Master's Thesis

Parameterized Complexity of Graph Constraint Logic

Author:
Tom C. van der Zanden

Supervisor:
Prof. Dr. Hans L. Bodlaender

ICA-3713784

Department of Information and Computing Sciences
Faculty of Science

Universiteit Utrecht

Abstract

Graph constraint logic is a framework introduced by Hearn and Demaine, which provides several problems that are often a convenient starting point for reductions. We study the parameterized complexity of CONSTRAINT GRAPH SATISFIABILITY and both bounded and unbounded versions of NONDETERMINISTIC CONSTRAINT LOGIC (NCL) with respect to solution length, treewidth and maximum degree of the underlying constraint graph as parameters. As a main result we show that restricted NCL remains *PSPACE*-complete on graphs of bounded bandwidth, strengthening Hearn and Demaine's framework. This allows us to improve upon existing results obtained by reduction from NCL, and we show that reconfiguration versions of several classical graph problems (including independent set, feedback vertex set and dominating set) are *PSPACE*-complete on planar graphs of bounded bandwidth.

Acknowledgements

Working on my Master's Thesis has been tremendous fun and a great experience. I would like to thank my advisor, Hans Bodlaender, not only for his terrific guidance and advice, but also for his encouragement in getting my results published. This has lead to one publication (not part of this thesis) which I presented at CIAC 2015. Part of this thesis is currently under review for publication at another conference. I am very glad that this thesis does not mark the end of our cooperation, as I eagerly look forward to working with Hans as a PhD student, starting in September. I would also like to thank all other staff members at Utrecht University who have made my studies there so enjoyable. Finally I would like to thank my fellow students who also worked on their theses in room BBG 502, with whom I played the occasional game of cards.

Contents

Chapter 1

Introduction

Consider a search or optimization problem where the goal is to find a *single solution* that meets some criteria. We can obtain a *reconfiguration* version by specifying an adjacency relation for solutions (usually a small local change): given a starting solution and a goal solution, we want to find a *sequence of solutions*, every consecutive pair of solutions adjacent, from the starting solution to the goal solution.

Reconfiguration problems have many practical applications, such as planning roadworks without obstructing traffic or moving items around in a storage space to retrieve a given item, though in this thesis we take a more theoretical approach (and also consider applications to the complexity of games and puzzles, which is somewhat recreational).

Many interesting optimization problems are *NP*-complete, meaning it is widely believed that these problems require exponential time to be solved. In this context, the *parameterized complexity* of these problems is studied: we try to identify special cases in which the problems become tractable. For reconfiguration problems the situation is even more dire: when we take an *NP*-complete optimization problem its reconfiguration variant is often *PSPACE*-complete[1]. This means not only that finding a solution is likely very hard, it also means that it is provably hard to execute the solution: the solution might have exponential length in the worst case! This means that identifying tractable special cases does not only make solving the problems easier, it is essential to ensure executing solutions is feasible.

[1]Though problems in P can have *PSPACE*-complete reconfiguration variants as well, for instance Constraint Graph Satisfiability on bounded treewidth 3-regular graphs.

In this thesis, we study the (parameterized) complexity of Constraint Graph Satisfiability and its reconfiguration variant, Nondeterministic Constraint Logic. We consider parametrizations involving the maximum degree, solution length and treewidth of the graph. Constraint Graph Satisfiability is *NP*-complete and Nondeterministic Constraint Logic is *PSPACE*-complete, even on planar, 3-regular graphs. As our main contribution, we show that Nondeterministic Constraint Logic remains *PSPACE*-complete even if we further restrict the instances to have bandwidth bounded by a constant.

1.1 Prior Work

Nondeterministic Constraint Logic (NCL) was introduced by Hearn and Demaine in [1] and extended in [2] to a more general graph constraint logic framework. The framework provides a number of problems complete for various complexity classes, that aim to be a convenient starting point for reductions - particularly for showing the hardness of games and puzzles.

As part of the constraint logic framework, Hearn and Demaine [1] provide a restricted variant of Nondeterministic Constraint Logic (restricted NCL), in which the constraint graph is planar, 3-regular, uses only weights in $\{1, 2\}$ and the graph is constructed from only two specific vertex types (AND and OR). Restricted NCL is *PSPACE*-complete, and (due to the restrictions) is particularly suitable for reductions. Hearn and Demaine's [1] reduction creates a graphs of unbounded treewidth. We strengthen their original result, by providing a new reduction, showing that restricted NCL remains *PSPACE*-complete, even limited to graphs of bandwidth at most a given constant (which is a subclass of graphs of treewidth at most a given constant). We show hardness by reduction from H-Word Reconfiguration, introduced in [3]. In this reduction we also use existing constraint logic gadgets due to Hearn and Demaine [2].

The first study of the complexity of reconfiguration problems in a parameterized setting is due to Mouawad et al. [4], in which the authors recognize two types of parametrization: the solution size k and the length of the reconfiguration sequence l. The use of solution size k as a parameter does not make sense for constraint logic. However, we do consider

l as a parameter. To avoid confusion, throughout this paper we will use $\mathcal{K}$ to refer to the treewidth of a graph (usually a lowercase k is used).

The use of treewidth as a parameter for reconfiguration problems is studied in [3, 5]. By reducing from H-Word Reconfiguration the authors show reconfiguration versions of several graph problems *PSPACE*-complete on graphs of bounded bandwidth. Our proof regarding the hardness of NCL on bounded treewidth graphs also reduces from the H-Word Reconfiguration problem and bears some similarities to the reductions given in these papers.

For several reconfiguration problems, their hardness on planar graphs of low maximum degree is known by reduction from NCL, while their hardness on bounded bandwidth graphs is known by reduction from H-Word Reconfiguration [3, 5, 6, 7]. We unify these results, showing that these problems are hard on graphs that are both planar, have low maximum degree, and are of bounded bandwidth.

1.2 Our Contributions

We study the parameterized complexity of Constraint Graph Satisfiability and Nondeterministic Constraint Logic with respect to combinations of solution length, maximum degree and treewidth as parameters. As our main result, we show that Nondeterministic Constraint Logic remains *PSPACE*-complete even on planar, 3-regular graphs of bandwidth at most a constant (note that bandwidth is a stronger parameter than treewidth).

Our results concerning the hardness of constraint logic problems are summarized in Table 1.1. This table shows the complexity of Constraint Graph Satisfiability (CGS), unbounded configuration-to edge (C2E) and configuration-to-configuration (C2C) variants of Nondeterministic Constraint Logic and their respective bounded counterparts (bC2E and bC2C) with respect to the parameters solution length (l), maximum degree (Δ) and treewidth ($\mathcal{K}$). Where a traditional complexity class is listed this denotes that the problem is hard for this class even when restricted to instances where the parameter is at most a constant.

		Problem				
		CGS	C2E	C2C	bC2E	bC2C
Parameters	-	*NP*-C	*PSPACE*-C	*PSPACE*-C	*NP*-C	*NP*-C
	l	-	$W[1]$-hard	$W[1]$-hard	$W[1]$-hard	*FPT*
	$l + \Delta$	-	*FPT*	*FPT*	*FPT*	*FPT*
	$\mathcal{K}$	weakly *NP*-C	*PSPACE*-C	*PSPACE*-C	weakly *NP*-H	weakly *NP*-H
	$\mathcal{K} + \Delta$	*FPT*	*PSPACE*-C	*PSPACE*-C	*FPT*	*FPT*

Table 1.1: Parameterized complexity of graph constraint logic problems.

The solution length parameter does not apply to CGS, since it does not have such a notion. By reduction from k-Clique, we show that C2E, C2C and bC2E variants of NCL are $W[1]$-hard when parameterized by solution length. On the positive side, the bC2C variant is fixed parameter tractable with respect to solution length. When parametrized with both solution length and maximum degree all variants become fixed parameter tractable. Note that on 3-regular graphs the unbounded variants are *PSPACE*-complete and the bounded variants are *NP*-complete [2], so considering maximum degree as a parameter on its own is not interesting.

