and A_{tar} agree on C, then we remove C from the graph and adjust A_{ini} and A_{tar} accordingly. Otherwise we output a* no*-instance.*

▶ **Reduction rule 2.** *Let C be a blue component that contains at least two cycles. Then we remove from G every blue vertex in C and attach to each red vertex in C a copy of the gadget in Figure 2. Additionally we modify A_{ini} and A_{tar} accordingly such that both agree on each copy of the gadget.*

▶ **Reduction rule 3.** *If G has a blue vertex v of degree one, delete v and its incident edge from G and remove the corresponding arc(s) from A_{ini} and A_{tar}.*

▶ **Reduction rule 4.** *Suppose G has a blue vertex v of degree 2, such that the two neighbors u and w of v are non-adjacent in G. Then delete v and its incident edges from G and add the blue edge uw. Remove any arcs incident to v from A_{ini} and A_{tar}. Finally, add (u, w) to A_{ini} (resp. A_{tar}) if $(u, v) \in A_{\mathrm{ini}}$ (resp., A_{tar}) and (w, u) otherwise.*

We show that applying any of the four rules is *safe*, that is any application results in a yes-instance if and only if I is a yes-instance.

▶ **Proposition 5.** *Reduction rules 1–4 are safe for C2C.*

By applying a depth-first-search we can check if any of the rules can be applied. Thus we have the following.

▶ **Proposition 6.** *Reduction rules 1-4 can be applied exhaustively in time $O(|V| \cdot (|V| + |E|))$.*

Theorem 3 now follows by the previous propositions and a simple counting argument. Using similar arguments we show that there is also a linear kernel for C2E parameterized by the number k of red edges. The main difference is that in reduction rule 2 we only add the gadget to each red vertex that is part of a cycle or connected to two distinct cycles by two disjoint paths. Furthermore, if the edge e that we wish to reverse is part of of a component containing two cycles, we add a gadget to the tail of e.

▶ **Corollary 7.** *C2E parameterized by the number k of red edges admits a kernel of size $O(k)$. Furthermore, C2E on AND/OR graphs parameterized by the number k' of AND vertices admits a kernel of size $O(k')$.*

5 NCL parameterized by the number of blue edges

The objective of this section is to show that C2C parameterized by the number k of blue edges is fixed parameter tractable.

▶ **Theorem 8.** *C2C parameterized by the number k of blue edges can be solved in time $2^{O(k)} \cdot \mathrm{poly}(|V|)$.*

In the remainder of this section, we prove Theorem 8. Let $I = (G, A_{\mathrm{ini}}, A_{\mathrm{tar}})$ be an instance of C2C, where G is any constraint graph and denote by V and E the set of vertices and edges of G, respectively.

Let $\mathcal{A}$ denote the set of all feasible orientations of E. Our key idea is to classify the feasible orientations into $2^{O(k)}$ classes, where each class is determined by the orientation A^{blue} of E^{blue} and the indegree sequence of A^{red}. Denote the set of these classes by $\mathcal{F}$, and define a mapping ϕ from $\mathcal{A}$ to $\mathcal{F}$ (see Section 5.1 for details). Then, in Section 5.2, we define a reconfiguration relation $\overset{}{\underset{\mathcal{F}}{\leftrightsquigarrow}}$ between elements in $\mathcal{F}$, which is consistent with $\leftrightsquigarrow$ in some sense. In our algorithm, instead of the original reconfiguration problem in $\mathcal{A}$, we first solve the reconfiguration problem in $\mathcal{F}$, which can be done in $2^{O(k)} \cdot \mathrm{poly}(|V|)$ time. If it has no reconfiguration sequence, then we can conclude that there is no reconfiguration sequence

in the original problem. Otherwise, we can reduce the original problem to the case when $A^{\mathrm{blue}}_{\mathrm{ini}} = A^{\mathrm{blue}}_{\mathrm{tar}}$ and $A^{\mathrm{red}}_{\mathrm{ini}} \setminus A^{\mathrm{red}}_{\mathrm{tar}}$ consists of arc-disjoint dicycles (see Section 5.3). Finally, in Section 5.4, we test whether the direction of each dicycle can be reversed or not.

Before starting the main part of the proof of Theorem 8, we show the following lemma that plays an important role in our argument. Roughly, it says that we can change the orientation of E^{red} keeping a certain indegree constraint.

▶ **Lemma 9.** *Let $A^{\mathrm{red}}_{\mathrm{ini}}$ and $A^{\mathrm{red}}_{\mathrm{tar}}$ be orientations of E^{red}. Then, there exists a sequence $A^{\mathrm{red}}_0, A^{\mathrm{red}}_1, \ldots, A^{\mathrm{red}}_l$ of orientations of E^{red} such that $A^{\mathrm{red}}_0 = A^{\mathrm{red}}_{\mathrm{ini}}$, $\rho_{A^{\mathrm{red}}_l} = \rho_{A^{\mathrm{red}}_{\mathrm{tar}}}$, $A^{\mathrm{red}}_{i-1} \leftrightarrow A^{\mathrm{red}}_i$ for $i = 1, \ldots, l$, and $\rho_{A^{\mathrm{red}}_i}(v) \geq \min\{\rho_{A^{\mathrm{red}}_{\mathrm{ini}}}(v), \rho_{A^{\mathrm{red}}_{\mathrm{tar}}}(v)\}$ for any $v \in V$ and any $i \in \{0, 1, \ldots, l\}$.*

