Discrete Optimization 8 (2011) 50–60

Contents lists available at ScienceDirect

Discrete Optimization

journal homepage: www.elsevier.com/locate/disopt

Parameterized complexity and approximability of the Longest Compatible Sequence problem☆

S. Guillemot

Institut Gaspard Monge - Université Paris-Est, 5 boulevard Descartes, Champs-sur-Marne, 77454 Marne-la-Vallée, France

ARTICLE INFO

Article history:
Available online 24 December 2010

Keywords:
Parameterized complexity
Sequence comparison
Longest common subsequence
Longest compatible sequence

ABSTRACT

We introduce the Longest Compatible Sequence (Slcs) problem. This problem deals with *p-sequences*, which are strings on a given alphabet where each letter occurs at most once. The Slcs problem takes as input a collection of k p-sequences on a common alphabet L of size n, and seeks a p-sequence on L which respects the precedence constraints induced by each input sequence, and is of maximal length with this property. We investigate the parameterized complexity and the approximability of the problem. As a by-product of our hardness results for the Slcs problem, we derive new hardness results for the Longest Common Subsequence problem and other problems that are hard for the W-hierarchy.

1. Introduction

The comparison of several sequences is an important task in several fields such as computational biology, pattern recognition, scheduling, data compression and data mining. Starting with [1], the computational complexity of several consensus problems on sequences has been investigated. As it was later realized, a natural framework to conduct these studies is the theory of parameterized complexity [2–4].

In this article, we initiate the study of a new consensus problem on collections of sequences, motivated by applications to the comparison of gene orders, and applications to the rank aggregation problem. The problem we introduce is called the Longest Compatible Sequence, and is abbreviated by Slcs. This problem deals with *p-sequences* [5]: we call a *p-sequence* on an alphabet L a string over L where each letter occurs at most once. Given a collection $\mathcal{C} = \{s_1, \ldots, s_k\}$ of p-sequences on a common alphabet L of size n, the Longest Compatible Sequence problem seeks a longest *compatible sequence* for $\mathcal{C}$, which is a p-sequence s on L respecting the precedence constraints induced by each input sequence. We also consider the complementary minimization problem, denoted by CSlcs, where the goal is to minimize the number of labels missing from a compatible sequence. In addition to studying the approximability of these optimization problems, we also investigate the parameterized complexity of their natural parameterizations. The corresponding parameters are denoted by q for the Slcs problem, and by p for the CSlcs problem.

We now discuss two potential applications of the problem. The first application is the *comparison of gene orders*. Identifying conservation among gene orders for several organisms is an important issue in bioinformatics, since it helps in gene prediction and can also be used in phylogenetic reconstruction as an alternative to DNA sequence analysis (see [6] for a survey). Given a set of k organisms $\mathcal{S}$ for which we have identified a set of n homologous genes $\mathcal{G}$, each organism can be described by its gene order, which is a p-sequence on $\mathcal{G}$. Therefore, seeking a largest gene order which is conserved among all organisms under study amounts to seeking a largest compatible sequence for the collection $\mathcal{C} = \{s_1, \ldots, s_k\}$ on the alphabet $\mathcal{G}$. The second application is the *aggregation of incomplete rankings*. This is a variant of the well-studied problem of

☆ An extended abstract of this article was published in *Proceedings of the 3rd International Workshop on Parameterized and Exact Computation* (IWPEC 2008), volume 5018 of *Lecture Notes in Computer Science*, pp. 115–128, Springer-Verlag, 2008.

E-mail address: sylvain.guillemot@univ-mlv.fr.

1572-5286/$ – see front matter © 2011 Published by Elsevier B.V.
doi:10.1016/j.disopt.2010.08.003

rank aggregation, which consists in merging k complete rankings of the same set of n elements in a single ranking (see [7] for a survey). Rank aggregation is useful in several situations, such as combining answers from search engines [8,9] or searching similarities in databases [10]. This framework can be extended to handle incomplete rankings, which are rankings defined only on a subset of the elements. Here, the Slcs problem can be used to find a largest subset of the elements on which all the rankings agree.

Our results are as follows. On the positive side, we give polynomial, approximation, and FPT algorithms in Section 3. Among other results, we show that the Slcs problem can be solved in $O(kn^k)$ time, and we give a k-approximation algorithm and an $O^*(k^p)$ FPT algorithm for the CSlcs problem. On the negative side, we describe parameterized intractability results in Section 4. We show W[1]-completeness of Slcs parameterized by (q, k), and we show WNL-completeness of Slcs parameterized by k. The latter result relies on the definition of a new parameterized complexity class WNL, and turns out to have important consequences for the parameterized complexity of other problems. Namely, we show in Section 5 that this result implies WNL-hardness results for the Lcs problem parameterized by the number of sequences, and for other problems previously classified as "hard for the W-hierarchy" [11–14]. Thus, the thesis of [11], which proposes to consider hardness for the W-hierarchy as a new intractability measure, can be recast with the more natural notion of WNL-hardness.

2. Definitions

We begin with definitions related to *sequences*. After [5], we call a *p-sequence* (or simply a *sequence*) on L a string s on L where each letter occurs at most once. The letters are called *labels*, and the set of letters appearing in s is called the *label set* of s, and is denoted by $L(s)$. The ith letter of s is denoted by $s[i]$, and the length of s is denoted by $|s|$. Given two elements $x, y \in L(s)$, $x <_s y$ means that x precedes y in s. Given $x \in L(s)$, we denote by $pred_s(x)$, resp. $succ_s(x)$, the predecessor, resp. successor, of x in s, or $\perp$ if no such label exists. Given a set $L' \subseteq L$, the *restriction* of s to L', denoted by $s|L'$, is the subsequence obtained from s by keeping labels in L'. Two sequences s, s' *agree* iff $s|L(s') = s'|L(s)$. Equivalently, they must verify the following: for each $x, y \in L(s) \cap L(s')$, $x <_s y$ iff $x <_{s'} y$.

We now give definitions related to *collections* and *compatible sequences*. Given a set L of n elements, a *collection on* L is a family $\mathcal{C} = \{s_1, \ldots, s_k\}$ of sequences on L; in the following, we will assume that each label of L appears in at least one sequence, i.e., $L = \cup_{i\in[k]} L(s_i)$. Given a set $L' \subseteq L$, the *restriction* of $\mathcal{C}$ to L' is the collection $\mathcal{C}|L'$ on L' defined by $\mathcal{C}|L' = \{s_1|L', \ldots, s_k|L'\}$. A *compatible sequence* for $\mathcal{C}$ is a sequence s such that $L(s) \subseteq L$, and for each $i \in [k]$ the sequences s, s_i agree. Equivalently, s must verify the following: for each $x, y \in L(s)$, and for each $i \in [k]$, if $x <_s y$ and $x, y \in L(s_i)$ then $x <_{s_i} y$. A *total compatible sequence* for $\mathcal{C}$ is a compatible sequence for $\mathcal{C}$ with label set L. $\mathcal{C}$ is said to be *compatible* iff it admits a total compatible sequence. A *conflict* in $\mathcal{C}$ is a set $L' \subseteq L$ such that $\mathcal{C}|L'$ is not compatible.