We show that CGS is weakly *NP*-complete even on treewidth 2 graphs, but becomes fixed parameter tractable when parametrized by a combination of treewidth and maximum degree. The bounded NCL variants respond similarly: they are weakly *NP*-hard even on treewidth 2 graphs, but become fixed parameter tractable when parameterized by both treewidth and maximum degree.

Our main result is regarding (unbounded) Nondeterministic Constraint Logic (NCL). Unlike CGS, which is fixed parameter tractable with respect to treewidth plus maximum degree, NCL (the reconfiguration version of CGS) remains *PSPACE*-complete even on planar, 3-regular graphs of bounded treewidth constructed using only "AND vertices" and "OR vertices". Our result is in fact stronger, since the graphs created in our reduction have bounded bandwidth - a stronger parameter than treewidth.

As an application of our hardness result for NCL, we show that reconfiguration versions of several classical graph problems are *PSPACE*-complete, even on bounded bandwidth, planar graphs of low maximum degree[2]. Additionally, we show that deciding whether an instance of the Rush Hour puzzle is solvable is *PSPACE*-complete, even when the game is played on a $n \times c$ board where c is a constant.

[2]Note that while having bounded bandwidth implies having bounded degree, the degree bounds we obtain are stronger than the bound obtained from bandwidth.

Chapter 2

Preliminaries

2.1 Graph Constraint Logic

Hearn and Demaine's framework [2] provides several decision problems that are based around the central notion of constraint graphs:

Definition 2.1 (Constraint Graph)**.** A *constraint graph* is a graph with edge weights and vertex weights. A *legal configuration* for a constraint graph is an assignment of an orientation to each edge such that for each vertex, the total weight of the edges pointing in to that vertex is at least that vertex' weight (its *minimum inflow*).

Note that we only count the edges incident to a given vertex towards satisfying its minimum inflow.

A fundamental decision problem regarding constraint graphs is that of their satisfiability:

Constraint Graph Satisfiability (CGS)

Instance: A constraint graph G.

Question: Does G have a legal configuration?

Constraint Graph Satisfiability is *NP*-complete [2]. Hearn and Demaine [2] provide a redution from 3-SAT.

Another important problem regarding constraint graphs is whether they can be reconfigured in to each other:

Nondeterministic Constraint Logic (C2C)

Instance: A constraint graph G and two legal configurations C_1, C_2 for G.

Question: Is there a sequence of legal configurations from C_1 to C_2, where every configuration is obtained from the previous configuration by changing the orientation of one edge?

This problem is called the configuration-to-configuration (C2C) variant of Nondeterministic Constraint Logic . It is *PSPACE*-complete [2]. The configuration-to-edge variant (C2E) is also *PSPACE*-complete [2]:

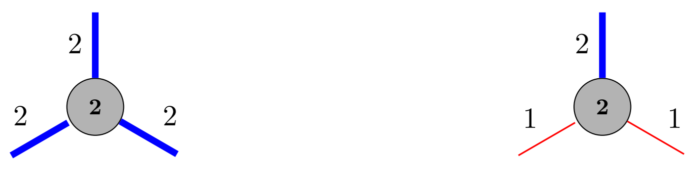
Nondeterministic Constraint Logic (C2E)

Instance: A constraint graph G, a target edge e from G and an initial legal configuration C_1 for G.

Question: Is there a sequence of legal configuration, starting with C_1, where every configuration is obtained from the previous by changing the orientation of one edge, so that eventually e is reversed?

For the C2C and C2E problems, bC2C and bC2E denote their bounded variants which ask whether there exists a reconfiguration sequence in which each edge is reversed at most once. These problems are *NP*-complete [2].

Hearn and Demaine [2] consider only a restricted subset of constraint graphs, those which are planar and are constructed using only two specific types of vertices: AND and OR vertices (Figure 2.1).



(a) OR vertex (b) AND vertex

Figure 2.1: The two vertex types from which a restricted constraint graph is constructed: (a) OR vertex and (b) AND vertex. Following the convention set in [1], as a mnemonic weight 2 edges are drawn blue (dark grey) and thick, while weight 1 edges are drawn red (light grey) and thinner.

The OR vertex has minimum inflow 2 and three incident weight 2 edges. Its inflow constraint thus is satisfied if and only if at least one of its incident edges is directed inwards (resembling an OR logic gate). The AND vertex has minimum inflow 2, two incident weight 1 edges and one incident weight 2 edge. Its constraint thus is satisfied if and only if both weight 1 edges are directed inwards or the weight 2 edge is directed inwards (resembling an AND logic gate).

Both C2C and C2E NCL remain *PSPACE*-complete even for this restricted subset of constraint graphs. CGS also remains *NP*-complete under these restrictions. We note that the hardness is in the strong sense, since the weights used are at most 2. We will however consider arbitrary weights, and some cases will turn out to have weak hardness.

Hearn and Demaine define several other constraint logic games and puzzles, complete for classes such as *P*, *EXPTIME*, *NEXPTIME* and even an undecidable game played on constraint graphs. However, we limit ourselves to considering just CGS and NCL.

2.2 H-Word Reconfiguration

To show hardness of NCL on bounded bandwidth graphs, we reduce from the H-Word Reconfiguration problem, introduced in [3].

Definition 2.2 (H-word)**.** Let $H = (\Sigma, E)$ where Σ is an alphabet and $E \subseteq \Sigma \times \Sigma$ a relation. An *H-word* is a word over Σ, such that each pair of consecutive characters (a, b) is an element of E.

H-Word Reconfiguration

Instance: Two H-words W_s, W_g of equal length

Question: Is there a sequence of H-words $W_1, \dots, W_n$, so that every pair of consecutive words W_i, W_{i+1} can be obtained from each other by changing exactly one character to another and $W_1 = W_s, W_n = W_g$?

Theorem 2.3 (Wrochna [3])**.** *There exists an H such that H-Word Reconfiguration is PSPACE-complete.*

In our reduction, the bandwidth of the resulting constraint graph will only depend on H. Due to Theorem 2.3, this means that the problem is hard even when the graph's bandwidth is bounded by a constant.

Wrochna's proof [3] of Theorem 2.3 is very interesting. It shows how to simulate a Turing Machine with a polynomially bounded tape as an H-Word Reconfiguration problem, and is in some sense an analogue of Cook's Theorem [8]. A simplified version of the proof (omitting some technical details) is reproduced here.

Proof. Fix some *PSPACE*-complete language L and a deterministic Turing machine T deciding L whose tape is polynomially bounded. We construct an H so that whether T accepts a given input can be reduced to H-Word Reconfiguration.

Let Q be the set of states of T and Σ be the alphabet of T. We begin by defining a string rewriting problem on the alphabet $\Sigma \cup (\Sigma \times Q)$. We represent a configuration of the machine where the tape contents are $\sigma_1, \ldots, \sigma_n$, the machine is in state q and the machine's head is in position i by a string $\sigma_1, \ldots, \sigma_n$ where σ_i is replaced by (σ_i, q).

The state transitions can now be modelled as follows: if the machine in state q_1 over symbol σ_i would write symbol σ_j, move left and change to state q_2 then we would introduce a string rewriting rule stating that characters $\sigma_k, (\sigma_i, q_1)$ can be replaced by $(\sigma_k, q_2), \sigma_j$. Similarly, we can define rules for transitions where the head moves right or stays in the same place.

These string rewriting rules are of the form $a_1, b_1 \to a_2, b_2$. We simplify this by (for each rule) introducing helper symbols X, Y. We replace the rule $a_1, b_1 \to a_2, b_2$ by a set of four new rules: $a_1, a_2 \to X, a_2$, $X, a_2 \to X, Y$, $X, Y \to b_1, Y$, $b_1, Y \to b_1, b_2$. Now, each rule only changes one symbol at a time.

We now reduce this string rewriting problem over alphabet $\Sigma' = \Sigma \cup (\Sigma \times Q)$ to a H-Word Reconfiguration problem over $H = (\Gamma, E)$ where $\Gamma = \Sigma' \times \Sigma'$.