Proof. We prove the lemma by induction on $|A^{\mathrm{red}}_{\mathrm{ini}} \setminus A^{\mathrm{red}}_{\mathrm{tar}}|$. If $\rho_{A^{\mathrm{red}}_{\mathrm{ini}}} = \rho_{A^{\mathrm{red}}_{\mathrm{tar}}}$, then the claim is obvious, because the sequence consisting of only one orientation $A^{\mathrm{red}}_0 = A^{\mathrm{red}}_{\mathrm{ini}}$ satisfies the conditions. Thus, it suffices to consider the case when $\rho_{A^{\mathrm{red}}_{\mathrm{ini}}} \neq \rho_{A^{\mathrm{red}}_{\mathrm{tar}}}$. In this case, there exists a vertex $u \in V$ such that $\rho_{A^{\mathrm{red}}_{\mathrm{ini}}}(u) > \rho_{A^{\mathrm{red}}_{\mathrm{tar}}}(u)$, because $\sum_{v \in V} \rho_{A^{\mathrm{red}}_{\mathrm{ini}}}(v) = \sum_{v \in V} \rho_{A^{\mathrm{red}}_{\mathrm{tar}}}(v)$. Then, there exists an arc $a \in A^{\mathrm{red}}_{\mathrm{ini}} \setminus A^{\mathrm{red}}_{\mathrm{tar}}$ that enters u. Let A^{red}_1 be the orientation of E^{red} obtained from $A^{\mathrm{red}}_{\mathrm{ini}}$ by reversing the direction of a. Since $|A^{\mathrm{red}}_1 \setminus A^{\mathrm{red}}_{\mathrm{tar}}| < |A^{\mathrm{red}}_{\mathrm{ini}} \setminus A^{\mathrm{red}}_{\mathrm{tar}}|$, by induction hypothesis, there exists a sequence $A^{\mathrm{red}}_1, \ldots, A^{\mathrm{red}}_l$ of orientations of E^{red} such that $\rho_{A^{\mathrm{red}}_l} = \rho_{A^{\mathrm{red}}_{\mathrm{tar}}}$, $A^{\mathrm{red}}_{i-1} \leftrightarrow A^{\mathrm{red}}_i$ for $i = 2, \ldots, l$, and $\rho_{A^{\mathrm{red}}_i}(v) \geq \min\{\rho_{A^{\mathrm{red}}_1}(v), \rho_{A^{\mathrm{red}}_{\mathrm{tar}}}(v)\}$ for any $v \in V$ and any $i \in \{1, \ldots, l\}$. By letting $A^{\mathrm{red}}_0 = A^{\mathrm{red}}_{\mathrm{ini}}$, the sequence $A^{\mathrm{red}}_0, A^{\mathrm{red}}_1, \ldots, A^{\mathrm{red}}_l$ satisfies the conditions, because $A^{\mathrm{red}}_0 \leftrightarrow A^{\mathrm{red}}_1$, $\rho_{A^{\mathrm{red}}_1}(v) \geq \rho_{A^{\mathrm{red}}_{\mathrm{ini}}}(v)$ for each $v \in V \setminus \{u\}$, and $\min\{\rho_{A^{\mathrm{red}}_1}(u), \rho_{A^{\mathrm{red}}_{\mathrm{tar}}}(u)\} = \rho_{A^{\mathrm{red}}_{\mathrm{tar}}}(u) = \min\{\rho_{A^{\mathrm{red}}_{\mathrm{ini}}}(u), \rho_{A^{\mathrm{red}}_{\mathrm{tar}}}(u)\}$. ◀

The proof of Lemma 9 is constructive, and hence we can find such a sequence efficiently.

5.1 Classification of $\mathcal{A}$

In this subsection, we classify the feasible orientations into $2^{O(k)}$ classes. Let $X \subseteq V$ be the set of all vertices to which edges in E^{blue} are incident. Define $\mathcal{F}$ as the set of all pairs (A^{blue}, d) where A^{blue} is an orientation of E^{blue} and d is a vector in $\{0, 1, 2\}^X$ satisfying the following conditions:

(1) $2\rho_{A^{\mathrm{blue}}}(v) + d(v) \geq 2$ for any $v \in X$.

(2) There exists an orientation A^{red} of E^{red} such that for any $v \in V$,

$$\rho_{A^{\mathrm{red}}}(v) \begin{cases} = 0 & \text{if } v \in X \text{ and } d(v) = 0, \\ = 1 & \text{if } v \in X \text{ and } d(v) = 1, \text{ and} \\ \geq 2 & \text{otherwise.} \end{cases}$$

We note that $|\mathcal{F}| \leq 2^{|E^{\mathrm{blue}}|} \cdot 3^{|X|} = 2^{O(k)}$, because $|X| \leq 2|E^{\mathrm{blue}}|$. For a vector $d \in \{0, 1, 2\}^X$, we say that an orientation A^{red} of E^{red} *realizes* d if A^{red} satisfies the condition (2) above. We can easily see that if $(A^{\mathrm{blue}}, d) \in \mathcal{F}$ holds and A^{red} realizes d, then $A := A^{\mathrm{blue}} \cup A^{\mathrm{red}}$ is a feasible orientation of E. Conversely, if $A = A^{\mathrm{blue}} \cup A^{\mathrm{red}}$ is a feasible orientation of E (i.e., $A \in \mathcal{A}$), then the vector $d \in \{0, 1, 2\}^X$ defined by $d(v) = \min\{\rho_{A^{\mathrm{red}}}(v), 2\}$ for each $v \in X$ satisfies that $(A^{\mathrm{blue}}, d) \in \mathcal{F}$. This defines a mapping ϕ from $\mathcal{A}$ to $\mathcal{F}$.

