Parameterized Problems Complete for Nondeterministic FPT time and Logarithmic Space

Hans L. Bodlaender, Carla Groenland and Jesper Nederlof
Department of Information and Computing Sciences
Utrecht University
Utrecht, The Netherlands
Email: h.l.bodlaender@uu.nl, c.e.groenland@uu.nl, j.nederlof@uu.nl

Céline M. F. Swennenhuis
Department of Mathematics and Computer Science
Eindhoven University of Technology
Eindhoven, The Netherlands
Email: c.m.f.swennenhuis@tue.nl

***Abstract*—Let XNLP be the class of parameterized problems such that an instance of size n with parameter k can be solved nondeterministically in time $f(k)n^{O(1)}$ and space $f(k)\log(n)$ (for some computable function f). We give a wide variety of XNLP-complete problems, such as LIST COLORING and PRECOLORING EXTENSION with pathwidth as parameter, SCHEDULING OF JOBS WITH PRECEDENCE CONSTRAINTS, with both number of machines and partial order width as parameter, BANDWIDTH and variants of WEIGHTED CNF-SATISFIABILITY and reconfiguration problems. In particular, this implies that all these problems are W[t]-hard for all t. This also answers a long standing question on the parameterized complexity of the BANDWIDTH problem.**

***Keywords*-Parameterized complexity; XNLP; Bandwidth; W-hierarchy**

I. INTRODUCTION

Already since the 1970's, an important paradigm in classical complexity theory has been that an increased number of alternations of existential and universal quantifiers increases the complexity of search problems: This led to the central definition of the polynomial hierarchy [1], whose study resulted in cornerstone results in complexity theory such as Toda's theorem and lower bounds for time/space tradeoffs for SAT [2]. In their foundational work in the early 1990s, Downey and Fellows introduced an analogue of this hierarchy for parameterized complexity theory, called the W-*hierarchy*. This hierarchy comprises of the complexity classes FPT, the parameterized analogue of P, W[1], the parameterized analogue of NP, and the classes W[2], ..., W[P], XP (see e.g. [3]–[5]).

While in the polynomial hierarchy only the classes with no quantifier alternation (i.e. P, NP and co-NP) are prominent, many natural parameterized problems are known to be hard or even complete for W[i] for some $i > 1$. Thus, the W-hierarchy substantially differentiates the complexity of hard parameterized problems. And such a differentiation has applications outside parameterized complexity as well: For example, for problems in W[1] we can typically improve over brute-force enumeration algorithms, while for problems in W[2] we can prove lower bounds under the Strong Exponential Time Hypothesis excluding such improvements (see e.g. the discussion in [6]).[1]

For several problems, completeness for a class is known, e.g., CLIQUE is W[1]-complete [4] and DOMINATING SET is W[2]-complete [3]. However, there are also several problems known to be hard for W[1], W[2], or even for W[t] for all positive integers t, but which are not known to be in the class W[P]; in many cases, only membership in XP was known. For such problems, it is an intriguing question to establish their exact position within the W-hierarchy as it can be expected to shed light on their complexity similarly as it did for the previous problems.

One example of such a problem is the BANDWIDTH problem. It has been known to be hard for all classes W[t] since 1994 [7], with a recently published proof that this already holds for a special subclass of trees (namely, caterpillars with hair length at most three [8]). However, already in the midst of the 1990s, Hallett argued that it is unlikely that BANDWIDTH belongs to W[P], see the discussion by Fellows and Rosamond in [9]. The argument intuitively boils down to the following: BANDWIDTH 'seems' to need certificates with $\Omega(n)$ bits, while problems in W[P] have certificates with $O(f(k)\log n)$ bits. A similar situation applies to several other W[1]-hard problems.

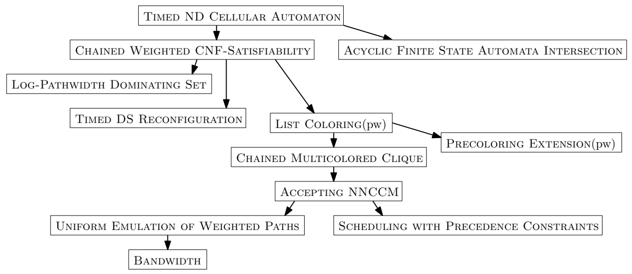
A (largely overlooked) breakthrough was made a few years ago by Elberfeld et al. [10], who studied several classes of parameterized problems, including a class which they called N[f poly, f log]: parameterized problems that can be solved with a non-deterministic algorithm with simultaneously, the running time bounded by $f(k)n^c$ and the space usage bounded by $f(k)\log n$, with k the parameter, n the input size, c a constant, and f a computable function. For easier future reference, we denote this class by XNLP. Elberfeld et al. [10] showed that a number of problems are complete for this class, including the LONGEST COMMON SUBSTRING problem. Since 1995, LONGEST COMMON

[1]For example the naïve algorithm $O(n^{k+1})$ time algorithm for finding cliques on k vertices on n-vertex graphs can be improved to run in $n^{0.8k}$ time, but similar run times for DOMINATING SET refute the Strong Exponential Time Hypothesis.

SUBSTRING is known to be hard for all W$[t]$ [11], but its precise parameterized complexity was unknown until the result by Elberfeld et al. [10].

Our contribution: We show that the class XNLP (i.e., N$[f \operatorname{poly}, f \log]$) can play an important role in establishing the parameterized complexity of a large collection of well studied problems, ranging from abstract problems on different types of automata (see e.g. [10] or later in this paper), logic, graph theory, scheduling, and more. In this paper, we give a number of different examples of problems that are complete for XNLP. These include BANDWIDTH, thus indirectly answering a question that was posed over 25 years ago.

Figure 1 shows for the problems from which problem the reduction starts to show XNLP-hardness.



Timed ND Cellular Automaton
Chained Weighted CNF-Satisfiability
Acyclic Finite State Automata Intersection
Log-Pathwidth Dominating Set
Timed DS Reconfiguration
List Coloring(pw)
Precoloring Extension(pw)
Chained Multicolored Clique
Accepting NNCCM
Uniform Emulation of Weighted Paths
Scheduling with Precedence Constraints
Bandwidth

Figure 1. Reductions between XNLP-hard problems from this paper. Several variants of problems are not shown.

Often, membership in XNLP can be seen by looking at the algorithm that establishes membership in XP. Many problems in XNLP typically have a dynamic programming algorithm that sequentially builds tables, with each individual table entry expressible with $O(f(k) \log n)$ bits. We then get membership in XNLP by instead of tabulating all entries of a table, guessing one entry of the next table — the step resembles the text-book transformation between a deterministic and non-deterministic finite automaton.

Interestingly, hardness for the class XNLP also has consequences for the use of memory of deterministic parameterized algorithms. Pilipczuk and Wrochna [12] conjecture that LONGEST COMMON SUBSEQUENCE (variant 1) has no XP algorithm that runs in $f(k)n^c$ space, for a computable function f and constant c; if this conjecture holds, then no XNLP-hard problem has such an algorithm. See Section V for more details.

When a problem is XNLP-hard, it is also hard for each class W$[t]$ (see Lemma II.2). Thus, XNLP-hardness proofs are also a tool to show hardness for W$[t]$ for all t. In this sense, our results strengthen existing results from the literature: for example, LIST COLORING and PRECOLORING EXTENSION parameterized by pathwidth (or treewidth) were known to be W[1]-hard [13], and PRECEDENCE CONSTRAINT K-PROCESSOR SCHEDULING parameterized by the number of processors K was known to be W[2]-hard [14]. Our XNLP-hardness proofs imply hardness for W$[t]$ for all t. Moreover, our XNLP-hardness proofs are often simpler than the existing proofs that problems are hard for W$[t]$ for all t.