For $a, b, c \in \Sigma'$, we let $(a, b), (b, c) \subseteq E$. A string $\sigma_1, \sigma_2, \sigma_3, \ldots, \sigma_{n-1}, \sigma_n \in \Sigma'^*$ is then represented as a string $(\sigma_1, \sigma_2), (\sigma_2, \sigma_3), \ldots, (\sigma_{n-1}, \sigma_n) \in \Gamma^*$.

We now need to show how to allow a string rewriting rule of the form $a, b_1 \to a, b_2$ as an operation in H-Word Reconfiguration problem. To this end, we introduce (for each rule) a helper character x and break down this transformation in to

3 steps: $(\cdot, a), (a, b_1), (b_1, \cdot)$ is replaced by $(\cdot, a), x, (b_1, \cdot)$, which is in turn replaced by $(\cdot, a), x, (b_2, \cdot)$, and finally replaced by $(\cdot, a), (a, b_2), (b_2, \cdot)$. Here $\cdot$ is a wildcard representing that any character can take its place. To allow this reconfiguration sequence to happen, we add the following pairs to H: $((\cdot, a), x)$, $(x, (b_1, \cdot))$ and $(x, (b_2, \cdot))$.

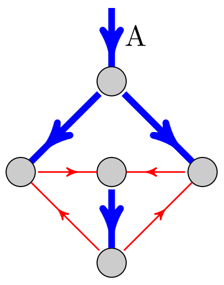
A rule $a_1, b \to a_2, b$ can be represented in a similar way.

For the full details of this proof we refer to [3] but we note one interesting detail: all steps in H-Word Reconfiguration are reversible. The Turing machine computation however, is not reversible. This might seem cause for concern, but since the Turing machine is deterministic, being able to reverse steps does not matter. $\square$

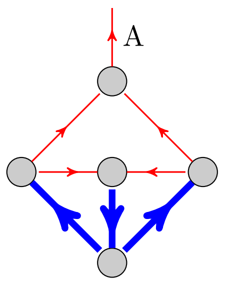
2.3 Constraint Graph Gadgets

In our reductions, we use several constraint graph gadgets due to Hearn and Demaine [2]. For the purpose of being self-contained, we reproduce these gadgets here and state (without proof) their functionality.

Edge Terminators. Hearn and Demaine [2] introduced 3 types of edge terminator gadgets. We do not use the free (blue-)edge terminator, so we only list the two types we do use. The constrained blue-edge (Figure 2.2a) terminator allows us to have a loose blue edge that is forced to point outwards, effectively removing the edge from the graph while still meeting the requirement that the graph is built from only AND and OR



(a) Constrained blue-edge terminator



(b) Free red-edge terminator

Figure 2.2: Gadgets for terminating loose edges. The constrained blue-edge terminator (a) forces the blue edge A to point in towards the gadget at all times, while the free red-edge terminator (b) allows the red edge A to point out of the gadget.

vertices. The free red-edge terminator (Figure 2.2b) allows us to have a loose red edge whose orientation can be freely chosen, effectively decreasing the minimum inflow of the vertex to which it is incident by one.

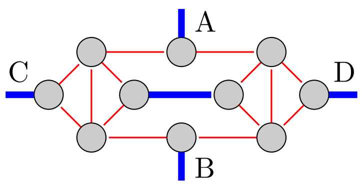
Red-blue Conversion. It is useful to be able to convert a blue edge to a red edge, i.e. we require a gadget which has a blue edge that can (be reconfigured to) point outwards only if its red edge is pointing inwards and vice-versa. Hearn and Demaine [2] provide a construction that allows red-blue conversion in pairs, but also note a simpler construction is possible: an AND vertex, with one of its red edges attached to a free red-edge terminator (Figure 2.2b) can serve as a red-blue conversion gadget.

Crossover Gadget. The crossover gadget (Figure 2.3a) has 4 incident blue edges A, B, C and D, with the property that A can (be reconfigured to) point outward only if B is pointing inwards and vice-versa, with the same property also holding for C and D.

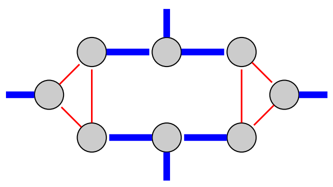
The crossover gadget is constructed using degree-4 vertices (of minimum inflow 2) that are incident to four weight-1 edges. These can be replaced with the half-crossover gadget (Figure 2.3b) to obtain a construction that only uses AND and OR vertices. Note that this replacement requires using red-blue conversion gadgets.

Note that to cross a red edge with a blue edge (or a red edge with another red edge) we can use the aforementioned crossover gadget, paired with red-blue conversion gadgets.

Latch Gadget. The final gadget we require is the latch gadget (Figure 2.4). It can be unlocked by reversing the edge L, after which its state (the orientation of edge A, and adjacent edges) can be changed. The latch can be locked by reversing the edge L



(a) Crossover



(b) Half-crossover

Figure 2.3: The crossover gadget (a) may be constructed with help of the half-crossover gadget (b).

again. It is useful to reduce C2E NCL problems to C2C NCL problems, replacing the target edge in the C2E problem by a latch gadget, and creating a C2C problem in which the goal configuration differs from the start configuration only in the state of the latch gadget.

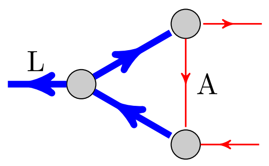


Figure 2.4: The latch gadget can be (un-)locked by reversing edge L, and edge A may reverse if and only if the gadget is unlocked. The gadget is shown in its locked state.

2.4 Graph Parameters

As one possible parametrization we consider a graph's treewidth. The main result (Theorems 3.3 and 3.4) considers graphs of bounded bandwidth, a subclass of graphs of bounded treewidth [9].

Definition 2.4 (Treewidth)**.** A *tree decomposition* of a graph $G = (V, E)$, is a tree T in which every vertex t of T is associated with a bag $X_t \subseteq V$, such that:

- For all $v \in V$, there is a t such that $v \in X_t$
- For all $(u, v) \in E$, there is a t such that $u \in X_t$ and $v \in X_t$
- For all $v \in V$, $T[\{t \in T : v \in X_t\}]$ is connected

The *width* of a tree decomposition is $\max_{t \in T} |X_t| - 1$. The *treewidth* of G is the minimum width over all tree decompositions of G.

Definition 2.5 (Bandwidth)**.** Let $G = (V, E)$ be a graph and define a one-to-one correspondence $f : V \to \{1, \ldots, |V|\}$. The *bandwidth* of a graph is the minimum over all such correspondences of $\max_{(u,v) \in E} |f(u) - f(v)|$.

Finally, in the proof of Theorem 5.3 we use the notion of cutwidth:

Definition 2.6 (Cutwidth)**.** Let $G = (V, E)$ be a graph and define a one-to-one correspondence $f : V \to \{1, \ldots, |V|\}$. The *cutwidth* of a graph is the minimum over all such correspondences of $\max_{w \in V} |\{(u, v) \in E | f(u) \leq f(w) < f(v)\}|$.

A graph that has bounded bandwidth also has bounded cutwidth [9].

2.5 Dynamic Programming on Tree Decompositions

For some of the positive results in this paper, we use the technique of dynamic programming on tree decompositions. For an excellent survey of and introduction to this topic, see [10].

To simplify the algorithms, we assume tree decompositions are given in nice form, that is, all of the nodes of the tree decomposition are of one of the following types:

- **Leaf**: all leaves of the tree decomposition contain exactly one vertex.
- **Introduce**: a node with one child that contains the same set of vertices as its child, with the addition of one new vertex.
- **Forget**: a node with one child that contains the same set of vertices as its child, but with one vertex removed.
- **Join**: a node with two children so that both the node itself and both of its children contain the exact same set of vertices.

Given a tree decomposition, it is possible to find a nice tree decomposition of the same width (and of linear size) in linear time.

Chapter 3

Parametrization by Treewidth

In this section, we study the complexity of constraint graph problems when parametrized by treewidth, and by treewidth plus maximum degree. We first examine Constraint Graph Satisfiability, then prove our main result, showing that (unbounded) Nondeterministic Constraint Logic is *PSPACE*-complete on graphs of bounded bandwidth. Finally we look at the bounded variant of Nondeterministic Constraint Logic.