We now introduce definitions related to the Slcs problem. The Slcs problem asks: given a collection $\mathcal{C}$ on L, find a set $L' \subseteq L$ of maximum cardinality such that $\mathcal{C}|L'$ is compatible; observe that this is equivalent to seeking a longest compatible sequence for $\mathcal{C}$. We denote by $\text{Slcs}(\mathcal{C})$ the size of such an optimal solution. The complementary CSlcs problem asks: given a collection $\mathcal{C}$ on L, find a set $L' \subseteq L$ of minimum cardinality such that $\mathcal{C}|(L \setminus L')$ is compatible.

To simplify notations, we also view these optimization problems as decision problems, i.e., we also denote by Slcs the decision problem which takes a collection $\mathcal{C}$ and an integer q, and asks if $\text{Slcs}(\mathcal{C}) \geq q$; similarly for CSlcs with the parameter p, asking if $\text{CSlcs}(\mathcal{C}) \leq p$. We use a bracket notation to denote parameterizations of the problem, e.g. Slcs$[q, k]$ stands for Slcs parameterized by the pair of parameters (q, k).

3. Algorithmic results for Slcs

In this section, we present polynomial and FPT algorithms for the Slcs problem. We first introduce a few notations and definitions.

Consider a collection $\mathcal{C} = \{s_1, \ldots, s_k\}$ on L. For each i, we define $L^\top(s_i) = L(s_i) \cup \{\top\}$, and we extend $<_{s_i}$ to $L^\top(s_i)$ such that $x <_{s_i} \top$ for each $x \in L(s_i)$. A *position in* $\mathcal{C}$ is a tuple $\pi = (x_1, \ldots, x_k)$, where $x_i \in L^\top(s_i)$. The *initial position* is $\pi_\perp = (s_1[1], \ldots, s_k[1])$, and the *final position* is $\pi_\top = (\top, \ldots, \top)$. The *index set* of a position π is $I(\pi) = \{i \in [k] : \pi[i] \neq \top\}$. The *label set* of π is $S(\pi) = \{\pi[i] : i \in I(\pi)\}$. In other words, a position π consists of one cursor in each s_i, which points either to a label of s_i or to the end of s_i. Then $I(\pi)$ is the set of sequences in which the cursor is not located at the end, and $S(\pi)$ is the set of labels under the cursors.

We define an order relation $\leq_{\mathcal{C}}$ on positions in $\mathcal{C}$ as follows: given π, π' positions in $\mathcal{C}$, $\pi \leq_{\mathcal{C}} \pi'$ iff $\pi[i] \leq_{s_i} \pi'[i]$ for each $i \in [k]$. We also define two notions of *successor positions*. Let π be a position in $\mathcal{C}$. Given $i \in I(\pi)$, we define $Succ_i(\pi)$ as the position π' such that (i) $\pi'[i] = succ_{s_i}(\pi[i])$, (ii) $\pi'[j] = \pi[j]$ for each $j \neq i$. Given $a \in S(\pi)$, we define $Succ_a(\pi)$ as the position π' such that (i) $\pi'[i] = succ_{s_i}(\pi[i])$ if $\pi[i] = a$, (ii) $\pi'[i] = \pi[i]$ otherwise. Intuitively, $Succ_i(\pi)$ advances the cursor in s_i, while $Succ_a(\pi)$ advances the cursor in every sequence containing a under the cursor.

We first show that the Slcs problem can be solved in polynomial time for fixed k, using dynamic programming:

Proposition 1. *The* Slcs *problem can be solved in* $O(kn^k)$ *time and* $O(n^k)$ *space.*

Proof. Given a position π in $\mathcal{C}$ and a label $x \in L$, we say that x is *allowed* at π iff, for each $i \in [k]$ such that $x \in L(s_i)$, we have $\pi[i] \leq_{s_i} x$. We denote by $L(\pi)$ the set of labels allowed at π. Given π position in $\mathcal{C}$, let $\textsc{Slcs}(\pi)$ denote the size of a longest compatible sequence of $\mathcal{C}|L(\pi)$. We denote by $F(\pi)$ the set of *full* elements of $S(\pi)$, i.e., the elements $a \in S(\pi)$ such that for each $i \in [k]$, $a \in L(s_i) \Rightarrow a = \pi[i]$. Given two positions π, π' and $a \in L(\mathcal{C})$, $\pi \rightarrow_a \pi'$ holds if $a \in F(\pi)$ and $\pi' \geq_{\mathcal{C}} succ_a(\pi)$. A π*-chain* is a chain $\pi_1 \rightarrow_{a_1} \pi_1 \rightarrow_{a_1} \cdots \rightarrow_{a_m} \pi_{m+1}$, with $\pi_1 \geq_{\mathcal{C}} \pi$ and $\pi_{m+1} = \pi_\top$. The *length* of the chain is m. The following can be shown.

Claim. $\textsc{Slcs}(\pi)$ is the length of a longest π-chain.

The above claim yields a dynamic-programming algorithm for solving the Slcs problem in $O(kn^k)$ time. The algorithm computes, for each position π, the size $\textsc{Slcs}(\pi)$ of a longest π-chain, using the following recurrence relations:

- if $\pi = \pi_\top$, $\textsc{Slcs}(\pi) = 0$;
- if $\pi \neq \pi_\top$, $\textsc{Slcs}(\pi) = \max(\textsc{Slcs}_1(\pi), 1 + \textsc{Slcs}_2(\pi))$, where:

$$\textsc{Slcs}_1(\pi) = \max\{\textsc{Slcs}(Succ_i(\pi)) : i \in I(\pi)\}$$
$$\textsc{Slcs}_2(\pi) = \max\{\textsc{Slcs}(Succ_a(\pi)) : a \in F(\pi)\}.$$

At the end of the algorithm, $\textsc{Slcs}(\mathcal{C})$ is obtained as $\textsc{Slcs}(\pi_\perp)$. Since there are $O(n^k)$ positions π, and since each value $\textsc{Slcs}(\pi)$ can be computed in $O(k)$ time from the values $\textsc{Slcs}(\pi')$ ($\pi' >_{\mathcal{C}} \pi$), the algorithm runs in the claimed time and space bounds. □

We now describe algorithms for the CSlcs problem. Given a collection $\mathcal{C} = \{s_1, \ldots, s_k\}$ on L, we define the digraph $G(\mathcal{C})$ as follows: (i) its vertex set is L, (ii) it contains an arc (x, y) whenever there exists $i \in [k]$ such that $x, y \in L(s_i)$ and $x <_{s_i} y$. It is easily seen that $\mathcal{C}$ is compatible iff $G(\mathcal{C})$ is acyclic, which yields a reduction from the CSlcs problem to the Directed Feedback Vertex Set (DFVS) problem. Known FPT and approximation algorithms for the DFVS problem [15–17] yield the following results for the CSlcs problem.