Related to the class XNLP is the class XNL: the parameterized problems that can be solved by a nondeterministic algorithm that uses $f(k) \log n$ space. There is no explicit time bound, but we can freely add a time bound of $2^{f(k) \log n}$, and thus XNL is a subset of XP. XNL can be seen as the parameterized counterpart of NL. Amongst others, XNL was studied by Chen et al. [15], who showed that COMPACT TURING MACHINE COMPUTATION is complete for XNL.

Hardness for a class is always defined with respect to a class of reductions. In our proofs, we use parameterized logspace reductions (or, in short, pl-reductions). A brief discussion of other reductions can be found in Subsection V-B.

Paper overview.: In Section II, we give a number of preliminary definitions and results. In Section III we introduce three new problems that are XNLP-complete. In Section IV we then use these problems as building blocks, to prove other problems to be either XNLP-complete or XNLP hard. For each of the problems, its background and a short literature review specific to it will be given inside its relevant subsection. Final comments and open problems are given in Section V. Proofs have been omitted and can be found in the full version of the paper [16].

II. PRELIMINARIES

In this section we formally define the class XNLP and give some preliminary results.

The section is organized as follows: first we introduce some basic notions in Subsection II-A, next we formally define the class XNLP in Subsection II-B. In Subsection II-C we then introduce the type of reductions that will be used in this paper and in Subsection II-D we go over some preliminary results. Subsection II-E ends the section with a discussion of Cellular automata, for which Elberfeld et al. [10] already established it was XNLP complete. From this problem we will (indirectly) derive the XNLP-hardness for all other XNLP-hard problems in this paper.

A. Basic notions

We assume the reader to be familiar with a number of notions from complexity theory, parameterized algorithms, and graph theory. A few of these are reviewed below, along with some new and less well-known notions.

We use interval notation for sets of integers, i.e., $[a, b] = \{i \in \mathbb{Z} \mid a \leq i \leq b\}$. All logarithms in this paper have base 2. $\mathbf{N}$ denotes the set of the natural numbers $\{0, 1, 2, \ldots\}$, and $\mathbf{Z}^+$ denotes the set of the positive natural numbers $\{1, 2, \ldots\}$.

B. Definition of the class XNLP

In this paper, we study parameterized decision problems, which are subsets of $\Sigma^* \times \mathbf{N}$, for a finite alphabet Σ. The following notation is used, also by e.g. [10], to denote classes of (non-)deterministic parameterized decision problems with a bound on the used time and space. Here, we use the following notations: poly for a polynomial function in the input size; log for $O(\log n)$; n for the input size; f for a computable function of the parameter; ∞ if there is no a priory bound for the resource. Let $\mathrm{D}[t, s]$ denote the class of parameterized decision problems that can be solved by a deterministic algorithm in t time and s space and let $\mathrm{N}[t, s]$ be analogously defined for non-deterministic algorithms. Thus, FPT can be denoted by $\mathrm{D}[f\,\mathrm{poly}, \infty]$; we can denote XP by $\mathrm{D}[n^f, \infty]$, NP by $\mathrm{N}[\mathrm{poly}, \infty]$, L by $\mathrm{D}[\infty, \log]$, etcetera.

A special role in this paper is played by the class $\mathrm{N}[f\,\mathrm{poly}, f\log]$: the parameterized decision problems that can be solved by a non-deterministic algorithm that simultaneously uses at most $f(k)n^c$ time and at most $f(k)\log n$ space, on an input (x, k), where x can be denoted with n bits, f a computable function, and c a constant. Because of the special role of this class, we use the shorter notation XNLP.

XNLP is a subclass of the class XNL, which was studied by Chen et al. [15]. XNL is the class of problems solvable with a non-deterministic algorithm in $f(k)\log n$ space (f, k, n as above), i.e, XNL is the class $\mathrm{N}[\infty, f\log]$.

We assume that the reader to be familiar with notions from parameterized complexity, such as XP, W[1], W[2], …, W[P] (see e.g. [5]). For classes of parameterized problems, we can often make a distinction between non-uniform (a separate algorithm for each parameter value), and uniform. Throughout this paper, we look at the uniform variant of the classes, but we also will assume that functions f of the parameter in time and resource bounds are computable — this is called *strongly uniform* by Downey and Fellows [5].

C. Reductions

Hardness for a class is defined in terms of reductions. We mainly use parameterized logspace reductions, which are a special case of fixed parameter tractable reductions. Both are defined below; the definitions are based upon the formulations in [10]. Two other types of reductions are briefly discussed in the Conclusion (Section V-B.)

- A *parameterized reduction* from a parameterized problem $Q_1 \subseteq \Sigma_1^* \times \mathbf{N}$ to a parameterized problem $Q_2 \subseteq \Sigma_2^* \times \mathbf{N}$ is a function $f : \Sigma_1^* \times \mathbf{N} \to \Sigma_2^* \times \mathbf{N}$, such that the following holds.
    1) For all $(x, k) \in \Sigma_1^* \times \mathbf{N}$, $(x, k) \in Q_1$ if and only if $f((x, k)) \in Q_2$.
    2) There is a computable function g, such that for all $(x, k) \in \Sigma_1^* \times \mathbf{N}$, if $f((x, k)) = (y, k')$, then $k' \le g(k)$.
- A *parameterized logspace reduction* or *pl-reduction* is a parameterized reduction for which there is an algorithm that computes $f((x, k))$ in space $O(g(k) + \log n)$, with g a computable function and $n = |x|$ the number of bits to denote x.
- A *fixed parameter tractable reduction* or *fpt-reduction* is a parameterized reduction for which there is an algorithm that computes $f((x, k))$ in time $O(g(k)n^c)$, with g a computable function, $n = |x|$ the number of bits to denote x and c a constant.

In the remainder of the paper, unless stated otherwise, completeness for XNLP is with respect to pl-reductions.

D. Preliminary results on XNLP

We give some easy observations that relate XNLP to other notions from parameterized complexity. The following easy observation can be seen as a special case of the fact that $\mathrm{N}[\infty, S(n)] \subseteq \mathrm{D}[2^{S(n)}, \infty]$, see [2, Theorem 4.3].

Lemma II.1. XNLP *is a subset of* XP.

The idea is to transform the non-deterministic algorithm to a deterministic algorithm that employs dynamic programming: tabulate all reachable configurations of the machine (a configuration is a tuple, consisting of the contents of the work tape, the state of the machine, and the position of the two headers). From a configuration, we can compute all configurations that can be reached in one step, and thus we can check if a configuration that has an accepting state can be reached.

Lemma II.2. *If a parameterized problem Q is* XNLP*-hard, then it is hard for each class* $\mathrm{W}[t]$ *for all* $t \in \mathbf{Z}^+$.

For this, we show that WEIGHTED t-NORMALIZED SATISFIABILITY (a $\mathrm{W}[t]$-complete problem) is in XNLP. We have a Boolean formula with parenthesis-depth t and ask if we can satisfy it by setting exactly k variables to true and all others to false. We can non-deterministically guess which of the k Boolean variables are true; verifying whether this setting satisfies the formula can be done with $O(t + k\log n)$ bits of space, see e.g. [5].

Lemma II.3 (Chen et al. [15])**.** *If* $\mathrm{NL} \ne \mathrm{P}$*, then there are parameterized problems in* FPT *that do not belong to* XNL *(and hence also not to* XNLP*).*

Chen et al. [15] introduce the following problem.

CNTMC
Input: the encoding of a non-deterministic Turing Machine M; the encoding of a string x over the alphabet of the machine.
Parameter: k.
Question: Is there an accepting computation of M on input x that visits at most k cells of the work tape?

Theorem II.4 (Chen et al. [15])**.** CNTMC *is* XNL-*complete under pl-reductions.*

It is possible to show XNLP-completeness for a 'timed' variant of this problem.

TIMED CNTMC
Input: the encoding of a non-deterministic Turing Machine M; the encoding of a string x over the alphabet of the machine; an integer T given in unary.
Parameter: k.
Question: Is there an accepting computation of M on input x that visits at most k cells of the work tape and uses at most T time?