3.1 Constraint Graph Satisfiability

We first examine Constraint Graph Satisfiability (CGS) with respect to treewidth $\mathcal{K}$ and maximum degree Δ as parameters. CGS is strongly *NP*-complete even for graphs where $\Delta = 3$ [2]. We show that CGS is weakly *NP*-complete for graphs of treewidth at most 2, but that CGS is fixed parameter tractable with respect to $\mathcal{K} + \Delta$ and solvable in polynomial time for fixed $\mathcal{K}$ if the input is given in unary. Note that even though CGS becomes fixed parameter tractable with respect to $\mathcal{K} + \Delta$, its reconfiguration variant remains hard (Theorem 3.3) even when this parameter is bounded by a constant.

Theorem 3.1. Constraint Graph Satisfiability *is weakly NP-complete on constraint graphs of treewidth* 2.

Proof. By reduction from Partition. Let $x_1, \ldots, x_n \in \mathbb{N}$ such that $\Sigma_{i=1}^{n} x_i = 2N$ be an instance of Partition. Construct a constraint graph with vertices U, W with minimum inflow N, and vertices $v_1 \ldots, v_n$ where the minimum inflow of v_i is equal to x_i. For

$1 \leq i \leq n$, create edges (U, v_i), (W, v_i), both of weight x_i. In any legal configuration for this constraint graph, either edge (U, v_i) or (W, v_i) must be directed towards vertex v_i and the other edge will be directed towards either U or W. A solution to the partition problem and a legal configuration for the constraint graph correspond as follows: for all x_i in one half of the partition, the edge (U, v_i) is directed towards U (and the edge (W, v_i) is directed towards v_i) and for all x_i in the other half of the partition, the edge (W, v_i) is directed towards W (and the edge (U, v_i) is directed towards v_i). The graph restricted to vertices $\{v_1, \ldots, v_n, U\}$ forms a tree, and thus the constraint graph itself has treewidth 2 (we can add the vertex W to all bags). We have thus shown CGS weakly *NP*-hard on constraint graphs of treewidth 2. □

The following theorem shows that CGS can be solved in polynomial time on graphs of treewidth 2 when the input is given in unary (and completes the proof of theorem 3.1).

Theorem 3.2. Constraint Graph Satisfiability *is fixed parameter tractable with respect to parameter* $\mathcal{K} + \Delta$ *and solvable in time* $O^*(|x|^{\mathcal{K}})$ *on constraint graphs of treewidth* $\mathcal{K}$ *when the size of the input numbers given in unary is* $|x|$.

Proof. We provide a dynamic programming algorithm on tree decompositions. In the case where Δ is a parameter, in each bag we store for every combination of orientations of the edges incident to vertices in that bag (at most $2^{\Delta(\mathcal{K}+1)}$ combinations) whether that combination leads to a valid configuration for the subgraph induced by the subtree rooted at that bag. We assume that the tree decomposition is given as a nice tree decomposition:

- **Leaf**: In a leaf node, we enumerate all possible combinations of orientations for edges incident to this vertex (at most 2^{Δ}), and see which combinations satisfy the minimum inflow of that vertex.
- **Introduce**: In an introduce node, enumerate all combinations of orientations of edges incident to the vertices in the bag (at most $2^{\Delta(\mathcal{K}+1)}$). A combination is valid if it satisfies the inflow of the vertex being introduced, and (when restricted to the appropriate edges) is also a valid assignment of orientations for the child bag.
- **Forget**: In a forget node, an assignment of orientations is valid if it can be extended (by picking appropriate orientations for the missing edges) to a valid configuration for the child bag.

- **Join**: In a join node, an assignment of orientations is valid if it is valid for both child nodes.

The amount of work done in each bag is $O^*(2^{\Delta(\mathcal{K}+1)})$, so the total running time is $O^*(|I| \cdot 2^{\Delta(\mathcal{K}+1)})$, where $|I|$ is the number of bags (which can be assumed to be linear in the size of the input graph, since there always a linear size tree decomposition).

A similar algorithm can be used in the case where the input is given in unary. Instead of storing the orientations of the edges directly, we store only the resulting inflow values. For all possible combinations of inflow that the vertices in the bag can be receiving (at most $|x|^{\mathcal{K}+1}$ combinations), we store whether that combination of inflows is attainable using only the edges present in the subgraph induced by the subtree rooted in that bag (i.e. edges to vertices in parent bags are not considered) in configurations that are legal for the vertices in the induced subgraph excluding the vertices in the bag itself. This requires (in particular) modification of the forget case, marking as invalid any inflow configurations that do not satisfy the minimum inflow of the vertex being forgotten (detection of whether a vertex is receiving sufficient inflow is deferred to the forget node instead of being checked in the leaf or introduce cases). □

3.2 Unbounded Nondeterministic Constraint Logic

In this section we prove the main result, namely that restricted Nondeterministic Constraint Logic remains *PSPACE*-complete even when restricted to graphs of bounded bandwidth (which is a subclass of graphs of bounded treewidth).

To show *PSPACE*-completeness, we reduce from H-Word Reconfiguration. The bandwidth of the constraint graph created in the reduction will depend only on the size of H. Since H-Word Reconfiguration is *PSPACE*-complete for a fixed (finite) H, we obtain a constant bound on the bandwidth.

Theorem 3.3. *There exists a constant c, such that C2C* Nondeterministic Constraint Logic *is PSPACE-complete on planar constraint graphs of bandwidth at most c that use only AND and OR vertices.*

Proof. Let $H = (\Sigma, E)$ be so that H-Word Reconfiguration is *PSPACE*-complete. We provide a reduction from H-Word Reconfiguration to (unrestricted) NCL and then show how to adapt this reduction to work for restricted NCL.

Let W_s, W_g be an instance of H-Word Reconfiguration and let n denote the length of W_s. In the following, all vertices are given minimum inflow 2.

We create a matrix of vertices $X_{i,j}$ for $i \in \{1, \dots, n\}, j \in \Sigma$, the *character vertices.* $X_{i,j}$ corresponds to whether the character at position i in the word is character j. For every row i of this matrix we create a *universal vertex* U_i and for every $j \in \Sigma$ we create a blue (weight 2) edge connecting U_i and $X_{i,j}$.

In each row $i \in 1, \dots, n-1$ and for all pairs $(A, B) \notin E$, we create a *relation vertex* $\Delta_{i,A,B}$. We create a red (weight 1) edge connected to $X_{i,A}$ and pass it through a red-blue conversion gadget and connect the blue edge leaving the conversion gadget to $\Delta_{i,A,B}$. Similarly, we create a red edge connected to $X_{i+1,B}$, convert it to blue, and connect it to $\Delta_{i,A,B}$. Finally, we connect one additional blue edge to $\Delta_{i,A,B}$ and connect it to a constrained blue-edge terminator. This edge serves no purpose (its inflow can never count towards $\Delta_{i,A,B}$) except to make $\Delta_{i,A,B}$ an OR vertex as claimed.

A single instance of this construction is shown in figure 3.1, which depicts a slice of two rows from the matrix of vertices. Having created an instance of this construction for A, B, we might need to create an instance for A, C. Rather than attaching another red edge to $X_{i,A}$, we instead split the existing red edge leaving $X_{i,A}$ by connecting it to a red-blue conversion gadget, and attaching the resulting blue edge to an AND vertex. The edges leaving the AND vertex may be oriented outward if and only if the red edge leaving $X_{i,A}$ is oriented in to the AND vertex, effectively splitting it. We then convert two red edges of the AND vertex to blue and attach them to $\Delta_{i,A,B}$ and $\Delta_{i,A,C}$ as before. To create further copies of this construction we can repeat the splitting process, so that $X_{i,A}$ eventually has two incident red edges (where one of the red edges is used to connect to vertices in the next layer, and the other used to connect to vertices in the previous layer, making it an AND vertex as claimed), one connecting to gadgets in layer $i-1$ and one connecting to layer $i+1$ (the excess red edges in rows 1 and n may instead be connected to a free red edge terminator).