Proposition 2. (i) *The* CSlcs *problem can be solved in* $O(4^p p! poly(n))$ *time.* (ii) *The* CSlcs *problem can be approximated within* $O(\log n \log\log n)$ *in polynomial time.*

We now describe faster FPT and approximation algorithms for the problem when k is bounded. Though the problem is solvable in $O(kn^k)$ time in this case, in practical applications it may be preferable to have algorithms with running time that is linear in n. This is the case for the algorithms we present (Proposition 4). They rely on the following result.

Proposition 3. *Given a collection* $\mathcal{C}$*, in* $O(kn)$ *time we can decide if* $\mathcal{C}$ *is compatible, return a total compatible sequence in the case of a positive answer, or return a conflict of size at most* k *in the case of a negative answer.*

Proof. Before describing the algorithm, let us introduce the following notations. Suppose that $\mathcal{C} = \{s_1, \ldots, s_k\}$ is a collection on L. Given position π in $\mathcal{C}$, and $x \in L$, let $n_\pi(x)$ denote the number of indices $i \in [k]$ such that ($x \in L(s_i)$ and $\pi[i] <_{s_i} x$).

The algorithm maintains a position π in $\mathcal{C}$, and for each $x \in L$ a counter n_x equal to $n_\pi(x)$. Additionally, it maintains a sequence s, which is the prefix of a hypothetical compatible sequence for $\mathcal{C}$. We start with $s = \epsilon$, $\pi = \pi_\perp$ and each n_x initialized to the number of $i \in [k]$ such that x is a non-initial label of s_i. While $\pi \neq \pi_\top$, we seek $x \in S(\pi)$ such that $n_x = 0$. If no such x exists, then the algorithm answers "no" and returns $S(\pi)$. Otherwise, (i) we choose an x_i, and for each $i \in [k]$ such that $\pi[i] = x$, where y is the successor of x in s_i (if it exists), we decrement n_y, (ii) we set $\pi \leftarrow succ_x(\pi)$, (iii) we set $s \leftarrow sx$. When $\pi = \pi_\top$ is reached, the algorithm answers "yes" and returns s.

The correctness of the algorithm follows from the fact that (i) if it answers negatively by returning $S(\pi)$, then $S(\pi)$ is a conflict of size $\leq k$ between $\mathcal{C}$, (ii) if it answers positively by returning s, then s is a total compatible sequence for $\mathcal{C}$. The running time is easily seen to be $O(kn)$, since the initialization takes $O(kn)$ time, and since in each step finding an $x \in S(\pi)$ such that $n_x = 0$ and updating s, π and the counters takes $O(k)$ time. □

As a consequence of Proposition 3, we obtain the following.

Proposition 4. (i) *The* CSlcs *problem can be solved in* $O(k^p \times kn)$ *time.* (ii) *The* CSlcs *problem can be approximated within* k *in* $O(kn)$ *time.*

4. Hardness results for the Slcs problem

We present two parameterized intractability results for the Slcs problem. The problem is shown to be W[1]-complete w.r.t. the parameters q, k (Section 4.2) and WNL-complete w.r.t. the parameter k (Section 4.3).

4.1. The classes W[1] and WNL

Let us consider the following problem.

Name: Nondeterministic Turing Machine Computation

Instance: A nondeterministic Turing machine M, an integer q in unary, and an integer k.

Question: Does M accept the empty string in q steps by examining at most k cells?

Given two parameterized problems Π, Π', we recall that a *parameterized reduction* (or *fpt-reduction*) from Π to Π' is an algorithm which maps each instance $I = (x, k)$ of Π to an instance $I' = (x', k')$ of Π', such that (i) the algorithm runs in $f(k)|x|^c$ time for some function f and some constant c; (ii) there exists a function g such that $k' \leq g(k)$; (iii) I is a positive instance of Π iff I' is a positive instance of Π'. Given a parameterized problem Π, we denote by $[\Pi]^{\text{fpt}}$ the set of problems Π' fpt-reducible to Π. We then define the classes W[1] and WNL as follows.

Definition 1. $\mathsf{W}[1] = [\text{NTMC}[q]]^{\text{fpt}}$, $\mathsf{WNL} = [\text{NTMC}[k]]^{\text{fpt}}$.

Our definition of W[1] is consistent with the results of [18], and shows the parallel between the classes W[1] and WNL, corresponding respectively to time-bounded and space-bounded computations of a nondeterministic Turing machine.

4.2. Complexity w.r.t. q, k

We need the following notations. If $s = a_1 \cdots a_m$ is a sequence, its mirror image is $\widetilde{s} = a_m \cdots a_1$. If $(V, <_V)$ is a total order and $\{s_x : x \in V\}$ is a family of sequences with disjoint label sets, we denote by $\prod_{x\in(V,<_V)}$ their concatenation in the order $<_V$. If V is the interval of integers $[p, q]$ and $<_V$ is the natural order on $\mathbb{N}$, then $\prod_{x\in(V,<_V)} s_x$ is abbreviated as $\prod_{i=p}^{q} s_i$.

Proposition 5. Slcs$[q, k]$ *is* W[1]*-complete.*

Proof. Membership in W[1] can be shown by reduction to NTMC$[q]$. W[1]-hardness is proved by a parameterized reduction from the Partitioned Clique [19]. Let $I = (G, k)$ be an instance of Partitioned Clique, where $G = (V, E)$ is a k-partite graph with partition $V_1, \ldots, V_k$. The corresponding instance $I' = (\mathcal{C}, k', q')$ of Slcs$[q, k]$ is defined as follows. We set $k' = 2k + 2$, $q' = 2k + k(k-1)/2$. We define the following collection $\mathcal{C}$.

- Label set: we introduce labels $a[v]$, $b[v]$ for each $v \in V$, and a label $c[e]$ for each $e \in E$.
- Sequences: we create two sequences s, s' and $2k$ sequences $t_1, t'_1, \ldots, t_k, t'_k$. We want to enforce that:
  1. a compatible sequence of length q' has the form
  $$s = \left(\prod_{i=1}^{k} a[v_i]\right)\left(\prod_{i=1}^{k}\prod_{j=i+1}^{k} c[e_{i,j}]\right)\left(\prod_{i=1}^{k} b[v'_i]\right)$$
  with $v_i, v'_i \in V_i$ (for each $i \in [k]$), $e_{i,j} \in E_{i,j}$ (for each $i, j \in [k]$, $i < j$);
  2. in addition, we have $v_i = v'_i$ for each $i \in [k]$, and $e_{i,j} = \{v_i, v_j\}$ for each $i, j \in [k]$, $i < j$.

  The sequences s, s' will be *control sequences*, whose role is to enforce point 1. The sequences t_i, t'_i will be *selection sequences*, whose role is to enforce point 2.