The fact that the time that the machine uses is given in unary, is needed to show membership in XNLP.

Theorem II.5. TIMED CNTMC *is* XNLP-*complete.*

We state the result without proof, as the proof is similar to the proof of Theorem II.4 from [15], and we do not build upon the result. We instead start with a problem on cellular automata which was shown to be complete for XNLP by Elberfeld et al. [10]. We discuss this problem in the next subsection. Elberfeld et al. [10] show a number of other problems to be XNLP-complete, including a timed version of the acceptance of multihead automata, and the LONGEST COMMON SUBSEQUENCE problem, parameterized by the number of strings. The latter result is discussed in the Conclusion, Section V-A.

E. Cellular automata

In this subsection, we discuss one of the results by Elberfeld et al. [10]. Amongst the problems that are shown to be complete for XNLP by Elberfeld et al. [10], of central importance to us is the TIMED NON-DETERMINISTIC CELLULAR AUTOMATON problem. We use the hardness of this problem to show the hardness of CHAINED CNF-SATISFIABILITY in Subsection III-A.

In this subsection, we describe the TIMED NON-DETERMINISTIC CELLULAR AUTOMATON problem, and a variant. We are given a linear cellular automaton, a time bound t given in unary, and a starting configuration for the automaton, and ask if after t time steps, at least one cell is in an accepting state.

More precisely, we have a set of states S. We assume there are two special states s_L and s_R which are used for the leftmost and rightmost cell. A *configuration* is a function $c : \{1, \ldots, q\} \to S$, with $c(1) = s_L$, $c(q) = s_R$ and for $i \in [2, q-1]$, $c(i) \in S \setminus \{s_L, s_R\}$. We say that we have q cells, and in configuration c, cell i has state $c(i)$. The machine is further described by a collection of 4-tuples $\mathcal{T}$ in $S \times (S \setminus \{s_L, s_R\}) \times S \times (S \setminus \{s_L, s_R\})$. At each time step, each cell $i \in [2, q]$ reads the 3-tuple (s_1, s_2, s_3) of states given by the current states of the cells $i-1,\ i$ and $i+1$ (in that order). If there is no 4-tuple of the form (s_1, s_2, s_3, s_4) for some $s_4 \in S$, then the machine halts and rejects; otherwise, the cell selects an $s_4 \in S$ with $(s_1, s_2, s_3, s_4) \in \mathcal{T}$ and moves in this time step to state s_4. (In a non-deterministic machine, there can be multiple such states s_4 and a non-deterministic step is done. For a deterministic cellular automaton, for each 3-tuple (s_1, s_2, s_3) there is at most one 4-tuple $(s_1, s_2, s_3, s_4) \in \mathcal{T}$.) Note that the leftmost and rightmost cell never change state: their states are used to mark the ends of the tape of the automaton.

TIMED NON-DETERMINISTIC CELLULAR AUTOMATON
Input: Cellular automaton with set of states S and set of transitions $\mathcal{T}$; configuration c on q cells; integer in unary t; subset $A \subseteq S$ of accepting states.
Parameter: q.
Question: Is there an execution of the machine for exactly t steps with initial configuration c, such that at time t at least one cell of the automaton is in A?

We will build on the following result.

Theorem II.6 (Elberfeld et al. [10])**.** TIMED NON-DETERMINISTIC CELLULAR AUTOMATON *is* XNLP-*complete.*

We recall that the class, denoted by XNLP in the current paper, is called N[f poly, f log] in [10].

Elberfeld et al. [10] state that asking that all cells are in an accepting state does not make a difference, i.e., if we modify the TIMED NON-DETERMINISTIC CELLULAR AUTOMATON problem by asking if all cells are in an accepting state at time t, then we also have an XNLP-complete problem.

We also discuss a variant that can possibly be useful as another starting point for reductions.

TIMED NON-HALTING NON-DETERMINISTIC CELLULAR AUTOMATON
Input: Cellular automaton with set of states S and set of transitions $\mathcal{T}$; configuration c on q cells; integer in unary t; subset $A \subseteq S$ of accepting states.
Parameter: q.
Question: Is there an execution of the machine for exactly t steps with initial configuration c, such that the machine does not halt before time t?

Corollary II.7. TIMED NON-HALTING NON-DETERMINISTIC CELLULAR AUTOMATON *is* XNLP-*complete.*

III. Building Blocks

In this section, we introduce three new problems and prove that they are XNLP-complete, namely Chained CNF-Satisfiability, Chained Multicolored Clique and Accepting NNCCM. These problems are called building blocks, as their main use is proving XNLP-hardness for many other problems (see Figure 1).

A. Chained CNF-Satisfiability

In this subsection we give a useful starting point for our transformations: a variation of Satisfiability which we call Chained Weighted CNF-Satisfiability. The problem can be seen as a generalization of the W[1]-hard problem Weighted CNF-Satisfiability [3].

Chained Weighted CNF-Satisfiability
Input: r disjoint sets of Boolean variables $X_1, X_2, \ldots X_r$, each of size q; integer $k \in \mathbf{N}$; Boolean formulas F_1, F_2, $\ldots$, F_{r-1}, where each F_i is an expression in conjunctive normal form on variables $X_i \cup X_{i+1}$.
Parameter: k.
Question: Is it possible to satisfy the formula $F_1 \wedge F_2 \wedge \cdots \wedge F_r$ by setting exactly k variables to true from each set X_i and all others to false?

Theorem III.1. Chained Weighted CNF-Satisfiability *is* XNLP-*complete.*

In the proof, $F_2 = F_3 = \cdots = F_{r-2}$, and more specifically, we have a condition on X_1, a condition on X_r, and identical conditions on all pairs $X_i \cup X_{i+1}$ with i from 1 to $r-1$. Thus, we also have XNLP-completeness of the following special case:

Regular Chained Weighted CNF-Satisfiability
Input: r sets of Boolean variables $X_1, X_2, \ldots X_r$, each of size q; an integer $k \in \mathbf{N}$; Boolean formulas F_0, F_1, F_2 in conjunctive normal form, where F_0 and F_2 are expressions on q variables, and F_1 is an expression on $2q$ variables.
Parameter: k.
Question: Is it possible to satisfy the formula

$$F_0(X_1) \wedge \bigwedge_{1 \le i \le r-1} F_1(X_i, X_{i+1}) \wedge F_2(X_r)$$

by setting exactly k variables to true from each set X_i and all others to false?

Moreover, we can also reduce to the case where there is only one set of constraints (by 'adding the constraints from F_0 and F_2 to F_1 and ensuring that they are only verified at the start and at the end of the chain'). Another simplification that we can make is as follows. Each set of variables X_i can be partitioned into k subsets, and a solution has exactly one true variable for each subset, e.g., for each t', exactly one $x_{t',r,s}$ is true in the proof of Theorem III.1. This still holds after the modification in the proofs of the later results. We define the following variant.

Partitioned Regular Chained Weighted CNF-Satisfiability
Input: r sets of Boolean variables $X_1, X_2, \ldots X_r$, each of size q; an integer $k \in \mathbf{N}$; Boolean formula F_1, which is in conjunctive normal form and an expression on $2q$ variables; for each i, a partition of X_i into $X_{i,1}, \ldots, X_{i,k}$ with for all i_1, i_2, j: $|X_{i_1,j}| = |X_{i_2,j}|$.
Parameter: k.
Question: Is it possible to satisfy the formula

$$\bigwedge_{1 \le i \le r-1} F_1(X_i, X_{i+1})$$

by setting from each set $X_{i,j}$ exactly 1 variable to true and all others to false?

For all these problems, there are also the special cases in which all literals that appear in the formulas F_i are positive, i.e., we have no negations. We call these special case Chained Weighted Positive CNF-Satisfiability, Regular Chained Weighted Positive CNF-Satisfiability and Partitioned Regular Chained Weighted Positive CNF-Satisfiability respectively. All problems above are shown to be XNLP-complete in the full version of this paper [16].