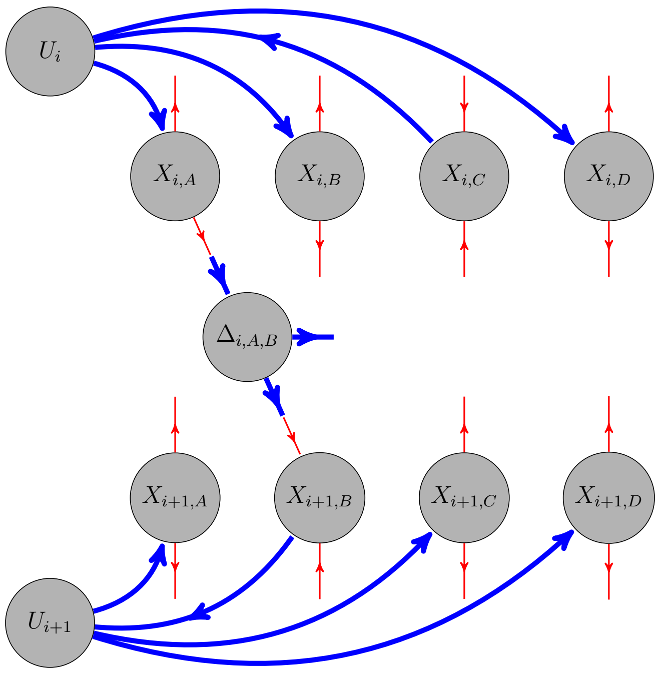


Figure 3.1: Slice of two rows from the matrix of vertices (over an alphabet of $\Sigma = \{A, B, C, D\}$), and the construction for enforcing non-adjacency of A and B.

We define a character j to be *in* the word at position i if the edge $(U_i, X_{i,j})$ is oriented towards U_i.

Claim. In any legal configuration, at least one character is in the word at each position.

Proof. U_i has minimum inflow 2, so at least one of its incident edges $(U_i, X_{i,j})$ must be oriented towards it and hence j is in the word at position i. □

Claim. In any legal configuration, if $(A, B) \notin E$ then A can be in the word at position i only if B is not in the word at position $i + 1$.

Proof. If A is in the word at position i, then $X_{i,A}$ is not receiving inflow from U_i, so both of the red edges incident to $X_{i,A}$ most point in towards $X_{i,A}$. After passing through the red-blue conversion gadgets and possible AND vertices splitting the red edge, the associated blue edge incident to $\Delta_{i,A,B}$ must point out of $\Delta_{i,A,B}$ (and in the direction of $X_{i,A}$ to help satisfy its minimum inflow). Hence the other blue edge incident to $\Delta_{i,A,B}$ (the third blue edge is constrained by the blue-edge terminator) must point in towards $\Delta_{i,A,B}$. Again passing through the conversion gadgets and the splitting AND vertices, the associated red edge must point out of $X_{i+1,B}$ (and in the direction of $\Delta_{i,A,B}$ to help satisfy its minimum inflow) and hence B can not be in the word at position $i + 1$ as claimed. □

Note that this claim implies that if for each position we pick *some* character that is in the position at that word, we end up with a valid H-Word. We say that a configuration for the constraint graph *encodes* a word W if each of that word's characters is in the word at the appropriate position.

Claim. If a legal configuration for the constraint graph encoding W_s may be reconfigured in to a legal configuration encoding only W_g (and that encodes no other word), then W_s may be reconfigured in to W_g.

Proof. Suppose we have a reconfiguration sequence of legal configurations $C_1, \dots, C_m$ so that C_1 encodes W_s and C_m encodes only W_g. We define a reconfiguration sequence of words $W_1, \dots, W_m$ which will have the property that C_i encodes W_i. Let $W_1 = W_s$, we recursively define $W_i, 1 < i \leq m$ as follows: Since C_{i+1} differs from C_i in the orientation of only one edge, the set of words encoded by C_{i+1} differs only from the set of words encoded by C_i by making one character at a position allowed (in the word) or disallowed (no longer in the word). If a character at a position becomes allowed in C_{i+1}, let $W_{i+1} = W_i$ (since C_i encodes W_i, C_{i+1} also encodes W_i). If a character at a position becomes disallowed, we obtain W_{i+1} from W_i by changing the character at that position to some allowed character (of which there is at least one). Following these steps, since the final configuration encodes only W_g, we obtain a reconfiguration sequence from W_s to W_g as claimed. □

Claim. Given a H-word W of length n, there exists a legal configuration for the constraint graph encoding only W.

Proof. Pick the orientation of edge $(U_i, X_{i,j})$ to be towards U_i if the character in W at position i is j, and towards $X_{i,j}$ otherwise. Clearly this constraint graph encodes only W. This configuration can be extended to a legal configuration for the remaining (relation) vertices by noticing the following: if there is a relation vertex $\Delta i, A, B$ then either A is not in the word at position i or B is not in the word at position j. Suppose w.l.g. that B is not in the word, then $X_{i+1,B}$ is receiving inflow from U_{i+1} and hence its incident red edges may be pointing outwards, which (after passing through the red-blue conversion gadgets and splitting AND vertices) allows us to satisfy the inflow requirement of $\Delta_{i,A,B}$. □

Claim. If two H-Words W_1, W_2 differ by only one character, then a constraint graph G encoding only W_1 can be reconfigured in to one encoding only W_2.

Proof. Suppose that W_1 and W_2 differ by changing the character at position i from A to B. We may first reconfigure G from encoding only W_1 to encoding both W_1 and W_2, and then reconfigure it to encode only W_2. This temporarily increases the inflow U_i receives from 2 (receiving inflow only from $X_{i,A}$) to 4 (receiving inflow from both $X_{i,A}$ and $X_{i,B}$) back to 2 (receiving inflow from only $X_{i,A}$). It may be required to reconfigure some edges in relation to relation vertices, but this is always possible because neither A nor B conflicts with any preceding or succeeding characters. □

Note that this claim implies that if W_s can be reconfigured in to W_g, then a constraint graph encoding only W_s can be reconfigured in to one encoding only W_g. This completes the reduction.

All that remains to show is that this graph can be adapted so that it only uses AND and OR vertices and is planar, and that the resulting graph has bounded bandwidth. The only vertices that are not already AND or OR vertices are the universal vertices U_i. These may be replaced by a binary tree of OR vertices (or some other structure) that repeatedly splits a blue edge, similar to splitting the red edges.

To make the graph planar, we can use Hearn and Demaine's crossover gadget. While this may increase the graph's bandwidth, this argument shows that it remains bounded by a constant:

We obtain a bag decomposition by (for each i) taking all vertices $U_i, \{X_{i,j} : j \in E\}, \{\Delta_{i,A,B} : (A,B) \notin E\}, \{X_{i+1,j} : j \in E\}$ and U_{i+1} and the vertices in gadgets connecting them in a single bag B_i. Since the size of such a bag depends only on H (which is fixed) and edges only run either between vertices in the same bag B_i, or from vertices in a bag B_i to vertices in a bag B_{i-1} or from vertices in a bag B_i to vertices in a bag B_{i+1}, the bandwidth of this graph is at most a constant c.

We have thus shown how to reduce H-WORD RECONFIGURATION to NCL, shown how to adapt our reduction to use only AND and OR vertices, how make the graph planar, that the resulting graph has bounded bandwidth and have thus shown Theorem 3.3. □

Theorem 3.3 also holds for C2E NONDETERMINISTIC CONSTRAINT LOGIC:

Theorem 3.4. *There is a constant c, such that C2E* NONDETERMINISTIC CONSTRAINT LOGIC *is PSPACE-complete, even on planar constraint graphs of treewidth (bandwidth) at most c that use only AND and OR vertices.*

Proof. H-Word Reconfiguration remains *PSPACE*-complete, even when instead of requiring that one H-word is reconfigured in to another, we ask whether it is possible to reconfigure a given H-word so that a given character appears at a specific position (without requiring anything regarding the remaining characters in the word). This can be seen by examining the proof in [3], noting that we may modify the Turing machine to, upon reaching an accepting state, move the head to the start of the tape and write a specific symbol there. The proof of Theorem 3.3 can then easily be adapted to work for C2E NCL (since a character appearing at a specific position corresponds to reversing a specific edge). □

We note that Theorems 3.3 and 3.4 may be further strengthened by requiring that all OR vertices in the graph are *protected*, that is, in any legal configuration at least one of its incident edges is directed outwards. Hearn and Demaine [2] show how to construct an OR vertex using only AND vertices and protected OR vertices.