We can also see that the membership problem of $\mathcal{F}$ can be decided in polynomial time.

▶ **Lemma 10.** *For an orientation A^{blue} of E^{blue} and a vector $d \in \{0, 1, 2\}^X$, we can test whether $(A^{\mathrm{blue}}, d) \in \mathcal{F}$ or not in polynomial time.*

Proof. We can easily check the condition (1). To check the condition (2), we construct a digraph $\hat{G} = (\hat{V}, \hat{A})$ and consider a network flow problem in it. Introduce a new vertex w_e for each $e \in E^{\mathrm{red}}$ and two new vertices s and t, and define $\hat{V} := V \cup \{w_e \mid e \in E^{\mathrm{red}}\} \cup \{s, t\}$. Define the arc set $\hat{A} := \hat{A}_1 \cup \hat{A}_2 \cup \hat{A}_3$ by

$$\hat{A}_1 := \{(s, w_e) \mid e \in E^{\mathrm{red}}\},$$
$$\hat{A}_2 := \{(w_e, v) \mid e \in E^{\mathrm{red}}, v \in V, e \text{ is incident to } v \text{ in } G\},$$
$$\hat{A}_3 := \{(v, t) \mid v \in V\}.$$

For each $a \in \hat{A}$, define the lower bound $l(a)$ and the upper bound $u(a)$ of the amount of flow through a as follows.

- For each $(s, w_e) \in \hat{A}_1$, define $l(s, w_e) := u(s, w_e) := 1$.
- For each $(w_e, v) \in \hat{A}_2$, define $l(w_e, v) := 0$ and $u(w_e, v) := 1$.
- For each $(v, t) \in \hat{A}_3$, define $l(v, t) := u(v, t) := d(v)$ if $v \in X$ and $d(v) \in \{0, 1\}$, and define $l(v, t) := 2$ and $u(v, t) := +\infty$ otherwise.

Then, the condition (2) holds if and only if $\hat{G}$ has an integral s-t flow satisfying the above constraint. This can be tested in polynomial time by a standard maximum flow algorithm (see e.g. [13, Corollary 11.3a]). ◀

5.2 Reconfiguration in $\mathcal{F}$

In this subsection, we consider a reconfiguration between elements in $\mathcal{F}$. For $(A_1^{\mathrm{blue}}, d_1)$, $(A_2^{\mathrm{blue}}, d_2) \in \mathcal{F}$, we denote $(A_1^{\mathrm{blue}}, d_1) \underset{\mathcal{F}}{\longleftrightarrow} (A_2^{\mathrm{blue}}, d_2)$ if

- $d_1 = d_2$ and $A_1^{\mathrm{blue}} \leftrightarrow A_2^{\mathrm{blue}}$, or
- $A_1^{\mathrm{blue}} = A_2^{\mathrm{blue}}$.

If there exists a sequence $(A_0^{\mathrm{blue}}, d_0), (A_1^{\mathrm{blue}}, d_1), \ldots, (A_l^{\mathrm{blue}}, d_l) \in \mathcal{F}$ such that $(A_{i-1}^{\mathrm{blue}}, d_{i-1}) \underset{\mathcal{F}}{\longleftrightarrow} (A_i^{\mathrm{blue}}, d_i)$ for $i = 1, \ldots, l$, then we denote $(A_0^{\mathrm{blue}}, d_0) \underset{\mathcal{F}}{\leftrightsquigarrow} (A_l^{\mathrm{blue}}, d_l)$. Then, we can easily see the following.

▶ **Lemma 11.** *Let $A_{\mathrm{ini}}, A_{\mathrm{tar}} \in \mathcal{A}$. If $A_{\mathrm{ini}} \leftrightsquigarrow A_{\mathrm{tar}}$, then $\phi(A_{\mathrm{ini}}) \underset{\mathcal{F}}{\leftrightsquigarrow} \phi(A_{\mathrm{tar}})$.*

Proof. If $A_{\mathrm{ini}} \leftrightarrow A_{\mathrm{tar}}$, then $\phi(A_{\mathrm{ini}}) \underset{\mathcal{F}}{\longleftrightarrow} \phi(A_{\mathrm{tar}})$ by definition. By using this relationship repeatedly, we obtain the claim. ◀

Although the opposite implication is not true, we show the following statement.