B. *Chained Multicolored Clique*

The Multicolored Clique problem is an important tool to prove fixed parameter intractability of various parameterized problems. It was independently introduced by Pietrzak [17] (under the name Partitioned Clique) and by Fellows et al. [18].

In this paper, we introduce a chained variant of Multicolored Clique. In this variant, we ask to find a sequence of cliques, that are overlapping with the previous and next clique in the chain.

Chained Multicolored Clique
Input: Graph $G = (V, E)$; partition of V into sets $V_1, \ldots, V_r$, such that for each edge $\{v, w\} \in E$, if $v \in V_i$ and $w \in V_j$, then $|i - j| \le 1$; function $f : V \to \{1, 2, \ldots, k\}$.
Parameter: k.
Question: Is there a subset $W \subseteq V$ such that for each $i \in [1, r]$, $W \cap (V_i \cup V_{i+1})$ is a clique, and for each $i \in [1, r]$ and each $j \in [1, k]$, there is a vertex $w \in V_i \cap W$ with $f(w) = j$?

Thus, we have a clique with $2k$ vertices in $V_i \cup V_{i+1}$ for each $i \in [1, r-1]$, with for each color a vertex with that color in V_i and a vertex with that color in V_{i+1}. Importantly, the same vertices in V_i are chosen in the clique for $V_{i-1} \cup V_i$

as for $V_i \cup V_{i+1}$ for each $i \in [2, r-1]$. Below, we call such a set a *chained multicolored clique*.

Theorem III.2. CHAINED MULTICOLORED CLIQUE *is* XNLP*-complete.*

A simple variation is the following. We are given a graph $G = (V, E)$, a partition of V into sets $V_1, \ldots, V_r$ with the property that for each edge $\{v, w\} \in E$, if $v \in V_i$ and $w \in V_j$ then $|i - j| \leq 1$, and a coloring function $f : V \to \{1, 2, \ldots, k\}$. A *chained multicolored independent set* is an independent set S with the property that for each $i \in [1, r]$ and each color $j \in [1, k]$, the set S contains exactly one vertex $v \in V_i$ of color $f(v) = j$. The CHAINED MULTICOLORED INDEPENDENT SET problem asks for the existence of such a chained multicolored independent set, with the number of colors k as parameter. We have the following simple corollary.

Corollary III.3. CHAINED MULTICOLORED INDEPENDENT SET *is* XNLP*-complete.*

C. Non-decreasing counter machines

In this subsection, we introduce a new simple machine model, which can also capture the computational power of XNLP (see Theorem III.4). This model will be a useful stepping stone when proving XNLP-hardness reductions in Section IV.

A *Nondeterministic Nondecreasing Checking Counter Machine* (or: NNCCM) is described by a 3-tuple (k, n, s), with k and n positive integers, and $s = (s_1, \ldots, s_r)$ a sequence of 4-tuples (called *checks*). For each $i \in \{1, \ldots, r\}$, the 4-tuple s_i is of the form (c_1, c_2, r_1, r_2) with $c_1, c_2 \in \{1, 2, \ldots, k\}$ positive integers and $r_1, r_2 \in \{0, 1, 2, \ldots, n\}$ non-negative integers. These model the indices of the counters and their values respectively.

An NNCCM (k, n, s) with $s = (s_1, \ldots, s_r)$ works as follows. The machine has k counters that are initially 0. For i from 1 to r, the machine first sets each of the counters to any integer that is at least its current value and at most n. After this, the machine performs the ith check $s_i = (c_1, c_2, r_1, r_2)$: if the value of the c_1th counter equals r_1 and the value of the c_2th counter equals r_2, then we say the ith check rejects and the machine halts and rejects. When the machine has not rejected after all r checks, the machine accepts.

The nondeterministic steps can be also described as follows. Denote the value of the cth counter when the ith check is done by $c(i)$. We define $c(0) = 0$. For each $i \in \{1, \ldots, r\}$, $c(i)$ is an integer that is nondeterministically chosen from $[c(i-1), n]$.

We consider the following computational problem.

ACCEPTING NNCCM
Given: An NNCCM (k, n, s) with all integers given in unary.
Parameter: The number of counters k.
Question: Does the machine accept?

Theorem III.4. ACCEPTING NNCCM *is* XNLP*-complete.*

The ACCEPTING NNCCM problem appears to be a very useful tool for giving XNLP-hardness proofs. Note that one step where the k counters can be increased to values at most n can be replaced by kn steps where counters can be increased by one, or possibly a larger number, again to at most n. This modification is used in some of the proofs in the following section.

IV. APPLICATIONS

In this section, we consider several problem, which we prove to be XNLP-complete.

A. List Coloring Parameterized by Pathwidth

LIST COLORING and PRECOLORING EXTENSION problems parameterized by treewidth or pathwidth belong to XP [19]. Fellows et al. [13] have shown that LIST COLORING and PRECOLORING EXTENSION parameterized by treewidth are W[1]-hard. (The transformation in their paper also works for parameterization by pathwidth.) We strengthen this result by showing that LIST COLORING and PRECOLORING EXTENSION parameterized by pathwidth are XNLP-complete. Note that this result also implies W[t]-hardness of the problems for all integers t.

Given a graph $G = (V, E)$ with lists L_v for vertex $v \in V(G)$, a *list coloring* for G is a choice of color $f(v) \in L_v$ for each vertex v such that $f(v) \neq f(w)$ when $\{v, w\} \in E$.

A *path decomposition* of a graph $G = (V, E)$ is a sequence $(X_1, X_2, \ldots, X_r)$ of subsets of V with the following properties.

1) $\bigcup_{1 \leq i \leq r} X_i = V$.
2) For all $\{v, w\} \in E$, there is an $i \in I$ with $v, w \in X_i$.
3) For all $1 \leq i_0 < i_1 < i_2 \leq r$, $X_{i_0} \cap X_{i_2} \subseteq X_{i_1}$.

The *width* of a path decomposition $(X_1, X_2, \ldots, X_r)$ equals $\max_{1 \leq i \leq r} |X_i| - 1$, and the *pathwidth* of a graph G is the minimum width of a path decomposition of G.

Theorem IV.1. LIST COLORING *and* PRECOLORING EXTENSION *parameterized by pathwidth are* XNLP*-complete.*

We assume that a path decomposition of the input graph G is given as part of the input. We conjecture that such a path decomposition can be found with a non-deterministic algorithm using logarithmic space and 'fpt' time, but the details need a careful study. Elberfeld et al. [20] show that for each fixed k, determining if the treewidth is at most k, and if so, finding a tree decomposition of width at most k belongs to L.

From Theorem IV.1, we can also directly conclude that LIST COLORING and PRECOLORING EXTENSION are XNLP-hard when parameterized by the treewidth. However, we leave membership of these problems when parameterized by treewidth as an open problem.

B. Logarithmic Pathwidth

There are several well known problems that can be solved in time $O(c^k n)$ for a constant c on graphs of pathwidth or treewidth at most k. Classic examples are INDEPENDENT SET and DOMINATING SET (see e.g., [21, Chapter 7.3]), but there are many others, e.g., [22], [23].

In this subsection, rather than bounding the pathwidth by a constant, we allow the pathwidth to be linear in the logarithm of the number of vertices of the graph. We consider the following problem.

LOG-PATHWIDTH DOMINATING SET
Input: Graph $G = (V, E)$, path decomposition of G of width ℓ, integer K.
Parameter: $\lceil \ell / \log |V| \rceil$.
Question: Does G have a dominating set of size at most K?

The independent set variant and clique variants are defined analogously.

Theorem IV.2. LOG-PATHWIDTH DOMINATING SET *and* LOG-PATHWIDTH INDEPENDENT SET *are* XNLP-*complete.*

We now briefly discuss the LOG-PATHWIDTH CLIQUE problem. We are given a graph $G = (V, E)$ with a path decomposition of width ℓ, and integer K and ask if there is a clique in G with at least K vertices. Let $k = \lceil \ell / \log |V| \rceil$.