It is interesting that while CGS is fixed parameter tractable with respect to treewidth plus maximum degree, NCL remains hard even for fixed values of these parameters. Much like how shortest path reconfiguration is *PSPACE*-complete [11] while finding a shortest path is in P, this gives another example of how a reconfiguration problem can be much harder than the respective decision problem. We note that the converse is also possible: finding a graph 3-colouring is a classical *NP*-complete problem but reconfiguring 3-colourings is in P [12].

3.3 Bounded Nondeterministic Constraint Logic

We now consider the bounded variant of NCL, in which each edge is allowed to reverse at most once. We show that the bounded variants of NCL are weakly *NP*-hard on graphs of treewidth at most 2, but, unlike their unbounded variants, fixed parameter tractable when parameterized by $\mathcal{K} + \Delta$.

Theorem 3.5. *Bounded NCL (both C2E and C2C) is weakly NP-hard on constraint graphs of treewidth 2.*

Proof. By reduction from Partition. Construct an NCL graph in the same manner as in the proof of Theorem 3.1, but create an additional edge (U, W) of weight N. In the initial configuration, let the edge (U, W) be directed towards W, the edges (U, v_i) towards U and the edges (W, v_i) towards v_i.

- For the C2E variant, ask whether it is possible to reverse the edge (U, W).
- For the C2C variant, ask whether the configuration in which all edges are reversed is reachable.

It is possible to reverse (U, W) is and only if there is a partition: initially, vertex U is receiving $2N$ units of inflow (from the edges (W, v_i)) and V is receiving N units of inflow (from the edge (U, W)). To reverse (U, W), we must reverse some of the edges (U, v_i) and (W, v_i) so that W is receiving exactly N units of inflow from them. If W is receiving less than N units of inflow from these edges it is not possible to reverse (U, W), if it is receiving more than N inflow then V would not be receiving enough inflow. Being able to select these edges so that W receives exactly N units of inflow from them corresponds to a partition.

This completes the proof for the C2E variant. For the C2C variant, note that after reversing (U, W) we may reverse the remaining edges (that have not been reversed yet) while the graph remains satisfied: after reversing (U, W), U is receiving $2N$ units of inflow and W is receiving N units, after reversing the remaining edges U will be receiving N units of inflow and W will be receiving $2N$ units. □

Bounded NCL (on graphs of treewidth 2) is clearly in *NP*, however, it is not clear whether there exists a pseudopolynomial algorithm for the problem, so we can not claim weak *NP*-completeness. This is an open problem.

Theorem 3.6. *Bounded NCL (both C2E and C2C) is fixed parameter tractable with respect to parameter $\mathcal{K} + \Delta$.*

Proof. The problems can be solved by dynamic programming on tree decompositions. For every bag, we consider all possible sequences of reversing the edges in that bag (since each edge may be reversed at most once, there are at most $(\Delta\mathcal{K})!$ such sequences) that result in a valid solution for the subgraph induced by the subtree rooted at that bag. For C2C, we additionally require that a reversal sequence reverses exactly those edges that need to be reversed. For C2E, we additionally require that the reversal sequence reverses the target edge. We assume that the tree decomposition is given as a nice tree decomposition:

- **Leaf**: Enumerate all reversal sequences of incident edges (at most $\Delta!$). For the C2C variant, a sequence is valid if it reverses exactly the edges that must be reversed and additionally respects the minimum inflow of the vertex. For C2E, all sequences that respect the minimum inflow of the vertex are valid, unless the vertex is incident to the target edge in which case we additionally require that a valid reversal sequence reverses the target edge.
- **Introduce**: Enumerate all reversal sequences of incident edges (at most $(\Delta\mathcal{K})!$ sequences). A sequence is valid if it is valid for the vertex being introduced (as described in the leaf case) and also valid for the child bag (when restricted to the appropriate edges).
- **Forget**: A reversal sequence is valid if and only if it can be extended (by introducing reversal operations for the missing edges) to a valid reversal sequence for the child bag. Note that this is simply a projection of the reversal sequences valid for the child.
- **Join**: A reversal sequence is valid if and only if it is valid for both child bags.

Since checking whether a reversal sequence is valid can be done in polynomial time (if the memoization tables are stored in a suitable data structure), the amount of work done in each bag is $O^*((\Delta\mathcal{K})!)$. Since the number of bags can be assumed to be linear [10], this gives a fixed-parameter tractable algorithm. $\square$

Chapter 4

Parametrization by Solution Length

We consider parametrization by the length of the reconfiguration sequence l. This parameter does not bound the input size, but rather bounds the set of potential solutions considered. It asks not whether a reconfiguration sequence exists, but whether a reconfiguration sequence exists that has length (number of edge reversals) at most l. This parametrization does not apply to CGS (since it does not have such a notion), we thus consider this parametrization only for C2E and C2C NCL and their respective bounded variants.

Theorem 4.1. *(Unbounded) C2E, C2C and bounded C2E* Nondeterministic Constraint Logic *are* $W[1]$*-hard with respect to* l*.*

Proof. We show $W[1]$-hardness by reduction from k-Clique, which is $W[1]$-complete due to Downey and Fellows [13]:

Let $G = (V, E)$ be an instance of k-Clique. Let $n = |V|$ and let Δ be the maximum degree of G. We create a constraint graph with vertices $v_1, \ldots, v_n$ (copies of the original vertices from G) with minimum inflow equal to that vertex' degree in G, vertices $v_{(i,j)}$ with minimum inflow 2 for all edges $(i, j) \in E$, and three additional vertices U, V, W with minimum inflows $n - \Delta \cdot k$, $k(k-1)$ and 0 respectively.

For every edge $(i, j) \in E$, two edges in the constraint graph are created: an edge $(v_i, v_{(i,j)})$ and an edge $(v_{(i,j)}, v_j)$, both with weight 1. Effectively, the edge (i, j) is split and an additional vertex $v_{(i,j)}$ is added between vertices v_i and v_j. In the initial configuration, both edges are oriented away from $v_{(i,j)}$ (and towards v_i and v_j). Note that after adding these edges, the minimum inflow constraints in all vertices $v_1, \ldots, v_n$ are satisfied (since their minimum inflow is equal to their degree).

For every vertex v_i, add an edge (U, v_i) to the graph with weight Δ. The edge is oriented towards U in the initial configuration, satisfying its inflow constraint.

For every edge $(i,j) \in E$, create an edge $(v_{(i,j)}, V)$ of weight 2, oriented towards $v_{(i,j)}$ in the initial configuration. This satisfies the minimum inflow constraint of $v_{(i,j)}$.

Finally, create an edge (V, W) of weight $k(k-1)$, oriented towards V in the initial configuration, thus satisfying V's minimum inflow. This edge is the target edge. Note that W's minimum inflow is trivially satisfied.

The target edge (V, W) can be reversed if and only if we have first reversed $\frac{k(k-1)}{2}$ of the weight 2 edges to point in to V (or else V's inflow constraint would no longer be satisfied after reversing (V, W). A weight 2 edge $(v_{(i,j)}, V)$ can be reversed if and only if we have first reversed the weight 1 edges $(v_i, v_{(i,j)})$ and $(v_{(i,j)}, v_j)$ (or else $v_{(i,j)}$'s minimum inflow would not remain satisfied). This in turn is possible if and only if both edges (U, v_i) and (U, v_j) have been reversed.