  These sequences are defined as follows. Let $<_V$ be an arbitrary total order on V, and let $<_E$ be an arbitrary total order on E. For $i, j \in [k]$ $(i < j)$, let $E_{i,j}$ denote the set of edges of G having one endpoint in V_i and one in V_j. We first define the following sequences:

$$\forall i \in [k], \quad A[i] = \prod_{v\in(V_i,<_V)} a[v], \qquad B[i] = \prod_{v\in(V_i,<_V)} b[v],$$
$$\forall i, j \in [k](i < j), \quad C[i, j] = \prod_{e\in(E_{i,j},<_E)} c[e].$$

  We then define s, s' as follows:

$$s = \left(\prod_{i=1}^{k} A[i]\right)\left(\prod_{i=1}^{k}\prod_{j=i+1}^{k} C[i, j]\right)\left(\prod_{i=1}^{k} B[i]\right)$$
$$s' = \left(\prod_{i=1}^{k} \widetilde{A[i]}\right)\left(\prod_{i=1}^{k}\prod_{j=i+1}^{k} \widetilde{C[i, j]}\right)\left(\prod_{i=1}^{k} \widetilde{B[i]}\right).$$

Suppose that $i, j \in [k]$, $i < j$. Given $v \in V_i \cup V_j$, let $E_{i,j}(v)$ denote the set of edges of $E_{i,j}$ incident to v. Now, for each $i \in [k]$, we define t_i, t'_i as follows:

$$t_i = \prod_{v \in (V_i, <_V)} a[v] \left(\prod_{j=i+1}^{k} \prod_{e \in (E_{i,j}(v), <_E)} c[e] \right) b[v]$$

$$t'_i = \prod_{v \in (V_i, >_V)} a[v] \left(\prod_{j=1}^{i-1} \prod_{e \in (E_{i,j}(v), <_E)} c[e] \right) b[v].$$

The reduction is clearly computable in polynomial time. Its correctness is ensured by the following.

Claim. G has a partitioned clique iff $\mathcal{C}$ has a compatible sequence of length q'.

Proof. ($\Rightarrow$): suppose that G has a partitioned clique $V' = \{v_1, \ldots, v_k\}$, with $v_i \in V_i$. Let us consider the sequence r defined as follows:

$$r = \left(\prod_{i=1}^{k} a[v_i] \right) \left(\prod_{i=1}^{k} \prod_{j=i+1}^{k} c[\{v_i, v_j\}] \right) \left(\prod_{i=1}^{k} b[v_i] \right).$$

Observe that r is well defined, since $v_i \in V_i$ for each i, and $\{v_i, v_j\} \in E_{i,j}$ for each i, j. Moreover, r has length $2k + k(k-1)/2 = q'$. We claim that r is a compatible sequence for $\mathcal{C}$. Indeed, r clearly agrees with s and with s'. Furthermore, for each $i \in [k]$, r agrees with t_i, since

$$r|L(t_i) = t_i|L(r) = a[v_i] \left(\prod_{j=i+1}^{k} c[\{v_i, v_j\}] \right) b[v_i].$$

Likewise, r agrees with t'_i.

($\Leftarrow$): suppose that $\mathcal{C}$ has a compatible sequence r of length q'. We make the following observations. □

Observation 1: There exists $v_i, v'_i \in V_i$ (for each i), $e_{i,j} \in E_{i,j}$ (for each i, j) such that

$$r = \left(\prod_{i=1}^{k} a[v_i] \right) \left(\prod_{i=1}^{k} \prod_{j=i+1}^{k} c[e_{i,j}] \right) \left(\prod_{i=1}^{k} b[v'_i] \right).$$

Proof. Since s and s' are complete sequences, r must be a subsequence of s and s'. By the definitions of s and s', the subsequence r contains at most one label in each block $A[i]$, $B[i]$, $C[i, j]$. Since these blocks are in number q', it follows that r contains exactly one label in each block. □

Observation 2: We have (i) for each i, $v_i = v'_i$; (ii) for each i, j, $e_{i,j} = \{v_i, v_j\}$.

Proof. Let us prove (i). Suppose for contradiction that $v_i \neq v'_i$. If $v_i <_V v'_i$, then $a[v_i] <_r b[v'_i]$ but $b[v'_i] <_{t'_i} a[v_i]$; impossible. If $v_i >_V v'_i$, then $a[v_i] <_r b[v'_i]$ but $b[v'_i] <_{t_i} a[v_i]$; impossible. Thus, we must have $v_i = v'_i$. □

Let us prove (ii). Suppose for contradiction that there exists i, j such that $e_{i,j} = \{x, y\}$ with $x \in V_i$, $y \in V_j$ and ($x \neq v_i$ or $y \neq v_j$).

Suppose first that $x \neq v_i$. If $x <_V v_i$, then $c[e_{i,j}] <_{t_i} a[v_i]$ but $a[v_i] <_r c[e_{i,j}]$; impossible. If $x >_V v_i$, then $b[v_i] <_{t_i} c[e_{i,j}]$ but $c[e_{i,j}] <_r b[v_i]$; impossible.

Suppose now that $x = v_i$ but $y \neq v_j$. If $y >_V v_j$, then $c[e_{i,j}] <_{t'_j} a[v_j]$ but $a[v_j] <_r c[e_{i,j}]$; impossible. If $y <_V v_j$, then $b[v_j] <_{t'_j} c[e_{i,j}]$ but $c[e_{i,j}] <_r b[v_j]$; impossible.

We conclude from Observation 2 that the set $V' = \{v_1, \ldots, v_k\}$ is a partitioned clique of G. □

4.3. Complexity w.r.t. k

While NTMC[k] is the canonical WNL-complete problem, we use another problem for the proof of our result. In the following, by a $q \times k$-*grid* we mean a directed grid with q lines and k columns, i.e., a digraph $G = (V, A)$ with vertex set $V = \{v_{i,j} : 1 \leq i \leq q, 1 \leq j \leq k\}$ and with arc set $A = \{(v_{i,j}, v_{i+1,j}) : 1 \leq i < q, 1 \leq j \leq k\} \cup \{(v_{i,j}, v_{i,j+1}) : 1 \leq i \leq q, 1 \leq j < k\}$. For $R \in \{=, \leq\}$, we define the following problems.

Name: GRID LABELLING-R (GL-R).

Instance: An integer m, a $q \times k$-grid $G = (V, A)$, a set S, a partition $\{S_v : v \in V\}$ of S, and for each $a = (u, v) \in A$ a function $f_a : S_u \cup S_v \to [m]$.

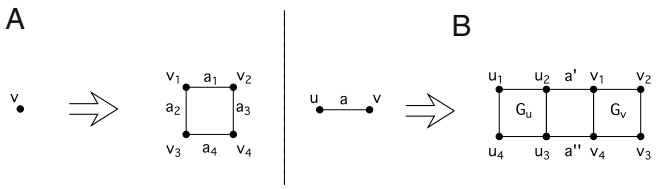

Fig. 1. (A): the gadget G_v associated to a vertex v of G. (B): interconnecting two gadgets G_u, G_v corresponding to an horizontal arc $a = (u, v)$.

Question: Does there exist an R-admissible labelling of G?