The problem appears to be significantly easier than the corresponding versions of DOMINATING SET and INDEPENDENT SET, mainly because of the property that for each clique W, there must be a bag of the path decomposition that contains all vertices of W (see e.g., [24].) Thus, the problem reverts to solving $O(n)$ instances of CLIQUE on graphs with $O(k \log n)$ vertices. This problem is related to a problem called MINI-CLIQUE where the input has a graph with a description size that is at most $k \log n$. MINI-CLIQUE is M[1]-complete under FPT Turing reductions (see [25, Corollary 29.5.1]), and is a subproblem of our problem, and thus LOG-PATHWIDTH CLIQUE is M[1]-hard. However, instances of LOG-PATHWIDTH CLIQUE can have description sizes of $\Omega(\log^2 n)$. The result below shows that it is unlikely that LOG-PATHWIDTH CLIQUE is XNLP-hard.

Proposition IV.3. LOG-PATHWIDTH CLIQUE *is in* $W[2]$.

C. Scheduling with precedence constraints

In 1995, Bodlaender and Fellows [14] showed that the problem to schedule a number of jobs of unit length with precedence constraints on K machines, minimizing the makespan, is W[2]-hard, with the number of machines as parameter. A closer inspection of their proof shows that W[2]-hardness also applies when we take the number of machines and the width of the partial order as combined parameter. In this subsection, we strengthen this result, showing that the problem is XNLP-complete (and thus also hard for all classes W[t], $t \in \mathbf{Z}^+$.) In the notation used in scheduling literature to characterize scheduling problems, the problem is known as $P|prec, p_j = 1|C_{\max}$, or, equivalently, $P|prec, p_j = p|C_{\max}$.

SCHEDULING WITH PRECEDENCE CONSTRAINTS
Given: K, D positive integers; T set of tasks; $\prec$ partial order on T of width w.
Parameter: $K + w$.
Question: Is there a schedule $f : T \to [1, D]$ with $|f^{-1}\{i\}| \leq K$ for all $i \in [1, D]$ such that $t \prec t'$ implies $f(t) < f(t')$.

In other words, we parametrise $P|prec, p_j = 1|C_{max}$ by the number of machines and the width of the partial order.

Theorem IV.4. SCHEDULING WITH PRECEDENCE CONSTRAINTS *is* XNLP-*complete.*

We remark that it is unclear if the problem is in XNLP when we take only the number of machines as parameter. In fact, it is a longstanding open problem whether SCHEDULING WITH PRECEDENCE CONSTRAINTS is NP-hard when there are three machines (see e.g., [26], [27]).

D. Uniform Emulation of Weighted Paths

In this subsection, we give a proof that the UNIFORM EMULATION OF WEIGHTED PATHS problem is XNLP-complete. The result is a stepping stone for the result that BANDWIDTH is XNLP-complete even for caterpillars with hair length at most three (see the discussion in Subsection IV-E).

The notion of *(uniform) emulation* of graphs on graphs was originally introduced by Fishburn and Finkel [28] as a model for the simulation of computer networks on smaller computer networks. Bodlaender [29] studied the complexity of determining for a given graph G and path P_m if there is a uniform emulation of G on P_m. In this subsection, we study a weighted variant, and show that already determining whether there is a uniform emulation of a weighted path on a path is hard.

An *emulation* of a graph $G = (V, E)$ on a graph $H = (W, F)$ is a mapping $f : V \to W$ such that for all edges $\{v, w\} \in E$, $f(v) = f(w)$ or $\{f(v), f(w)\} \in F$. We say that an emulation is *uniform* if there is an integer c, such that $|f^{-1}\{w\}| = |\{v \mid f(v) = w\}| = c$ for all $w \in W$. We call c the emulation factor.

Determining whether there is a uniform emulation of a graph H on a path P_m is NP-complete, even for emulation factor 2, if we allow H to be disconnected. The problem to determine for a given *connected* graph H if there is a uniform emulation of H on a path P_m belongs to XP with the emulation factor c as parameter [29].

Recently, Bodlaender [8] looked at the weighted variant of uniform emulation on paths, for the case that H is a path. Now, we have a path P_n and a path P_m, a weight function $w : [1, n] \to \mathbf{Z}^+$ and ask for an emulation $f : [1, n] \to [1, m]$, such that there is a constant c with $\sum_{i \in f^{-1}\{j\}} w(i) = c$ for all $j \in [1, m]$. Again, we call c the emulation factor.

It is not hard to see that the problem, given n, m, and weight function $w : [1, n] \to \mathbf{Z}^+$, to determine if there is a uniform emulation of P_n on P_m with emulation factor c is in XP, with the emulation factor c as parameter; the dynamic programming algorithm from [29] can easily be adapted.

As an intermediate step for a hardness proof for BANDWIDTH, Bodlaender [8] showed that UNIFORM EMULATION OF WEIGHTED PATHS is hard for all classes W[t], $t \in \mathbf{Z}^+$. In the current subsection, we give a stronger result, and show the same problem to be XNLP-complete. Our proof is actually simpler than the proof in [8] — by using ACCEPTING NNCCM as starting problem, we avoid a number of technicalities.

UNIFORM EMULATION OF WEIGHTED PATHS
Input: Positive integers n, m, c, weight function $w : [1, n] \to [1, c]$.
Parameter: c.
Question: Is there a function $f : [1, n] \to [1, m]$, such that f is a uniform emulation of P_n on P_m with emulation factor c, i.e., $|f(i) - f(i + 1)| \leq 1$ for all $i \in [1, n - 1]$ and $\sum_{i \in f^{-1}\{j\}} w(i) = c$ for all $j \in [1, m]$?

Theorem IV.5. UNIFORM EMULATION OF WEIGHTED PATHS *is* XNLP-*complete.*

E. Bandwidth

In this subsection, we discuss the XNLP-completeness problem of the BANDWIDTH problem. The question where the parameterized complexity of BANDWIDTH lies was actually the starting point for the investigations whose outcome is reported in this paper; with the main result of this subsection (Corollary IV.7) we answer a question that was asked over a quarter of a century ago.

In the BANDWIDTH problem, we are given a graph $G = (V, E)$ and an integer k and ask if there is a bijection $f : V \to [1, |V|]$ such that for all $\{v, w\} \in E$: $|f(v) - f(w)| \leq k$. The problem models the question to permute rows and columns of a symmetric matrix, such that all non-zero entries are at a small 'band' along the main diagonal. Already in 1976, the problem was shown to be NP-complete by Papadimitriou [30]. Later, several special cases were shown to be hard; these include caterpillars with hairs of length at most three [31]. A *caterpillar* is a tree where all vertices of degree at least three are on a common path; the *hairs* are the paths attached to this main path.

We are interested in the parameterized variant of the problem, where the target bandwidth is the parameter:

BANDWIDTH
Given: Integer k, undirected graph $G = (V, E)$
Parameter: k
Question: Is there a bijection $f : V \to [1, |V|]$ such that for all edges $\{v, w\} \in E$: $|f(v) - f(w)| \leq k$?

BANDWIDTH belongs to XP. In 1980, Saxe [32] showed that BANDWIDTH can be solved in $O(n^{k+1})$ time; this was later improved to $O(n^k)$ by Gurari and Sudborough [33]. In 1994, Bodlaender and et. [7] reported that BANDWIDTH for trees is W[t]-hard for all $t \in \mathbf{Z}^+$ — the proof of that fact was published 26 years later [8]. A sketch of the proof appears in the monograph by Downey and Fellows [5]. More recently, Dregi and Lokshtanov [34] showed that BANDWIDTH is W[1]-hard for trees of pathwidth at most two. In addition, they showed that there is no algorithm for BANDWIDTH on trees of pathwidth at most two with running time of the form $f(k)n^{o(k)}$ assuming the Exponential Time Hypothesis. Recently, Bodlaender [8] published a proof that BANDWIDTH is W[t]-hard for all t, even for caterpillars with hairs of length at most three. That result is obtained by first showing that UNIFORM EMULATION OF WEIGHTED PATHS is W[t] hard for all t, and then giving a transformation from that problem to BANDWIDTH for caterpillars with maximum hair length tree. The latter transformation uses gadgets from the NP-completeness proof by Monien [31].