Because the minimum inflow of U is $n - \Delta \cdot k$, we may reverse at most k edges to point away from U. Since we may reverse the edge $(v_{(i,j)}, V)$ if and only if both edges (U, v_i) and (U, v_j) have been reversed, the NCL instance thus constructed has a solution if and only if there are k vertices in G such that G restricted to these vertices has $\frac{k(k-1)}{2}$ edges. This corresponds to these vertices forming a clique.

We have thus reduced k-Clique to C2E NCL. Note that if the instance created in the reduction has a solution of any length, then it has a solution of length at most $O(k^2)$ in which each edge reverses at most once. Hence the reduction is parameter-preserving, and works for both bounded and unbounded C2E NCL which we have thus shown $W[1]$-hard.

To show $W[1]$-hardness for C2C NCL, we replace the vertex W with a latch gadget. The latch gadget has two states, and can change between these states if and only if the edge (U, W) is pointing in to it. We ask whether there is a reconfiguration sequence that changes the state of the latch gadget, but where all the other edges are in their initial orientation. Such a sequence exists if and only if (in the original instance) it is possible to reverse the edge (U, W), since we may first reverse (U, W), activate the latch gadget, and then backtrack to return the rest of the graph to its original state. □

The construction using the latch gadget does not work for bounded C2C NCL, since each edge can only be reversed once. Effectively, the hardness of NCL when parameterized by solution length is not due to the complexity of finding a reconfiguration sequence, but rather that of finding a suitable goal configuration. This is captured by the following theorem:

Theorem 4.2. *Bounded C2C* Nondeterministic Constraint Logic *is fixed parameter tractable in $O^*(2^l)$ time.*

Proof. If the number of edges that differ in orientation between the goal and start configuration exceeds l, we may reject immediately because is is impossible to reverse all of them in l moves. We may thus assume at most l edges need to be reversed. Since in bounded NCL each edge may be reversed at most once, feasible solutions consist of some order of reversing the edges that need to be reversed; reversing an edge that does not need to be reversed makes the instance unsolvable, and reversing an edge more than once is not possible. There are thus at most $l!$ solutions each of which can be evaluated in polynomial time. This can be improved to a $O^*(2^l)$ time algorithm using a dynamic programming approach, similar to the Held-Karp algorithm [14] for TSP. □

Parametrization by Maximum Degree. The graphs created by the reduction in the proof of Theorem 4.1 may have arbitrary degree, while Hearn et al. [2] consider only graphs of maximum degree 3. We show that the hardness result of Theorem 4.1 is strict in this regard, that is, when we parameterize by the maximum degree Δ in addition to l, the problem becomes fixed parameter tractable.

Theorem 4.3. *Both bounded and unbounded C2E and C2C* Nondeterministic Constraint Logic *are fixed parameter tractable with respect to $l + \Delta$.*

Proof. For the *C*2*E* variants, note that an edge that is at a distance greater than or equal to l from the target edge never needs to be reversed in a solution of length at most l: The only valid solution of length at most 1 is to reverse the target edge, which indeed is at distance 0 from itself. Consider a solution of length at most $l + 1$ and that it reverses an edge e at distance $l + 1$ or greater from the target edge. After this edge is reversed, we are left with a solution of length at most l which by induction only needs to reverse edges at distances less than l from the target edge. We note that we can omit the reversal of e from the reconfiguration sequence, since the vertex that gains inflow as a result of that reversal is not incident to any of the edges reversed subsequently, and hence omitting this reversal also results in a valid solution.

For any candidate solution, we only need to consider edges at distance at most $l-1$ from the target edge, of which there are at most $O((\Delta-1)^{l-1})$. We may omit the other edges from the graph (adjusting the minimum inflows of their incident vertices according to the initial orientations of the edges) and consider only the graph restricted to these edges and their incident vertices.

In the $C2C$ variants, similar to the proof of Theorem 4.2, we may assume at most l edges need to be reversed. We can thus bound the size of the set of edges that can be reversed in any candidate solution by $O(l(\Delta-1)^{l-1})$, since as before, for any of the l edges needing to be reversed, we are only concerned with the edges at distance at most l from that edge. As before, we can omit the remaining edges from the graph.

The problems thus admit kernelizations and are therefore fixed parameter tractable. $\square$

Chapter 5

Applications

As a result of our hardness proof for NCL, we (nearly) automatically obtain tighter bounds for classical problems that reduce from NCL in their hardness proofs. If the reduction showing *PSPACE*-hardness for a problem is bandwidth-preserving in the sense that the bandwidth of the resulting graph only depends on the bandwidth of the original constraint graph, then that problem remains *PSPACE*-hard even when limited to instances of bounded bandwidth. Since reductions from NCL usually work by locally replacing AND and OR vertices in the graph with gadgets that simulate their functionality, such reductions are almost always bandwidth-preserving.

Theorem 5.1. Independent Set *(IS-R),* Vertex Cover *(VC-R),* Feedback Vertex Set *(FVS-R),* Induced Forest *(IF-R),* Odd Cycle Transversal *(OCT-R) and* Induced Bipartite Subgraph *(IBS-R) are PSPACE-complete on planar graphs of bounded bandwidth.*

Proof. Hearn and Demaine [15] provide a reduction from restricted NCL to IS-R. This reduction is for reconfiguration under a Token Sliding (TS) model, however the same proof also works for Token Jumping (TJ) which in turn is equivalent to Token Addition and Removal (TAR). For definitions and this equivalence, see [6]. In this thesis, we consider the TAR model.

By Theorem 3.3, restricted NCL is *PSPACE*-complete on graphs of bounded bandwidth. The reduction from restricted NCL to IS-R in [16] creates planar, maximum degree 3 graphs. Because the reduction works by replacing every AND and OR vertex with a construction of at most 9 vertices, the bandwidth is at most 9 times the bandwidth of the input graph. IS-R is thus *PSPACE*-complete on planar, maximum degree 3 graphs of bounded bandwidth.

The following argument (due to Wrochna [3]) shows that this hardness result also applies to the other problems: Since a vertex cover is the complement of an independent set, VC-R is also *PSPACE*-complete on planar, maximum degree 3 graphs of bounded bandwidth. Replacing edges with triangles (increasing the bandwidth by at most a multiplicative factor) shows that FVS-R, IF-R, OCT-R and IBS-R are *PSPACE*-complete on planar, maximum degree 6 graphs of bounded bandwidth. □

Another related reconfiguration problem is that of Dominating Set Reconfiguration (DS-R). In [7], the authors show that DS-R is *PSPACE*-complete on planar graphs of maximum degree six by reduction from NCL and *PSPACE*-complete on graphs of bounded bandwidth by a reduction from VC-R. The following theorem that unifies these results follows immediately from our improved result concerning VC-R:

Theorem 5.2. Dominating Set Reconfiguration *is PSPACE-complete, even on planar, maximum degree six graphs of bounded bandwidth.*

The implications of our result showing that NCL is *PSPACE*-complete even on bounded bandwidth graphs are not limited to graph problems. For instance, the puzzle Rush Hour, in which the player moves cars horizontally and vertically with the goal to free a specific car from the board, is *PSPACE*-complete [17] when played on an $n \times n$ board. A consequence of our result is that Rush Hour is *PSPACE*-complete even on rectangular boards where one of the dimensions of the board is constant:

Theorem 5.3. *There exists a constant c, so that Rush Hour is PSPACE-complete when played on boards of size $n \times c$.*

Proof. Hearn and Demaine [16] provide a reduction from restricted NCL, showing how to construct AND and OR vertices as gadgets in Rush Hour and how to connect them together using straight line and turn gadgets. Given a planar constraint graph, it can be drawn in the grid (with vertices on points of the grid and edges running along the lines of the grid), after which vertices can be replaced by their appropriate gadgets and edges with the necessary line and turn gadgets. However, starting with a graph of bounded bandwidth does not immediately give a suitable drawing in a grid of which one of the dimensions is bounded (from which the theorem would follow, since all of the gadgets have constant size).