A labelling of G is an assignment of a value $l_v \in S_v$ to each $v \in V$. The labelling is R-admissible iff, for each $a = (u, v) \in A$, $f_a(l_u) R f_a(l_v)$.

Proposition 6. (i) GL- $= [k]$ *is* WNL*-complete;* (ii) GL- $\leq [k]$ *is* WNL*-complete.*

Proof. Membership in WNL of these two problems can be shown by reduction to the NTMC$[k]$ problem. WNL-hardness of the GL- $= [k]$ problem follows from a series of reductions described in the Appendix. WNL-hardness of the GL- $\leq [k]$ problem is shown by reduction from GL- $= [k]$. Given an instance I of GL- $= [k]$ involving a $q \times k$-grid G, we construct an instance I' of GL- $\leq [k]$ involving a $2q \times 2k$-grid G'.

For each vertex v of G, we introduce the gadget G_v depicted in Fig. 1(A). It consists of four vertices v_1, v_2, v_3, v_4 and four arcs $a_1 = (v_1, v_2)$, $a_2 = (v_1, v_3)$, $a_3 = (v_2, v_4)$, $a_4 = (v_3, v_4)$. Given the set S_v associated to v in I, we associate to v_1, v_2, v_3, v_4 disjoint copies $S_1, \ldots, S_4$ of these sets, where $S_i = \{x_i : x \in S_v\}$. Let $s = |S_v|$, and let ϕ be a bijection of S_v into $[s]$. For odd i, we set $f_{a_i}(x_j) = \phi(x)$; for even i, we set $f_{a_i}(x_j) = s + 1 - \phi(x)$. It can be checked that in a $\leq$-admissible labelling of G_v, the four vertices must be labelled by the four copies of a same element.

We map each vertex v to a gadget G_v, and we interconnect these gadgets by arcs to form a grid G'. For each horizontal arc $a = (u, v)$ of G, we create two horizontal arcs $a' = (u_2, v_1)$ and $a'' = (u_3, v_4)$, as depicted in Fig. 1(B). We set $f_{a'}(x_j) = f_a(x)$ and $f_{a''}(x_j) = m + 1 - f_a(x)$. We proceed similarly for the vertical arcs. Then G' is a $2q \times 2k$-grid, and it can be checked that G has a $=$-admissible labelling iff G' has a $\leq$-admissible labelling. □

Proposition 7. *The* SLCS$[k]$ *problem is* WNL*-complete.*

Proof. Membership in WNL follows from the claim given in the proof of Proposition 1. This claim yields a reduction to NTMC$[k]$: given an instance $I = (\mathcal{C}, q, k)$ of SLCS$[k]$, we construct a nondeterministic Turing machine M which proceeds as follows. The first k cells of the tape store a position π in $\mathcal{C}$. The machine starts by nondeterministically choosing a position π. At each round, the machine seeks $a \in F(\pi)$, nondeterministically chooses a position π', checks that $\pi \to_a \pi'$, and overwrites π by π'. Then SLCS$(\mathcal{C}) \geq q$ iff M accepts using time $q' = O(kq)$ and space $\leq 2k$.

WNL-hardness is shown by a parameterized reduction from GL- $\leq [k]$. Let I be an instance of GL- $\leq [k]$, consisting of an integer m, of a $q \times k$-grid $G = (V, A)$, of a set S, of a partition $\{S_v : v \in V\}$ of S, and for each $a = (u, v) \in A$ of a function $f_a : S_u \cup S_v \to [m]$. For each $i \in [q]$, $j \in [k]$, let $v_{i,j}$ be the vertex of G in line i, column j. For each $1 \leq i \leq q$, $1 \leq j \leq k - 1$, let $a_{i,j}$ denote the horizontal arc $(v_{i,j}, v_{i,j+1})$. For each $1 \leq i \leq q - 1$, $1 \leq j \leq k$, let $a'_{i,j}$ denote the vertical arc $(v_{i,j}, v_{i+1,j})$.

We construct an instance $I' = (\mathcal{C}, q', k')$ of SLCS$[k]$ in the following way. We set $q' = 3kq - k - q$ and $k' = 2k + 4$. We define $\mathcal{C}$ as follows.

- Label set: we introduce labels $a[v, x]$ for each $v \in V$, $x \in S_v$, and $b[a, i]$ for each $a \in A$, $i \in [m]$.
- Sequences: we define two sequences s, s', two sequences t, t', and $2k$ sequences $u_1, u'_1, \ldots, u_k, u'_k$. Let $<_S$ be a total order on S. We first define the following sequences.

$$\forall v \in V, \quad A[v] = \prod_{x \in (S_v, <_S)} a[v, x], \quad \forall a \in A, \qquad B[a] = \prod_{i=1}^{m} b[a, i]$$

$$\forall a = (u, v) \in A,\ i \in [m], \quad C[a, i] = \prod_{x \in S_u : f_a(x) = i} a[u, x], \qquad C'[a, i] = \prod_{x \in S_v : f_a(x) = i} a[v, x].$$

The sequences of $\mathcal{C}$ are defined below.

– The sequences s, s' are control sequences; they constrain the shape of a compatible sequence for $\mathcal{C}$.

$$s_i = \left(\prod_{j=1}^{k-1} A[v_{i,j}] B[a_{i,j}]\right) A[v_{i,k}], \qquad s'_i = \left(\prod_{j=1}^{k-1} \widetilde{A[v_{i,j}]}\,\widetilde{B[a_{i,j}]}\right) \widetilde{A[v_{i,k}]}$$

$$s = \left(\prod_{i=1}^{q-1} s_i \left(\prod_{j=1}^{k} B[a'_{i,j}]\right)\right) s_q, \qquad s' = \left(\prod_{i=1}^{q-1} s'_i \left(\prod_{j=1}^{k} \widetilde{B[a'_{i,j}]}\right)\right) s'_q.$$

– The sequences t, t' are selection sequences; they ensure that the constraints corresponding to the horizontal arcs $a_{i,j}$ are satisfied.
$$t = \prod_{i=1}^{q}\prod_{j=1}^{k-1}\prod_{p=1}^{m} C[a_{i,j}, p]b[a_{i,j}, p], \qquad t' = \prod_{i=1}^{q}\prod_{j=1}^{k-1}\prod_{p=1}^{m} b[a_{i,j}, p]C'[a_{i,j}, p].$$
– For each $j \in [k]$, the sequences u_j, u'_j are selection sequences; they ensure that the constraints corresponding to the vertical arcs $a'_{i,j}$ are satisfied.
$$u_j = \prod_{i=1}^{q-1}\prod_{p=1}^{m} C[a'_{i,j}, p]b[a'_{i,j}, p], \qquad u'_j = \prod_{i=1}^{q-1}\prod_{p=1}^{m} b[a'_{i,j}, p]C'[a'_{i,j}, p].$$

The reduction is computable in polynomial time, and its correctness follows by proving the following.

Claim. G has a $\leq$-admissible labelling iff $\mathcal{C}$ has a compatible sequence of length q'.