Lemma IV.6. BANDWIDTH *is in* XNLP.

One proof of this is via the XP algorithms for BANDWIDTH from [32] or [33], observing that when instead of making full tables in the dynamic programming algorithm, we guess from each table one entry, one obtains an algorithm in XNLP.

We now obtain our main result as a corollary of Lemma IV.6 and a reduction from UNIFORM EMULATION OF WEIGHTED PATHS with the exact same transformation as given in the proof of Theorem 4.1 in [8] (which is in fact a pl-reduction).

Corollary IV.7. BANDWIDTH *for caterpillars with hairs of length at most three is* XNLP-*complete.*

F. Timed Reconfiguration

We now consider yet another very different setting: reconfiguration.

Given a graph G and dominating sets S and S', we wish to know whether we can go from one dominating set to the other via a sequence of dominating sets. All dominating sets are of the same size k (which is our parameter) and can be visualised by placing k 'tokens' on the vertices of the graph. The following two rules that are commonly considered for when dominating sets S_1 and S_2 are adjacent (that is, can be consecutive in the sequence).

- Token Jumping (TJ). We may 'jump' a single token, that is, the dominating sets S_1 and S_2 of size k are

adjacent if $S_1 = S_2 \setminus \{u\} \cup \{v\}$ for some $u, v \in V(G)$. This rule is equivalent[2] to Token Addition and Removal (TAR), in which a token can be either added or removed, as long as the total number of tokens does not go above $k+1$.
- Token Sliding (TS). We may 'slide' a single token along an edge, that is, the dominating sets S_1 and S_2 of size k are adjacent if $S_1 = S_2 \setminus \{u\} \cup \{v\}$ for some $uv \in E(G)$.

The TJ/TAR rule has been most widely studied, and the problem is known to be PSPACE-complete (even for simple graph classes such as planar graphs and classes of bounded bandwidth) [35] and W[2]-hard with parameter $k+\ell$, for k the number of tokens and ℓ the length of the reconfiguration sequence [36]. Under TS, the problem is also known to be PSPACE-complete even for various restricted graph classes [37].

It turns out that these problems are XNLP-complete when we bound the number of steps in the reconfiguration sequences as follows.

TIMED TS-DOMINATING SET RECONFIGURATION
Given: Graph $G = (V, E)$; dominating sets S, S' of size k; integer T given in unary.
Parameter: k.
Question: Does there exist a sequence $S = S_1, S_2, \ldots, S_T = S'$ of dominating sets of size k, with for all $i \in [2, T]$, $S_i = S_{i-1} \setminus \{u\} \cup \{v\}$ for some $uv \in E(G)$?

TIMED TJ-DOMINATING SET RECONFIGURATION
Given: Graph $G = (V, E)$; dominating sets S, S' of size k; integer T given in unary.
Parameter: k.
Question: Does there exist a sequence $S = S_1, S_2, \ldots, S_T = S'$ of dominating sets of size k, with for all $i \in [2, T]$, $S_i = S_{i-1} \setminus \{u\} \cup \{v\}$ for some $u, v \in V(G)$?

Variants for independent set and clique instead of dominating set are defined analogously.

Independent set reconfiguration has been widely studied and is known to be PSPACE-complete for both TJ [38] and TS [39], and W[1]-hard for TJ when parameterized by the number of tokens [40].

Theorem IV.8. *TIMED TS-DOMINATING SET RECONFIGURATION, TIMED TJ-DOMINATING SET RECONFIGURATION, TIMED TS-CLIQUE RECONFIGURATION, TIMED TJ-CLIQUE RECONFIGURATION, TIMED TS-INDEPENDENT SET RECONFIGURATION and TIMED TJ-INDEPENDENT SET RECONFIGURATION are XNLP-complete.*

[2]Formally, there is a reconfiguration sequence for TJ with k tokens between S_1 and S_2 if and only if there is one for TAR with upper bound $k+1$.

We remark that token jumping with k tokens is equivalent to the token addition-removal rule where the solution always needs to contain at least $k-1$ vertices. On the other hand, token sliding and token jumping are not equivalent. If complements cannot be taken efficiently, specific graph classes are studied or the token sliding rule is used, the clique reconfiguration and the independent set reconfiguration could have different complexities. Many other solution concepts have also been studied for reconfiguration, such as satisfiability and coloring versions, and we refer the reader to the survey [41] for more information.

G. Acyclic Finite State Automata Intersection

In the FINITE STATE AUTOMATA INTERSECTION problem, we are given k deterministic finite state automata on an alphabet Σ and ask if there is a string $s \in \Sigma^*$ that is accepted by each of the automata.

In the overview of the parameterized complexity of various problems in [5], the problem is mentioned to be hard for all classes W$[t]$, $t \in \mathbf{Z}^+$, when either parameterized by the number of machines k or by the combination of the number of machines k and the size of the alphabet Σ; the result is due to Hallett, but has not been published.

More recently,the problem and many variations were studied by Wehar [42]. Amongst others, he showed that FINITE STATE AUTOMATA INTERSECTION with the number of machines as parameter is XNL-complete. He also considered the variant where the automata are acyclic.

ACYCLIC FINITE STATE AUTOMATA INTERSECTION
Input: k deterministic finite state automata on an alphabet Σ for which the underlying graphs are acyclic (except for self-loops at an accepting or rejecting state).
Parameter: k.
Question: Is there a string $s \in \Sigma^*$ that is accepted by each of the automata?

Wehar [42, Chapter 5] showed that ACYCLIC FINITE STATE AUTOMATA INTERSECTION is equivalent under LBL-reductions (parameterized reductions that do not change the parameter) to a version of TIMED CNTMC (see Section II-D) where the given time bound T is linear. The proof technique of Wehar [42] can also be used to show that ACYCLIC FINITE STATE AUTOMATA INTERSECTION is XNLP-complete. We use a different, simple reduction from LONGEST COMMON SUBSEQUENCE.

Theorem IV.9. *ACYCLIC FINITE STATE AUTOMATA INTERSECTION is XNLP-complete.*

In fact, the problem remains XNLP-complete when restricted to a binary alphabet.

V. Conclusion

We end the paper with some discussions and open problems. We start by discussing a conjecture on the space usage of XNLP-hard problems, then discuss the type of reductions we use, and then give a number of open problems.

A. *Space efficiency of* XNLP*-hard problems*

Pilipczuk and Wrochna [12] made the following conjecture. In the Longest Common Subsequence problem, we are given k strings $s^1, \ldots, s^k$ over an alphabet Σ and an integer r and ask if there is a string t of length r that is a subsequence of each s^i, $i \in [1,k]$.

Conjecture V.1 (Pilipczuk and Wrochna [12])**.** *The* Longest Common Subsequence *problem has no algorithm that runs in $n^{f(k)}$ time and $f(k)n^c$ space, for a computable function f and constant c, with k the number of strings, and n the total input size.*

Interestingly, this conjecture leads to similar conjectures for a large collection of problems. As Longest Commom Subsequence with the number of strings k as parameter is XNLP-complete [10], Conjecture V.1 is equivalent to the following conjecture. Recall that the class XNLP is the same as the class $\mathrm{N}[f\,\mathrm{poly}, f\log]$.

Conjecture V.2. $\mathrm{N}[f\,\mathrm{poly}, f\log] \not\subseteq \mathrm{D}[n^f, f\,\mathrm{poly}]$.