▶ **Lemma 12.** *Let $A_{\mathrm{ini}}, A_{\mathrm{tar}} \in \mathcal{A}$. If $\phi(A_{\mathrm{ini}}) \underset{\mathcal{F}}{\leftrightsquigarrow} \phi(A_{\mathrm{tar}})$, then there exists $A^{\circ}_{\mathrm{tar}} \in \mathcal{A}$ such that $\phi(A^{\circ}_{\mathrm{tar}}) = \phi(A_{\mathrm{tar}})$ and $A_{\mathrm{ini}} \leftrightsquigarrow A^{\circ}_{\mathrm{tar}}$.*

Proof. It suffices to consider the case when $\phi(A_{\mathrm{ini}}) \underset{\mathcal{F}}{\longleftrightarrow} \phi(A_{\mathrm{tar}})$. Denote $\phi(A_{\mathrm{ini}}) = (A_{\mathrm{ini}}^{\mathrm{blue}}, d_{\mathrm{ini}})$ and $\phi(A_{\mathrm{tar}}) = (A_{\mathrm{tar}}^{\mathrm{blue}}, d_{\mathrm{tar}})$. By definition, we have either $d_{\mathrm{ini}} = d_{\mathrm{tar}}$ and $A_{\mathrm{ini}}^{\mathrm{blue}} \leftrightarrow A_{\mathrm{tar}}^{\mathrm{blue}}$, or $A_{\mathrm{ini}}^{\mathrm{blue}} = A_{\mathrm{tar}}^{\mathrm{blue}}$.

If $d_{\mathrm{ini}} = d_{\mathrm{tar}}$ and $A_{\mathrm{ini}}^{\mathrm{blue}} \leftrightarrow A_{\mathrm{tar}}^{\mathrm{blue}}$, then $A_{\mathrm{ini}} \leftrightarrow A_{\mathrm{tar}}^{\mathrm{blue}} \cup A_{\mathrm{ini}}^{\mathrm{red}}$ and $\phi(A_{\mathrm{tar}}^{\mathrm{blue}} \cup A_{\mathrm{ini}}^{\mathrm{red}}) = (A_{\mathrm{tar}}^{\mathrm{blue}}, d_{\mathrm{ini}}) = (A_{\mathrm{tar}}^{\mathrm{blue}}, d_{\mathrm{tar}}) = \phi(A_{\mathrm{tar}})$, which means that $A^{\circ}_{\mathrm{tar}} := A_{\mathrm{tar}}^{\mathrm{blue}} \cup A_{\mathrm{ini}}^{\mathrm{red}}$ satisfies the conditions.

Otherwise, let $A^{\mathrm{blue}} := A_{\mathrm{ini}}^{\mathrm{blue}} = A_{\mathrm{tar}}^{\mathrm{blue}}$. By Lemma 9, we obtain a sequence $A_0^{\mathrm{red}}, A_1^{\mathrm{red}}, \ldots, A_l^{\mathrm{red}}$ of orientations of E^{red} such that $A_0^{\mathrm{red}} = A_{\mathrm{ini}}^{\mathrm{red}}$, $\rho_{A_l^{\mathrm{red}}} = \rho_{A_{\mathrm{tar}}^{\mathrm{red}}}$, $A_{i-1}^{\mathrm{red}} \leftrightarrow A_i^{\mathrm{red}}$ for $i = 1, \ldots, l$, and $\rho_{A_i^{\mathrm{red}}}(v) \geq \min\{\rho_{A_{\mathrm{ini}}^{\mathrm{red}}}(v), \rho_{A_{\mathrm{tar}}^{\mathrm{red}}}(v)\}$ for any $v \in V$ and any $i \in \{0, 1, \ldots, l\}$. Then, for any $i \in \{0, 1, \ldots, l\}$, we have

$$2\rho_{A^{\mathrm{blue}}}(v) + \rho_{A_i^{\mathrm{red}}}(v) \geq \min\{2\rho_{A^{\mathrm{blue}}}(v) + \rho_{A_{\mathrm{ini}}^{\mathrm{red}}}(v), 2\rho_{A^{\mathrm{blue}}}(v) + \rho_{A_{\mathrm{tar}}^{\mathrm{red}}}(v)\} \geq 2$$

for any $v \in V$, and hence $A^{\mathrm{blue}} \cup A_i^{\mathrm{red}}$ is feasible. Since $A^{\mathrm{blue}} \cup A_{i-1}^{\mathrm{red}} \leftrightarrow A^{\mathrm{blue}} \cup A_i^{\mathrm{red}}$ for $i = 1, \ldots, l$, we have

$$(A_{\mathrm{ini}} =) A^{\mathrm{blue}} \cup A_{\mathrm{ini}}^{\mathrm{red}} \leftrightsquigarrow A^{\mathrm{blue}} \cup A_l^{\mathrm{red}}.$$

Furthermore, since $\rho_{A_l^{\mathrm{red}}} = \rho_{A_{\mathrm{tar}}^{\mathrm{red}}}$, we have $\phi(A^{\mathrm{blue}} \cup A_l^{\mathrm{red}}) = \phi(A_{\mathrm{tar}})$. Therefore, $A_{\mathrm{tar}}^{\circ} := A^{\mathrm{blue}} \cup A_l^{\mathrm{red}}$ satisfies the conditions in the lemma. ◀

Note that we can construct A_{tar}° and a reconfiguration sequence in Lemma 12 efficiently.