Proof. ($\Rightarrow$): suppose that G has a $\leq$-admissible labelling. Let us introduce the following notations:

- for $1 \leq i \leq q$, $1 \leq j \leq k$, let $x_{i,j}$ be the value assigned to vertex $v_{i,j}$;
- for $1 \leq i \leq q$, $1 \leq j < k$, let $y_{i,j} = f_{a_{i,j}}(x_{i,j})$;
- for $1 \leq i < q$, $1 \leq j \leq k$, let $z_{i,j} = f_{a'_{i,j}}(x_{i,j})$.

Consider the sequence r defined as follows.

$$r = \left(\prod_{i=1}^{q-1} r_i \left(\prod_{j=1}^{k} b[a'_{i,j}, z_{i,j}]\right)\right) r_q \quad \text{where } r_i = \left(\prod_{j=1}^{k-1} a[v_{i,j}, x_{i,j}]b[a_{i,j}, y_{i,j}]\right) a[v_{i,k}, x_{i,k}]. \tag{1}$$

Then r has length $kq + k(q-1) + (k-1)q = q'$. We claim that r is a compatible sequence for $\mathcal{C}$. Clearly, r agrees with s and with s'. Moreover:

- r agrees with t; indeed,
$$r|L(t) = t|L(r) = \prod_{i=1}^{q}\prod_{j=1}^{k-1} a[v_{i,j}, x_{i,j}]b[a_{i,j}, y_{i,j}].$$
This follows by writing t as $t = \prod_{i=1}^{q}\prod_{j=1}^{k-1} t_{i,j}$, and by observing that r contains exactly two labels of $t_{i,j}$, which are $a[v_{i,j}, x_{i,j}]$ and $b[a_{i,j}, y_{i,j}]$. These labels appear in this order in r, and they appear in the same order in $t_{i,j}$, since $f_{a_{i,j}}(x_{i,j}) = y_{i,j}$.
- r agrees with t'; indeed,
$$r|L(t') = t'|L(r) = \prod_{i=1}^{q}\prod_{j=1}^{k-1} b[a_{i,j}, y_{i,j}]a[v_{i,j+1}, x_{i,j+1}].$$
This follows by writing t' as $t' = \prod_{i=1}^{q}\prod_{j=1}^{k-1} t'_{i,j}$, and by observing that r contains exactly two labels of $t'_{i,j}$, which are $b[a_{i,j}, y_{i,j}]$ and $a[v_{i,j+1}, x_{i,j+1}]$. These labels appear in this order in r. They appear in the same order in $t'_{i,j}$, since $f_{a_{i,j}}(x_{i,j+1}) \geq f_{a_{i,j}}(x_{i,j}) = y_{i,j}$.
- r agrees with u_j and with u'_j; by a similar argument.

($\Leftarrow$): suppose that G has a compatible sequence r of length q'. □

We make the following observations.

Observation 1: r has the form described in Eq. (1).

Proof. Since s and s' are complete sequences, r must be a subsequence of s and of s'. By definition of s, s', it can contain at most one letter in each block $A[v_{i,j}]$, $B[a_{i,j}]$, $B[a'_{i,j}]$. Since these blocks are q' in number, r contains in fact exactly one letter of each block; hence the result.

In the following observation, Points 1 and 2 respectively state that the constraints corresponding to the horizontal, resp. vertical, arcs are satisfied. □

Observation 2: The following hold.

1. For each $1 \leq i \leq q$, $1 \leq j < k$, if $a = a_{i,j}$, $x = x_{i,j}$, $x' = x_{i,j+1}$, $y = y_{i,j}$, then $f_a(x) \leq y \leq f_a(x')$.
2. For each $1 \leq i < q$, $1 \leq j \leq k$, if $a = a'_{i,j}$, $x = x_{i,j}$, $x' = x_{i+1,j}$, $y = z_{i,j}$, then $f_a(x) \leq y \leq f_a(x')$.

Proof. Consider Point 1. Let $1 \leq i \leq q$, $1 \leq j < k$. Observe that $a[v_{i,j}, x_{i,j}] <_r b[a_{i,j}, y_{i,j}] <_r a[v_{i,j+1}, x_{i,j+1}]$. Let $a = a_{i,j}$, $x = x_{i,j}$, $x' = x_{i,j+1}$, $y = y_{i,j}$. Since r agrees with t, we have $a[v_{i,j}, x_{i,j}] <_t b[a_{i,j}, y_{i,j}]$, which yields $f_a(x) \leq y$. Since r agrees with t', we have $b[a_{i,j}, y_{i,j}] <_{t'} a[v_{i,j+1}, x_{i,j+1}]$, which yields $f_a(x') \geq y$. □

The proof of Point 2 is similar, by observing this time that $a[v_{i,j}, x_{i,j}] <_r b[a'_{i,j}, z_{i,j}] <_r a[v_{i+1,j}, x_{i+1,j}]$, and by considering u_j, u'_j.

We conclude from the observations that the labelling of G which assigns $x_{i,j}$ to each vertex $v_{i,j}$ is a $\leq$-admissible labelling of G. □

5. Consequences for the parameterized complexity of other problems

As a by-product of our hardness results for the Slcs problem, we obtain new hardness results for the Lcs problem (Section 5.1), and other problems previously known to be "hard for the W-hierarchy" (Section 5.2).

5.1. Consequences for the Lcs problem

In this section, a sequence is a string *allowing* repetitions of letters, and we explicitly use the term p-sequence when considering strings with no repeated letter. We recall the definition of the Longest Common Subsequence problem.

Name: Longest Common Subsequence (Lcs).

Instance: A collection of k sequences $\mathcal{C} = \{s_1, \ldots, s_k\}$ on an alphabet Σ of size m, and an integer q.

Question: Do the sequences admit a common subsequence of length $\geq q$?

We first give a reduction from Slcs to Lcs, which relies on a simple padding argument.

Lemma 1. *There is a polynomial-time parameter-preserving reduction from* Slcs$[q, k]$ *to* Lcs$[q, k]$.

Proof. Given an instance $I = (\mathcal{C}, q, k)$ of Slcs$[q, k]$, we construct the following instance $I' = (\mathcal{C}', q, k)$ of Lcs$[q, k]$. Given the collection of p-sequences $\mathcal{C} = \{s_1, \ldots, s_k\}$ on L, we define the collection of sequences $\mathcal{C}' = \{s'_1, \ldots, s'_k\}$, where each s'_i is constructed from s_i by inserting, at the beginning, at the end, and between each two consecutive labels of s_i, a word u_i which is the qth power of an arbitrary enumeration of the labels of $L \setminus L(s_i)$.