If Conjecture V.1 holds, then no XNLP-hard problem has an algorithm that uses XP time and simultaneously 'FPT' space (i.e., space bounded by the product of a computable function of the parameter and a polynomial of the input size). Thus, XNLP-hardness proofs yield conjectures about the space usage of XP algorithms, and Conjecture V.1 is equivalent to the same conjecture for Bandwidth, List Coloring parameterized by pathwidth, Chained CNF-Satisfiability, etc.

B. *Reductions*

In this paper, we mainly used parameterized logspace reductions (pl-reductions), i.e., parameterized reductions that run in $f(k)+O(\log n)$ space, with f a computable function.

Elberfeld et al [10] use a stronger form of reductions, namely *parameterized first-order reductions* or pFO-reductions, where the reduction can be computed by a logarithmic time-uniform para AC^O-circuit family. In [10], it is shown that Timed Non-Deterministic Cellular Automaton and Longest Common Subsequence (with the number of strings as parameter) are XNLP-complete under pFO-reductions. We have chosen to use the easier to handle notion of logspace reductions throughout the paper, and not to distinguish which steps can be done with pl-reductions and which not.

One might want to use the least restricted form of reductions, under which XNLP remains closed, and that are transitive, in order to be able to show hardness for XNLP for as many problems as possible. Instead of using $O(f(k)+\log n)$ space, one may want to use $O(f(k)\cdot\log n)$ space — thus allowing to use a number of counters and pointers that depends on the parameter, instead of being bounded by a fixed constant. However, it is not clear that XNLP is closed under parameterized reductions with a $O(f(k)\cdot\log n)$ space bound, as the reduction may use $O(n^{f(k)})$ time.

To remedy this, we can simultaneously bound the time and space of the reduction. A *parameterized tractable logspace reduction* (ptl-reductions) is a parameterized reduction that simultenously uses $O(f(k)\cdot\log n)$ space and $O(g(k)\cdot n^c)$ time, with f and g computable functions, k the parameter, and n the input size. One can observe that the same argument ('repeatedly recomputing input bits when needed') that shows transitivity of L-reductions (see [2, Lemma 4.15]) can be used to show transitivity of parameterized tractable logspace reductions (and of parameterized logspace reductions).

We currently are unaware of a problem where we would use ptl-reductions instead of pl-reductions. However, the situation reminds of a phenomenon that also shows up for hardness proofs for classes in the W-hierarchy. Pl-reductions allow us to use time that grows faster than polynomial in the parameter value. If we have an fpt-reduction that uses $O(f(k)n^c)$ time with c a constant, and f *a polynomial function*, then this reduction is also a many-to-one reduction, and could be used in an NP-hardness proof for the unparameterized version of the problem. Most *but not all* fpt-reductions from the literature have such a polynomial time bound. However, in the published hardness proofs, the distinction is usually not made explicit.

C. *Candidate* XNLP*-complete problems*

In this paper, we introduced a new parameterized complexity class, and showed a number of parameterized problems complete for the class, including Bandwidth. We expect that there are more problems complete for XNLP. Typical candidates are problems that are known to be hard for $\mathrm{W}[t]$ for all integers t. Possibly, in some cases, only small modifications of proofs may be needed, but in other cases, new proofs have to be invented. A number of such candidates are the following:

- Linear graph ordering problems, like Colored Cutwidth (and variants), Feasible Register Allocation, Triangulating Colored Graphs (see [43]), Topological Bandwidth.
- Domino Treewidth, see [44]. (We conjecture that XNLP-hardness can be proved with help of Accepting NNCCM; membership in XNLP is unclear due to the tree-like structure of positive instances.)
- Shortest Common Supersequence as mentioned in [5].

- Restricted Completion to a Proper Interval Graph with Bounded Clique Size, see [45],
- Problems parameterized by treewidth or pathwidth that are known to be in XP but not in FPT. Membership in XNLP for several problems parameterized by treewidth is also open (including Dominating Set and Independent Set). See e.g. [46] for some candidates for problems that might be XNLP-hard when parameterized by treewidth or pathwidth.

Acknowledgment

The authors would like to thank the referees for useful comments.

References

[1] L. J. Stockmeyer, "The polynomial-time hierarchy," *Theor. Comput. Sci.*, vol. 3, no. 1, pp. 1–22, 1976. [Online]. Available: https://doi.org/10.1016/0304-3975(76)90061-X

[2] S. Arora and B. Barak, *Computational Complexity: A Modern Approach*. Cambridge University Press, 2007. [Online]. Available: https://theory.cs.princeton.edu/complexity/book.pdf

[3] R. G. Downey and M. R. Fellows, "Fixed-parameter tractability and completeness I: Basic results," *SIAM J. Comput.*, vol. 24, no. 4, pp. 873–921, 1995. [Online]. Available: https://doi.org/10.1137/S0097539792228228

[4] ——, "Fixed-parameter tractability and completeness II: On completeness for $W[1]$," *Theoretical Computer Science*, vol. 141, no. 1&2, pp. 109–131, 1995. [Online]. Available: https://doi.org/10.1016/0304-3975(94)00097-3

[5] ——, *Parameterized Complexity*. Springer, 1999.

[6] A. Abboud, K. Lewi, and R. Williams, "Losing weight by gaining edges," in *Algorithms - ESA 2014 - 22th Annual European Symposium, Wroclaw, Poland, September 8-10, 2014. Proceedings*, ser. Lecture Notes in Computer Science, A. S. Schulz and D. Wagner, Eds., vol. 8737. Springer, 2014, pp. 1–12. [Online]. Available: https://doi.org/10.1007/978-3-662-44777-2_1

[7] H. L. Bodlaender, M. R. Fellows, and M. Hallett, "Beyond *NP*-completeness for problems of bounded width: Hardness for the W hierarchy," in *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, STOC 1994*. New York: ACM Press, 1994, pp. 449–458.

[8] H. L. Bodlaender, "Parameterized complexity of bandwidth of caterpillars and weighted path emulation," *arXiv:2012.01226*, 2020, to appear in Proceedings WG 2021. [Online]. Available: https://arxiv.org/abs/2012.01226

[9] M. R. Fellows and F. A. Rosamond, "Collaborating with Hans: Some remaining wonderments," in *Treewidth, Kernels, and Algorithms - Essays Dedicated to Hans L. Bodlaender on the Occasion of His 60th Birthday*, ser. Lecture Notes in Computer Science, F. V. Fomin, S. Kratsch, and E. J. van Leeuwen, Eds., vol. 12160. Springer, 2020, pp. 7–17. [Online]. Available: https://doi.org/10.1007/978-3-030-42071-0_2

[10] M. Elberfeld, C. Stockhusen, and T. Tantau, "On the space and circuit complexity of parameterized problems: Classes and completeness," *Algorithmica*, vol. 71, no. 3, pp. 661–701, 2015. [Online]. Available: https://doi.org/10.1007/s00453-014-9944-y

[11] H. L. Bodlaender, R. G. Downey, M. R. Fellows, M. T. Hallett, and H. T. Wareham, "Parameterized complexity analysis in computational biology," *Comput. Appl. Biosci.*, vol. 11, no. 1, pp. 49–57, 1995. [Online]. Available: https://doi.org/10.1093/bioinformatics/11.1.49

[12] M. Pilipczuk and M. Wrochna, "On space efficiency of algorithms working on structural decompositions of graphs," *ACM Trans. Comput. Theory*, vol. 9, no. 4, pp. 18:1–18:36, 2018. [Online]. Available: https://doi.org/10.1145/3154856

[13] M. R. Fellows, F. V. Fomin, D. Lokshtanov, F. A. Rosamond, S. Saurabh, S. Szeider, and C. Thomassen, "On the complexity of some colorful problems parameterized by treewidth," *Inf. Comput.*, vol. 209, no. 2, pp. 143–153, 2011. [Online]. Available: https://doi.org/10.1016/j.ic.2010.11.026

[14] H. L. Bodlaender and M. R. Fellows, "$W[2]$-hardness of precedence constrained K-processor scheduling," *Operations Research Letters*, vol. 18, no. 2, pp. 93–97, 1995. [Online]. Available: https://doi.org/10.1016/0167-6377(95)00031-9

[15] Y. Chen, J. Flum, and M. Grohe, "Bounded nondeterminism and alternation in parameterized complexity theory," in *18th Annual IEEE Conference on Computational Complexity (Complexity 2003), 7-10 July 2003, Aarhus, Denmark*. IEEE Computer Society, 2003, pp. 13–29. [Online]. Available: https://doi.org/10.1109/CCC.2003.1214407

[16] H. L. Bodlaender, C. Groenland, J. Nederlof, and C. M. F. Swennenhuis, "Parameterized problems complete for nondeterministic FPT time and logarithmic space," *arXiv:2105.14882*, 2021.