5.3 Algorithm

Let $I = (G, A_{\mathrm{ini}}, A_{\mathrm{tar}})$ be an instance of C2C. We first compute $\phi(A_{\mathrm{ini}})$ and $\phi(A_{\mathrm{tar}})$, and test whether $\phi(A_{\mathrm{ini}}) \underset{\mathcal{F}}{\leftrightsquigarrow} \phi(A_{\mathrm{tar}})$ or not. If $\phi(A_{\mathrm{ini}}) \not\underset{\mathcal{F}}{\leftrightsquigarrow} \phi(A_{\mathrm{tar}})$, then we can immediately conclude that $A_{\mathrm{ini}} \not\leftrightsquigarrow A_{\mathrm{tar}}$ by Lemma 11.

Thus, in what follows, suppose that $\phi(A_{\mathrm{ini}}) \underset{\mathcal{F}}{\leftrightsquigarrow} \phi(A_{\mathrm{tar}})$. In this case, by applying Lemma 12, we can construct $A_{\mathrm{tar}}^{\circ} \in \mathcal{A}$ with $\phi(A_{\mathrm{tar}}^{\circ}) = \phi(A_{\mathrm{tar}})$ such that $A_{\mathrm{ini}} \leftrightsquigarrow A_{\mathrm{tar}}^{\circ}$. This shows that $A_{\mathrm{ini}} \leftrightsquigarrow A_{\mathrm{tar}}$ is equivalent to $A_{\mathrm{tar}}^{\circ} \leftrightsquigarrow A_{\mathrm{tar}}$, which means that we can regard A_{tar}° as a new initial configuration instead of A_{ini}. Thus, the problem is reduced to the case when $\phi(A_{\mathrm{ini}}) = \phi(A_{\mathrm{tar}})$. In particular, we have $A_{\mathrm{ini}}^{\mathrm{blue}} = A_{\mathrm{tar}}^{\mathrm{blue}}$.

Suppose that $A_{\mathrm{ini}}^{\mathrm{blue}} = A_{\mathrm{tar}}^{\mathrm{blue}} =: A^{\mathrm{blue}}$ and $\rho_{A_{\mathrm{ini}}^{\mathrm{red}}} \neq \rho_{A_{\mathrm{tar}}^{\mathrm{red}}}$. Then, by applying Lemma 9, we obtain an orientation A_l^{red} of E^{red} such that $\rho_{A_l^{\mathrm{red}}} = \rho_{A_{\mathrm{tar}}^{\mathrm{red}}}$ and $A_{\mathrm{ini}} \leftrightsquigarrow A^{\mathrm{blue}} \cup A_l^{\mathrm{red}}$. This shows that $A_{\mathrm{ini}} \leftrightsquigarrow A_{\mathrm{tar}}$ is equivalent to $A^{\mathrm{blue}} \cup A_l^{\mathrm{red}} \leftrightsquigarrow A_{\mathrm{tar}}$, which means that we can regard $A^{\mathrm{blue}} \cup A_l^{\mathrm{red}}$ as a new initial configuration instead of A_{ini}. Thus, the problem is reduced to the case when $\rho_{A_{\mathrm{ini}}^{\mathrm{red}}} = \rho_{A_{\mathrm{tar}}^{\mathrm{red}}}$. If $A_{\mathrm{ini}}^{\mathrm{red}} = A_{\mathrm{tar}}^{\mathrm{red}}$, we conclude that $A_{\mathrm{ini}} \leftrightsquigarrow A_{\mathrm{tar}}$. Otherwise, since $\rho_{A_{\mathrm{ini}}^{\mathrm{red}}} = \rho_{A_{\mathrm{tar}}^{\mathrm{red}}}$, the set $A_{\mathrm{ini}}^{\mathrm{red}} \setminus A_{\mathrm{tar}}^{\mathrm{red}}$ can be decomposed into arc-disjoint cycles.

Note that all of the above procedures can be executed in $2^{O(k)} \cdot \mathrm{poly}(|V|)$ time, since $|\mathcal{F}| = 2^{O(k)}$. In what follows, we give an algorithm for testing whether the direction of each cycle can be reversed or not. For this purpose, we show the following lemma.

▶ **Lemma 13.** *Let $A_{\mathrm{ini}} \in \mathcal{A}$ and let C be a dicycle with all the arcs in $A_{\mathrm{ini}}^{\mathrm{red}}$. Then, the followings are equivalent.*

- **(i)** *$A_{\mathrm{ini}} \leftrightsquigarrow (A_{\mathrm{ini}} \setminus C) \cup \overline{C}$, where $\overline{C}$ is the reverse dicycle of C.*
- **(ii)** *For any arc a in C, there exists an orientation $A \in \mathcal{A}$ such that $A_{\mathrm{ini}} \leftrightsquigarrow A$ and $a \notin A$.*
- **(iii)** *For any $u \in V(C)$, there exists an orientation $A \in \mathcal{A}$ such that $A_{\mathrm{ini}} \leftrightsquigarrow A$ and $2\rho_{A^{\mathrm{blue}}}(u) + \rho_{A^{\mathrm{red}}}(u) \geq 3$.*

Proof. We prove (i)⇒(ii), (ii)⇒(iii), and (iii)⇒(i), respectively.

(i)⇒(ii) If (i) holds, then $A := (A_{\mathrm{ini}} \setminus C) \cup \overline{C}$ satisfies the conditions in (ii), since it contains no arc in C.