The reduction is parameter preserving by definition, and is clearly polynomial. Its correctness follows by observing that $\mathcal{C}$ has a compatible sequence of length $\geq q$ iff the sequences of $\mathcal{C}'$ have a common subsequence of length $\geq q$. Indeed, if $s = a_1 \cdots a_q$ is a compatible sequence of $\mathcal{C}$, then s is a subsequence of the s'_i: since s agrees with s_i, the letters common to s and s_i appear in the same order in s'_i, and the remaining letters of s can be chosen in the blocks u_i of s'_i; hence s is a subsequence of s'_i. Conversely, if the sequences s'_i have a common subsequence $s = a_1 \cdots a_q$, then s is a compatible sequence of $\mathcal{C}$: since the letters common to s and s_i do not appear in the blocks u_i of s'_i, it follows that they must appear in the same order in s and s_i; hence s agrees with s_i. □

Lemma 1, together with Propositions 5 and 7, implies the following for the Lcs problem.

Proposition 8. (i) Lcs$[q, k]$ *is* W[1]*-complete;* (ii) Lcs$[k]$ *is* WNL*-complete;* (iii) Lcs$[k, m]$ *is* WNL*-complete.*

Proof. For Point (i), the hardness result follows from Proposition 5 and Lemma 1, and the membership result is shown in [14].

For Point (ii), the hardness result follows from Proposition 7 and Lemma 1. The membership result is shown by a parameterized reduction to NTMC$[k]$. Let $I = (\mathcal{C}, q, k)$ be an instance of Lcs$[k]$ with $\mathcal{C} = \{s_1, \ldots, s_k\}$. We construct a nondeterministic Turing machine M whose first k cells of the tape store a tuple $(p_1, \ldots, p_k)$ with p_i an integer between 0 and $|s_i|$. The machine starts with the tuple $(0, \ldots, 0)$ on its tape. At each round $i \leq q$, if the tuple on the tape is $t = (p_1, \ldots, p_k)$, the machine nondeterministically overwrites t by a tuple $t' = (p'_1, \ldots, p'_k)$ such that $p'_j > p_j$ for each $j \in [k]$, and $s_1[p'_1] = \cdots = s_k[p'_k]$. If no such tuple t' exists, the machine rejects. Then M accepts the empty string in $q' = O(kq)$ steps using space $\leq 2k$ iff the sequences s_i have a common subsequence of length q.

For Point (iii), hardness follows from a parameterized reduction from Lcs$[k]$ to Lcs$[k, m]$ described in [13], and membership follows from Point (ii). □

Proposition 8 improves known hardness results in several ways. First, it gives an alternative proof for the W[1]-hardness of Lcs$[q, k]$, simpler than the original proof of [14]. Second, it classifies precisely the complexity of Lcs$[k]$. The problem was only known to be W$[t]$-hard for each $t \geq 1$ [14], while we obtain a stronger WNL-completeness result.

5.2. Consequences for other problems

In [11,12], the W[t]-hardness result for the Lcs$[k]$ problem shown in [14] was transferred to other problems by parameterized reductions. Hence, our stronger WNL-hardness result for Lcs$[k]$ also holds for these problems.

Proposition 9. *The problems* Colored Cutwidth, Feasible Register Assignment, Domino Treewidth, Triangulating Colored Graphs *are* WNL*-hard.*

We now consider a problem on automata introduced in [20].

Name: Bounded Deterministic Automaton Intersection Intersection (Bdfa).

Instance: A family of k dfa $\mathcal{A} = \{A_1, \ldots, A_k\}$ on an alphabet Σ, and an integer q.

Question: Does there exist a word in Σ^q that is accepted by every A_i?

The Bdfa2 problem is the restriction of Bdfa to instances with $|\Sigma| = 2$.

Proposition 10. (i) *The* Bdfa$[k]$ *problem is* WNL*-complete;* (ii) *The* Bdfa2$[k]$ *problem is* WNL*-complete.*

Proof. Let us show Point (i). To show WNL-hardness, we reduce from Lcs$[k]$. Given an instance $I = (\mathcal{C}, q, k)$ of Lcs$[k]$, we create an instance $I' = (\mathcal{A}, q, k)$ of Bdfa$[k]$, where $\mathcal{A} = \{A_1, \ldots, A_k\}$ is such that, for each i, A_i is a dfa recognizing the set of subwords of s_i. To show membership in WNL, we reduce to NTMC$[k]$. Let $I = (\mathcal{A}, q, k)$ be an instance of Bdfa$[k]$ with $\mathcal{A} = \{A_1, \ldots, A_k\}$, $A_i = (Q_i, \Sigma, \delta_i, q_i^0, F_i)$. We construct a nondeterministic Turing machine M whose first k cells of the tape represents a tuple $t = (q_1, \ldots, q_k)$, with $q_i \in Q_i$. The machine starts with the tuple $t = (q_1^0, \ldots, q_k^0)$ on its tape. At each round $i \leq q$, if the tuple on the tape is $t = (q_1, \ldots, q_k)$, the machine nondeterministically chooses a letter $a \in \Sigma$, and overwrites t by the new tuple $t' = (\delta_1(q_1, a), \ldots, \delta_k(q_k, a))$. At the end of round q, the machine accepts iff the tuple written on the tape has the form $t = (q_1, \ldots, q_k)$ with $q_i \in F_i$ for each i.

Point (ii) is easy to see: membership in WNL follows from Point (i), and WNL-hardness follows by a simple reduction from Bdfa$[k]$ described in [20]. □

6. Concluding remarks

We first mention some open questions regarding the approximability and the parameterized complexity of Slcs. First, we have obtained a k-approximation for the CSlcs problem in Proposition 4: is it optimal or can it be improved, e.g. to $\Omega(\log k)$? Second, is the problem solvable in $p^{O(k)}n^c$ time? An algorithm with this running time would be preferable to the algorithm of Proposition 4, for small k. A third question concerns the existence of polynomial kernels for the CSlcs problem: while the kernelizability for the single parameter p is equivalent to an open question for Directed Feedback Vertex Set, it may be the case that the problem admits a polynomial kernel for both parameters k, p.

Another area of investigation of general interest would be to extend the applicability of the WNL class to other problems. Aside from the problems considered in this paper, there are other problems known to be hard for the W-hierarchy, which would be natural candidates to WNL-hardness. Prominent examples are the Bandwidth problem, which is W$[t]$-hard even for trees [11], and the Shortest Common Supersequence problem [21]. Finally, we would like to point out that the Lcs problem on bounded alphabets, parameterized by the number of sequences, is only known to be W[1]-hard [19]. Since this seems a weak result in view of the structure of the problem, it seems natural to conjecture that it could be hard for the W-hierarchy or even for the WNL class.

Acknowledgement

We are grateful to Vincent Berry and Eric Rivals for their careful proofreading.

Appendix

This appendix describes the series of reductions leading to the WNL-hardness of GL- $= [k]$.

For convenience, we consider nondeterministic Turing machines having one initial state and one final state. We recall that this a tuple $M = (Q, \Sigma, \perp, q_i, q_f, \Delta)$, where Q is the set of states, Σ is the tape alphabet, $\perp \in \Sigma$ is the blank symbol, $q_i \in Q$ is the initial state, $q_f \in Q$ is the final state, and Δ is a set of transitions of the form $q \rightarrow^{a/b,d} q'$ with $q, q' \in Q$, $a, b \in \Sigma$, $d \in \{L, R\}$. Given $i \in \mathbb{N}$ and $d \in \{L, R\}$, we define $s(i, d)$ as $i + 1$ if $d = R$, and as $i - 1$ if $d = L$ and $i > 0$.