Starting with a restricted NCL graph of bounded bandwidth, we note that this graph also has bounded cutwidth [9]. Given a restricted NCL graph on n vertices with cutwidth k, it is possible to arrange it in a $3n \times (k+1)$ grid, so that vertices of the graph are mapped to vertices of the grid, the edges of the graph run along edges of the grid, and no two edges or vertices get mapped to the same edge or vertex in the grid. This is achieved by placing all of the vertices of the graph in a single row (noting that 3 columns are required for each vertex since it has 3 incident edges) and using the remaining k rows for the edges (placing each edge in a distinct row). However, even when starting with a planar NCL graph, the resulting graph may have crossings. These can be eliminated using the crossover gadget, noting that since the crossover gadget has constant size, the resulting graph can still be drawn in a $O(n) \times O(k)$ grid. We can now use the existing gadgets from [16] to finish the reduction, showing Rush Hour *PSPACE*-complete on boards of which one of the dimensions is bounded by a constant. $\square$

Chapter 6

Conclusions

We have studied the parameterized complexity of constraint logic problems with regards to (combinations of) solution length, maximum degree and treewidth as parameters. As a main result, we showed that NONDETERMINISTIC CONSTRAINT LOGIC is *PSPACE*-complete on planar, 3-regular graphs of bounded bandwidth constructed using only AND and (protected) OR vertices. We gave several applications of this result, showing reconfiguration versions of several classical graph problems *PSPACE*-hard on planar graphs of bounded bandwidth, as well as showing Rush Hour *PSPACE*-complete when played on a rectangular board where one of the lengths of the sides is bounded by a constant.

We showed that when parameterized by solution length, C2E, C2C and bounded C2E variants of NONDETERMINISTIC CONSTRAINT LOGIC are $W[1]$-hard, but bounded C2C NCL becomes fixed parameter tractable. Essentially, the hardness of bounded C2E NCL when parameterized by solution length is due to the complexity of finding a suitable target configuration, rather than from that of determining a reconfiguration sequence. However, when parametrizing by maximum degree in addition to solution length, all cases become fixed parameter tractable.

When parameterized by treewidth, CONSTRAINT GRAPH SATISFIABILITY becomes weakly *NP*-complete (rather than strongly *NP*-complete). When considering maximum degree in addition to treewidth, CGS becomes fixed parameter tractable. Note that this is in stark contrast to the complexity of NCL, which remains *PSPACE*-complete even for fixed values of this parameter. This is an interesting example of how reconfiguration problems can be much harder than their decision variants.

Bounded NCL also becomes fixed parameter tractable when parameterized by treewidth and maximum degree. In the case where bounded NCL is parameterized by treewidth alone we proved weak *NP*-hardness. It is an open problem whether there exists a pseudo-polynomial algorithm for this case, or if the problem is also strongly *NP*-complete.

We have strengthened Hearn and Demaine's framework [2] by proving that NCL remains *PSPACE*-complete even on graphs of bounded bandwidth. This result benefits any reduction from NCL that preserves the bandwidth of the original constraint graph.

By combining Wrochna's proof [3] and Hearn and Demaine's [2] constraint logic techniques, we have managed to get the best of both worlds: for a several reconfiguration problems, we showed hardness for graphs that are not only planar and have low maximum degree, but that also have bounded bandwidth. This not only strengthens Wrochna's results, but also makes it easier to prove hardness for reconfiguration on bounded bandwidth graphs by distilling Wrochna's proof in to Hearn and Demaine's NCL framework.

The prior work of Raman et al. [5] and Wrochna [3] has already exhibited problems that, even though their satisfiability version are fixed parameter tractable with respect to treewidth, their reconfiguration versions remain hard when parameterized by treewidth or even bandwidth. This raises the question of, if treewidth and bandwidth are not good parameters for bringing down the complexity of reconfiguration problems, then on what kinds of graphs do reconfiguration problems become easy? For instance, the Token Jumping model of Independent Set Reconfiguration is known to be fixed parameter tractable for certain classes of planar graphs [18], and also on cographs [19]. These classes might be suitable candidates on which NCL is also tractable.

We have studied the parameterized complexity of CGS and NCL. Hearn and Demaine [2] have defined several other problems related to constraint graphs, including two player and multiple player constraint graph games, complete for classes such as *EXPTIME* and *NEXPTIME*. Studying these problems in a parameterized setting gives rise to several interesting open problems.

Bibliography

[1] Hearn, Robert A., Demaine, E.D.: The nondeterministic constraint logic model of computation: Reductions and applications. In: Widmayer, P., Eidenbenz, S., Triguero, F., Morales, R., Conejo, R., Hennessy, M. Automata, Languages and Programming. LNCS, vol. 2380, pp. 401-413. Springer, Heidelberg (2002)

[2] Hearn, R.A., Demaine, E.D.: Games, Puzzles, and Computation. AK Peters, Wellesley (2009)

[3] Wrochna, M.: Reconfiguration in bounded bandwidth and treedepth. arXiv preprint, arXiv:1405.0847 (2014)

[4] Mouawad, A.E., Nishimura, N., Raman, V., Simjour, N., Suzuki A.: On the parameterized complexity of reconfiguration problems. In: Gutin, G., Szeider, S. Parameterized and Exact Computation. LNCS, vol. 8246, pp 281-294. Springer, Heidelberg (2013)

[5] Mouawad, A.E., Nishimura, N. Raman, V., Wrochna, M.: Reconfiguration over Tree Decompositions. In: Cygan, M., Heggernes, P. Parameterized and Exact Computation. LNCS, vol. 8894, pp. 246-257. Springer, Heidelberg (2014).

[6] Kamiński, M., Medvedev, P., Milanič, M.: Complexity of independent set reconfigurability problems. Theoretical Computer Science, vol. 439, pp. 9-15. (2012)

[7] Haddadan, A., Ito, T., Mouawad, A.E., Nishimura, N., Ono, H., Suzuki, A., Tebbal, Y.: The complexity of dominating set reconfiguration. arXiv:1503.00833 (2015)

[8] Cook, S.A.: The complexity of theorem-proving procedures. Proceedings of the third annual ACM symposium on Theory of computing, pp. 151-158. ACM, New York. (1971)

[9] Bodlaender, H.L.: A partial k-arboretum of graphs with bounded treewidth. Theoretical Computer Science, vol. 209, pp. 1-45. (1998)

[10] Bodlaender, H.L.: Treewidth: Algorithmic techniques and results. In: Igor, P., Ružička, P. Mathematical Foundations of Computer Science. LNCS, vol. 1295, pp. 19-36. Springer, Heidelberg. (1997)

[11] Bonsma, P.: The Complexity of Rerouting Shortest Paths. arXiv:1009.3217 (2010)

[12] Cereceda, L., van den Heuvel, J., Johnson, M.: Finding paths between 3-colorings. J. Graph Theory, vol. 67, pp. 69-82. (2011)

[13] Downey, R.G., Fellows, M.R.: Fixed-parameter tractability and completeness II: On completeness for W[1]. Theoretical Computer Science, vol. 141, pp. 109-131. (1995)

[14] Held, M., Karp R.M.: A dynamic programming approach to sequencing problems. Journal of the Society for Industrial & Applied Mathematics, vol. 10, pp. 196-210. (1962)

[15] Hearn, R.A., Demaine E.D.: PSPACE-Completeness of Sliding-Block Puzzles and Other Problems through the Nondeterministic Constraint Logic Model of Computation. Theoretical Computer Science, Game Theory Meets Theoretical Computer Science, Special Issue, vol. 343, pp.72–96. (2005)

[16] Hearn, R.A., Demaine, E.D.: PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. Theoretical Computer Science, vol. 343, pp. 72-96. (2005)

[17] Flake, G.W., Baum, E B.: Rush Hour is PSPACE-complete, or "Why you should generously tip parking lot attendants". Theoretical Computer Science, vol. 270, pp. 895-911. (2005)

[18] Takehiro, I., Kamiński, M., Ono H.: Fixed-parameter tractability of token jumping on planar graphs. In: Ahn, H., Shin. C. Algorithms and Computation LNCS, vol. 8889, pp. 208-219. Springer, Heidelberg. (2014)

[19] Bonamy, M., Bousquet, N.: Reconfiguring independent sets in cographs. arXiv:1406.1433 (2014)