(ii)⇒(iii) We prove the contraposition. Assume that (iii) does not hold, that is, there exists a vertex $u \in V(C)$ such that $2\rho_{A^{\mathrm{blue}}}(u) + \rho_{A^{\mathrm{red}}}(u) = 2$ for any $A \in \mathcal{A}$ with $A_{\mathrm{ini}} \leftrightsquigarrow A$. Let a be the arc in C that enters u. Since we cannot reverse the direction of a without violating the feasibility, a is contained in any orientation $A \in \mathcal{A}$ with $A_{\mathrm{ini}} \leftrightsquigarrow A$.

(iii)⇒(i) Suppose that (iii) holds. We take a sequence $A_0, A_1, \ldots, A_l$ of feasible orientations of E such that $A_0 = A_{\mathrm{ini}}$, A_i is obtained from A_{i-1} by reversing an arc $a_i \in A_{i-1}$ for $i \in \{1, 2, \ldots, l\}$, and there exists $u \in V(C)$ such that $2\rho_{A_l^{\mathrm{blue}}}(u) + \rho_{A_l^{\mathrm{red}}}(u) \geq 3$. By taking a minimal sequence with these conditions, we may assume that a_i is not contained in C for $i \in \{1, 2, \ldots, l\}$. Since $2\rho_{A_l^{\mathrm{blue}}}(u) + \rho_{A_l^{\mathrm{red}}}(u) \geq 3$, starting from A_l, we can change the

Algorithm 1 Algorithm for the Reconfiguration Problem.

Input : a graph $G = (V, E)$ and orientations $A_{\mathrm{ini}}, A_{\mathrm{tar}} \in \mathcal{A}$.
Output : "yes" if $A_{\mathrm{ini}} \leftrightsquigarrow A_{\mathrm{tar}}$, and "no" otherwise.

Compute $\mathcal{F}, \phi(A_{\mathrm{ini}})$, and $\phi(A_{\mathrm{tar}})$;
if $\phi(A_{\mathrm{ini}}) \not\underset{\mathcal{F}}{\leftrightsquigarrow} \phi(A_{\mathrm{tar}})$ **then return** "no" ;
if $\phi(A_{\mathrm{ini}}) \neq \phi(A_{\mathrm{tar}})$ *or* $\rho_{A^{\mathrm{red}}_{\mathrm{ini}}} \neq \rho_{A^{\mathrm{red}}_{\mathrm{tar}}}$ **then**
    Compute $A \in \mathcal{A}$ such that $\phi(A) = \phi(A_{\mathrm{tar}})$, $\rho_{A^{\mathrm{red}}} = \rho_{A^{\mathrm{red}}_{\mathrm{tar}}}$, and $A_{\mathrm{ini}} \leftrightsquigarrow A$;
    $A_{\mathrm{ini}} \leftarrow A$; `// See Section 5.2`
while $A^{\mathrm{red}}_{\mathrm{ini}} \setminus A^{\mathrm{red}}_{\mathrm{tar}}$ *contains a dicycle* C **do**
    Take $u \in V(C)$ and solve Problem A;
    if Problem A *has no feasible solution* **then**
        Return "no" ;
    else
        $A_{\mathrm{ini}} \leftarrow (A_{\mathrm{ini}} \setminus C) \cup \overline{C}$; `// See Section 5.3`
return "yes" ;

direction of each arc in C one by one without violating the feasibility, which shows that $A_l \leftrightsquigarrow (A_l \setminus C) \cup \overline{C}$. On the other hand, since $(A_i \setminus C) \cup \overline{C}$ is obtained from $(A_{i-1} \setminus C) \cup \overline{C}$ by reversing a_i for $i \in \{1, 2, \ldots, l\}$, we obtain $(A_{\mathrm{ini}} \setminus C) \cup \overline{C} \leftrightsquigarrow (A_l \setminus C) \cup \overline{C}$. Thus, it holds that $A_{\mathrm{ini}} \leftrightsquigarrow A_l \leftrightsquigarrow (A_l \setminus C) \cup \overline{C} \leftrightsquigarrow (A_{\mathrm{ini}} \setminus C) \cup \overline{C}$. ◀

Let C be a dicycle in $A^{\mathrm{red}}_{\mathrm{ini}} \setminus A^{\mathrm{red}}_{\mathrm{tar}}$. Fix a vertex $u \in V(C)$ and consider the following problem, for which an algorithm is presented later in Section 5.4.

Problem A
Input: A constraint graph G, an orientation $A_{\mathrm{ini}} \in \mathcal{A}$, and a vertex $u \in V(G)$.
Task: Find an orientation $A \in \mathcal{A}$ s.t. $2\rho_{A^{\mathrm{blue}}}(u) + \rho_{A^{\mathrm{red}}}(u) \geq 3$ and $A_{\mathrm{ini}} \leftrightsquigarrow A$ (if exists).

If Problem A has no solution, then condition (iii) in Lemma 13 does not hold. This shows that the condition (ii) in Lemma 13 does not hold, that is, there exists an arc a in C that is contained in any orientation $A \in \mathcal{A}$ with $A_{\mathrm{ini}} \leftrightsquigarrow A$. In this case, since $a \in A_{\mathrm{ini}} \setminus A_{\mathrm{tar}}$, we conclude that $A_{\mathrm{ini}} \not\leftrightsquigarrow A_{\mathrm{tar}}$.