[17] K. Pietrzak, "On the parameterized complexity of the fixed alphabet shortest common supersequence and longest common subsequence problems," *J. Comput. Syst. Sci.*, vol. 67, no. 4, pp. 757–771, 2003. [Online]. Available: https://doi.org/10.1016/S0022-0000(03)00078-3

[18] M. R. Fellows, D. Hermelin, F. A. Rosamond, and S. Vialette, "On the parameterized complexity of multiple-interval graph problems," *Theoretical Computer Science*, vol. 410, no. 1, pp. 53–61, 2009. [Online]. Available: https://doi.org/10.1016/j.tcs.2008.09.065

[19] K. Jansen and P. Scheffler, "Generalized coloring for tree-like graphs," *Discret. Appl. Math.*, vol. 75, no. 2, pp. 135–155, 1997. [Online]. Available: https://doi.org/10.1016/S0166-218X(96)00085-6

[20] M. Elberfeld, A. Jakoby, and T. Tantau, "Logspace versions of the theorems of Bodlaender and Courcelle," in *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*. IEEE Computer Society, 2010, pp. 143–152. [Online]. Available: https://doi.org/10.1109/FOCS.2010.21

[21] M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh, *Parameterized Algorithms*. Springer, 2015. [Online]. Available: https://doi.org/10.1007/978-3-319-21275-3

[22] H. L. Bodlaender, M. Cygan, S. Kratsch, and J. Nederlof, "Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth," *Inf. Comput.*, vol. 243, pp. 86–111, 2015. [Online]. Available: https://doi.org/10.1016/j.ic.2014.12.008

[23] J. A. Telle and A. Proskurowski, "Algorithms for vertex partitioning problems on partial *k*-trees," *SIAM J. Discret. Math.*, vol. 10, no. 4, pp. 529–550, 1997. [Online]. Available: https://doi.org/10.1137/S0895480194275825

[24] H. L. Bodlaender and R. H. Möhring, "The pathwidth and treewidth of cographs," *SIAM J. Discret. Math.*, vol. 6, no. 2, pp. 181–188, 1993. [Online]. Available: https://doi.org/10.1137/0406014

[25] R. G. Downey and M. R. Fellows, *Fundamentals of Parameterized Complexity*, ser. Texts in Computer Science. Springer, 2013. [Online]. Available: https://doi.org/10.1007/978-1-4471-5559-1

[26] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[27] D. Prot and O. Bellenguez-Morineau, "A survey on how the structure of precedence constraints may change the complexity class of scheduling problems," *Journal of Scheduling*, vol. 21, no. 1, pp. 3–16, 2018. [Online]. Available: https://doi.org/10.1007/s10951-017-0519-z

[28] J. P. Fishburn and R. A. Finkel, "Quotient networks," *IEEE Trans. Comput.*, vol. C-31, pp. 288–295, 1982.

[29] H. L. Bodlaender, "The complexity of finding uniform emulations on paths and ring networks," *Information and Computation*, vol. 86, no. 1, pp. 87–106, 1990.

[30] C. H. Papadimitriou, "The NP-completeness of the bandwidth minimization problem," *Computing*, vol. 16, pp. 263–270, 1976.

[31] B. Monien, "The bandwidth minimization problem for caterpillars with hair length 3 is NP-complete," *SIAM Journal on Algebraic and Discrete Methods*, vol. 7, pp. 505–512, 1986.

[32] J. B. Saxe, "Dynamic programming algorithms for recognizing small-bandwidth graphs in polynomial time," *SIAM J. Algebraic and Discrete Methods*, vol. 1, pp. 363–369, 1980.

[33] E. M. Gurari and I. H. Sudborough, "Improved dynamic programming algorithms for bandwidth minimization and the MinCut linear arrangement problem," *J. of Algorithms*, vol. 5, pp. 531–546, 1984.

[34] M. S. Dregi and D. Lokshtanov, "Parameterized complexity of bandwidth on trees," in *41st International Colloquium on Automata, Languages, and Programming, ICALP 2014*, ser. Lecture Notes in Computer Science, J. Esparza, P. Fraigniaud, T. Husfeldt, and E. Koutsoupias, Eds., vol. 8572. Springer, 2014, pp. 405–416. [Online]. Available: https://doi.org/10.1007/978-3-662-43948-7_34

[35] A. Haddadan, T. Ito, A. E. Mouawad, N. Nishimura, H. Ono, A. Suzuki, and Y. Tebbal, "The complexity of dominating set reconfiguration," *Theor. Comput. Sci.*, vol. 651, no. C, pp. 37–49, 2016. [Online]. Available: https://doi.org/10.1016/j.tcs.2016.08.016

[36] A. E. Mouawad, N. Nishimura, V. Raman, N. Simjour, and A. Suzuki, "On the parameterized complexity of reconfiguration problems," *Algorithmica*, vol. 78, no. 1, pp. 274–297, 2017.

[37] M. Bonamy, P. Dorbec, and P. Ouvrard, "Dominating sets reconfiguration under token sliding," *arXiv:1912.03127*, 2019.

[38] T. Ito, E. D. Demaine, N. J. Harvey, C. H. Papadimitriou, M. Sideri, R. Uehara, and Y. Uno, "On the complexity of reconfiguration problems," *Theoretical Computer Science*, vol. 412, no. 12, pp. 1054–1065, 2011.

[39] R. A. Hearn and E. Demaine, "Pspace-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation," *Theor. Comput. Sci.*, vol. 343, pp. 72–96, 2005.

[40] T. Ito, M. Kamiński, H. Ono, A. Suzuki, R. Uehara, and K. Yamanaka, "Parameterized complexity of independent set reconfiguration problems," *Discrete Applied Mathematics*, vol. 283, pp. 336–345, 2020.

[41] J. van den Heuvel, "The complexity of change," *arXiv:1312.2816*, 2013.

[42] M. Wehar, "On the complexity of intersection non-emptiness problems," Ph.D. dissertation, University at Buffalo, State University of New York, 2016.

[43] H. L. Bodlaender, M. R. Fellows, M. T. Hallett, T. Wareham, and T. J. Warnow, "The hardness of perfect phylogeny, feasible register assignment and other problems on thin colored graphs," *Theor. Comput. Sci.*, vol. 244, no. 1-2, pp. 167–188, 2000. [Online]. Available: https://doi.org/10.1016/S0304-3975(98)00342-9

[44] H. L. Bodlaender and J. Engelfriet, "Domino treewidth," *J. Algorithms*, vol. 24, no. 1, pp. 94–123, 1997. [Online]. Available: https://doi.org/10.1006/jagm.1996.0854

[45] H. Kaplan and R. Shamir, "Pathwidth, bandwidth, and completion problems to proper interval graphs with small cliques," *SIAM J. Comput.*, vol. 25, no. 3, pp. 540–561, 1996. [Online]. Available: https://doi.org/10.1137/S0097539793258143

[46] S. Szeider, "Not so easy problems for tree decomposable graphs," *arXiv:1107.1177*, 2011. [Online]. Available: http://arxiv.org/abs/1107.1177