Otherwise, Problem A has a solution, and hence the condition (iii) in Lemma 13 holds. Since it is equivalent to the condition (i) in Lemma 13, we have that $A_{\mathrm{ini}} \leftrightsquigarrow (A_{\mathrm{ini}} \setminus C) \cup \overline{C}$. Therefore, $A_{\mathrm{ini}} \leftrightsquigarrow A_{\mathrm{tar}}$ is equivalent to $(A_{\mathrm{ini}} \setminus C) \cup \overline{C} \leftrightsquigarrow A_{\mathrm{tar}}$, which means that we can regard $(A_{\mathrm{ini}} \setminus C) \cup \overline{C}$ as a new initial configuration instead of A_{ini}. Then, the problem is reduced to the case with smaller $|A_{\mathrm{ini}} \setminus A_{\mathrm{tar}}|$. By applying this procedure at most $O(|E|)$ times repeatedly, we can solve the original reconfiguration problem. The entire algorithm is shown in Algorithm 1.

5.4 Algorithm for Problem A

The remaining task is to give a polynomial time algorithm for Problem A. For this purpose, we use a similar argument to Section 5.2. Suppose we are given a graph $G = (V, E)$ and a vertex $u \in V$. Recall that $X \subseteq V$ is the set of all vertices to which edges in E^{blue} are incident. Define $\mathcal{F}_u$ as the set of all pairs (A^{blue}, d), where A^{blue} is an orientation of E^{blue} and d is a vector in $\{0, 1, 2, 3\}^{X \cup \{u\}}$ satisfying the following conditions:

(1) $2\rho_{A^{\mathrm{blue}}}(v) + d(v) \geq 2$ for any $v \in X \cup \{u\}$.
(2) There exists an orientation A^{red} of E^{red} such that for any $v \in V$,

$$\rho_{A^{\mathrm{red}}}(v) \begin{cases} = d(v) & \text{if } v \in X \cup \{u\} \text{ and } d(v) \in \{0,1,2\}, \\ \geq 3 & \text{if } v \in X \cup \{u\} \text{ and } d(v) = 3, \text{ and} \\ \geq 2 & \text{if } v \in V \setminus (X \cup \{u\}). \end{cases}$$

We note that $|\mathcal{F}_u| \leq 2^{|E^{\mathrm{blue}}|} \cdot 4^{|X \cup \{u\}|} = 2^{O(k)}$. We define $\underset{\mathcal{F}_u}{\longleftrightarrow}$, $\underset{\mathcal{F}_u}{\leftrightsquigarrow}$, and ϕ_u in the same way as $\underset{\mathcal{F}}{\longleftrightarrow}$, $\underset{\mathcal{F}}{\leftrightsquigarrow}$, and ϕ. We obtain the following lemmas in the same way as Lemmas 10, 11, and 12.

▶ **Lemma 14.** *For an orientation A^{blue} of E^{blue} and a vector $d \in \{0,1,2,3\}^X$, we can test whether $(A^{\mathrm{blue}}, d) \in \mathcal{F}_u$ or not in polynomial time.*

▶ **Lemma 15.** *Let $A_{\mathrm{ini}}, A_{\mathrm{tar}} \in \mathcal{A}$. If $A_{\mathrm{ini}} \leftrightsquigarrow A_{\mathrm{tar}}$, then $\phi_u(A_{\mathrm{ini}}) \underset{\mathcal{F}_u}{\leftrightsquigarrow} \phi_u(A_{\mathrm{tar}})$.*

▶ **Lemma 16.** *Let $A_{\mathrm{ini}}, A_{\mathrm{tar}} \in \mathcal{A}$. If $\phi_u(A_{\mathrm{ini}}) \underset{\mathcal{F}_u}{\leftrightsquigarrow} \phi_u(A_{\mathrm{tar}})$, then there exists $A^{\circ}_{\mathrm{tar}} \in \mathcal{A}$ with $\phi_u(A^{\circ}_{\mathrm{tar}}) = \phi_u(A_{\mathrm{tar}})$ such that $A_{\mathrm{ini}} \leftrightsquigarrow A^{\circ}_{\mathrm{tar}}$.*

▶ **Proposition 17.** Problem A *has a solution if and only if there exists a pair $(A^{\mathrm{blue}}, d) \in \mathcal{F}_u$ such that $2\rho_{A^{\mathrm{blue}}}(u) + d(u) \geq 3$ and $\phi_u(A_{\mathrm{ini}}) \underset{\mathcal{F}_u}{\leftrightsquigarrow} (A^{\mathrm{blue}}, d)$.*

Proof. If A is a solution of Problem A, then $\phi_u(A) = (A^{\mathrm{blue}}, d)$ satisfies the conditions by Lemma 15. Conversely, assume that there exists a pair $(A^{\mathrm{blue}}, d) \in \mathcal{F}_u$ such that $2\rho_{A^{\mathrm{blue}}}(u) + d(u) \geq 3$ and $\phi_u(A_{\mathrm{ini}}) \underset{\mathcal{F}_u}{\leftrightsquigarrow} (A^{\mathrm{blue}}, d)$. By Lemma 16, there exists an orientation $A \in \mathcal{A}$ with $\phi_u(A) = (A^{\mathrm{blue}}, d)$ such that $A_{\mathrm{ini}} \leftrightsquigarrow A$. Since $2\rho_{A^{\mathrm{blue}}}(u) + \rho_{A^{\mathrm{red}}}(u) \geq 2\rho_{A^{\mathrm{blue}}}(u) + d(u) \geq 3$, A is a solution of Problem A. ◀

By this proposition, in order to solve Problem A, it suffices to test whether there exists a pair (A^{blue}, d) such that $2\rho_{A^{\mathrm{blue}}}(u) + d(u) \geq 3$ and $\phi_u(A_{\mathrm{ini}}) \underset{\mathcal{F}_u}{\leftrightsquigarrow} (A^{\mathrm{blue}}, d)$. Since $|\mathcal{F}_u| = 2^{O(k)}$, it can be checked in $2^{O(k)} \cdot \mathrm{poly}(|V|)$ time. Note that the elements of $\mathcal{F}_u$ can be computed in $2^{O(k)} \cdot \mathrm{poly}(|V|)$ time by Lemma 14. Thus, Algorithm 1 solves the problem C2C in $2^{O(k)} \cdot \mathrm{poly}(|V|)$ time.

Using similar arguments as in Theorem 8 we can also solve the C2E version.

▶ **Corollary 18.** *C2E parameterized by the number k of blue edges can be solved in time $2^{O(k)} \cdot \mathrm{poly}(|V|)$.*

6 Conclusion

We investigated the parameterized complexity of NCL for four natural parameters related to the constraint graph: The number of AND/OR vertices of an AND/OR graph and the number of red/blue edges of a general constraint graph. We give FPT algorithms for the C2C and C2E version of NCL for each parameter and in particular a linear kernel for NCL parameterized by he number of red edges. An interesting question for future work is whether there is a polynomial kernel for NCL parameterized by the number of OR vertices or the number of blue edges.

References

1. Marthe Bonamy, Marc Heinrich, Takehiro Ito, Yusuke Kobayashi, Haruka Mizuta, Moritz Mühlenthaler, Akira Suzuki, and Kunihiro Wasa. Diameter of colorings under kempe changes. In *Computing and Combinatorics - 25th International Conference, COCOON 2019, Xi'an, China, July 29-31, 2019, Proceedings*, pages 52–64, 2019. doi:10.1007/978-3-030-26176-4_5.
2. Erik D Demaine, David Eppstein, Adam Hesterberg, Kshitij Jain, Anna Lubiw, Ryuhei Uehara, and Yushi Uno. Reconfiguring undirected paths. In *Workshop on Algorithms and Data Structures*, pages 353–365. Springer, 2019.
3. Gary William Flake and Eric B Baum. Rush hour is PSPACE-complete, or "why you should generously tip parking lot attendants". *Theoretical Computer Science*, 270(1-2):895–911, 2002.
4. Arash Haddadan, Takehiro Ito, Amer E Mouawad, Naomi Nishimura, Hirotaka Ono, Akira Suzuki, and Youcef Tebbal. The complexity of dominating set reconfiguration. *Theoretical Computer Science*, 651:37–49, 2016.
5. Tatsuhiko Hatanaka, Takehiro Ito, and Xiao Zhou. Complexity of reconfiguration problems for constraint satisfaction. *CoRR*, abs/1812.10629, 2018. arXiv:1812.10629.
6. Weihua He, Ziwen Liu, and Chao Yang. Snowman is pspace-complete. *Theoretical Computer Science*, 677:31–40, 2017.
7. Robert A. Hearn and Erik D. Demaine. PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. *Theoretical Computer Science*, 343(1-2):72–96, 2005. doi:10.1016/j.tcs.2005.05.008.
8. Hendrik Jan Hoogeboom, Walter A Kosters, Jan N van Rijn, and Jonathan K Vis. Acyclic constraint logic and games. *ICGA Journal*, 37(1):3–16, 2014.
9. Robin Houston and Willem Heijltjes. Proof equivalence in MLL is PSPACE-complete. *Logical Methods in Computer Science*, 12, 2016.
10. 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(12–14):1054–1065, 2011. doi:10.1016/j.tcs.2010.12.005.
11. Marcin Kamiński, Paul Medvedev, and Martin Milanič. Complexity of independent set reconfigurability problems. *Theoretical computer science*, 439:9–15, 2012.
12. Naomi Nishimura. Introduction to reconfiguration. *Algorithms*, 11(4):52, 2018.
13. Alexander Schrijver. *Combinatorial Optimization - Polyhedra and Efficiency*, volume 24 of *Algorithms and Combinatorics*. Springer Berlin Heidelberg, 2003.
14. Jan van den Heuvel. The complexity of change. In Simon R Blackburn, Stefanie Gerke, and Mark Wildon, editors, *Surveys in Combinatorics 2013*, volume 409. London Mathematical Society Lectures Note Series, 2013.
15. Tom C van der Zanden. Parameterized complexity of graph constraint logic. In *10th International Symposium on Parameterized and Exact Computation (IPEC 2015)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2015.
16. Tom C van der Zanden and Hans L Bodlaender. PSPACE-completeness of Bloxorz and of games with 2-buttons. In *International Conference on Algorithms and Complexity*, pages 403–415. Springer, 2015.