We say that an execution of M *strongly accepts* iff it starts in q_i with the head on position 0 and the tape being empty, and ends in q_f with the head on position 0 and the tape being empty. We consider the following variant of NTMC.

Name: NTMC-2.

Instance: A nondeterministic Turing machine M, and integers q, k.

Question: Does M strongly accept in q steps by examining at most k cells?

Given two biparameterized problems Π, Π', we say that Π *strongly fpt-reduces* to Π' iff there exists an fpt-reduction from Π to Π' which maps an instance (I, q, k) of Π to an instance (I', q', k') of Π', with $q' \leq f(k, q)$ and $k' \leq g(k)$.

Proposition 11. NTMC$[q, k] \leq_{sfpt}$ NTMC-2$[q, k]$.

Proof. Given an instance $I = (M, q, k)$ of NTMC$[q, k]$, we define an instance $I' = (M', q, k')$ of NTMC-2$[q, k]$, where M' simulates M and, when a final state of M has been reached, proceeds to a "cleaning" step which erases the contents of the tape, moves back the head in position 0, and enters the final state of M'. Then M accepts the empty string in q steps iff M' accepts the empty string in $q' = q + 2k$ steps, with the same space usage. □

We now introduce an intermediary problem on existential (one-player) pebble games. We define a *k-pebble game* as a tuple $G = (V, T, R, I, F)$, where V is a set of vertices, T is a set of transitions, $R = \{\rightarrow_t : t \in T\}$ is a set of binary relations on V, $I \in V^k$ is the initial configuration, and $F \in V^k$ is the final configuration. A *configuration* of G is a tuple $C \in V^k$, where $C[i]$ is the position of the ith pebble. Given two configurations C, C' of G and a transition $t \in T$, we denote $C \rightarrow_t C'$ iff $C[i] \rightarrow_t C'[i]$ for each $i \in [k]$. A *play* of G is a chain of configurations $C_0 \rightarrow_{t_1} C_1 \rightarrow_{t_2} \cdots \rightarrow_{t_q} C_q$ such that $C_0 = I$, $C_q = F$. The *length* of the play is q.

We consider the following problem.

Name: Existential Pebble Game (EPG).

Instance: A k-pebble game G, and an integer q.

Question: Does G have a play of length q?

Proposition 12. NTMC-2$[q, k] \leq_{sfpt}$ EPG$[q, k]$.

Proof. Let $I = (M, q, k)$ be an instance of NTMC-2$[q, k]$ with $M = (Q, \Sigma, \perp, q_i, q_f, \Delta)$. We construct an instance $I' = (G, q', k')$ of EPG$[q, k]$, with $q' = 4q$ and $k' = k + 1$.

The intuition is that an execution of M will be simulated by a play in G, with each step of M simulated by four steps in G. A configuration C of G will be a tuple with $k + 1$ components, and will encode a configuration of M: the first component of C will represent the state of the machine, and the k other components will represent the contents of the tape.

The pebble game is formally defined as $G = (V, T, R, I, F)$. We set $V = V_0 \cup V_1 \cup \cdots \cup V_k$, where

- V_0 contains a set of pairs $(0, s)$, where s is a term representing the current state of the machine, and takes the following values: (i) $s = idleStep(q, i)$ with $q \in Q, i \in [k]$; (ii) $s = readStep(q, i, v)$ with $q \in Q, i \in [k], v \in \Sigma$; (iii) $s = writeStep(q, i, j, v)$ with $q \in Q, i, j \in [k], v \in \Sigma$.
- for each $i \in [k]$, V_i contains the set of pairs (i, x) with $x \in \Sigma$.

A configuration of G is a tuple $C = (v_0, v_1, \ldots, v_k)$ with $v_i \in V_i$. The initial configuration is $I = ((0, idleStep(q_i, 1)), (1, \perp), \ldots, (k, \perp))$, the final configuration is $F = ((0, idleStep(q_f, 1)), (1, \perp), \ldots, (k, \perp))$.

The transitions of T have a particular form. Each transition can move the pebble i from x to y, conditioned by the presence of the pebble j in z. Such a transition is be denoted by $t = (i, j, x, y, z)$. The corresponding relation $\rightarrow_t$ is therefore defined as follows: (i) $(i, x) \rightarrow_t (i, y)$, (ii) $(j, z) \rightarrow (j, z)$, (iii) for each $p \neq i, j$, and for each $v \in V_p$, $v \rightarrow_t v$.

We now define the transitions of T. A step of M in position i of the tape is decomposed into four steps of G: (i) reading the letter a in position i of the tape, and moving pebble 0 from $(0, idleStep(q, i))$ to $(0, readStep(q, i, a))$; (ii) choosing a transition $q \rightarrow^{a/b,d} q'$ of M, and moving pebble 0 into state $(0, writeStep(q', i, i', b))$ with $i' = s(i, d)$; (iii) writing the letter b in position i of the tape, thus moving pebble i from (i, a) to (i, b); (iv) once the writing is done, moving pebble 0 into state $(0, idleStep(q', i'))$. Therefore, the transitions of T are as follows.

- $(0, i, idleStep(q, i), readStep(q, i, a), a)$ for $q \in Q, i \in [k], a \in \Sigma$.
- $(0, i, readStep(q, i, a), writeStep(q', i, i', b), a)$ for $q \in Q, i \in [k], a \in \Sigma$, and for a transition $q \rightarrow^{a/b,d} q'$ of M, with $i' = s(i, d)$.
- $(i, 0, a, b, writeStep(q', i, i', b))$ for $q' \in Q, i, i' \in [k], a, b \in \Sigma$.
- $(0, i, writeStep(q', i, i', b), idleStep(q', i'), b)$ for $q' \in Q, i, i' \in [k], b \in \Sigma$.

It is easy to see that the reduction is polynomial, and that M accepts in q steps using space $\leq k$ iff G has a play of length q'. □

Proposition 13. EPG$[q, k] \leq_{\text{sfpt}}$ GL- $= [q, k]$.

Proof. Let $I = (G, q, k)$ be an instance of EPG$[q, k]$, where $G = (V, T, R, I, F)$ is a k-pebble game, with V set of vertices, T a set of transitions, $R = \{\rightarrow_t : t \in T\}$, $I \in V^k$ the initial configuration, and $F \in V^k$ the final configuration. Without loss of generality, we assume that V and T are subsets of $[m]$ for some integer m. We create an instance I' of GL- $= [q, k]$.

The intuition is that a play of G will be encoded by a labelling of a $q \times k$-grid, where the labelling of the ith line of the grid represents the configuration of the game at step i. The set of labels will consist of elements of the form $M_{i,t,j,x,y}$; the meaning of a label is that, at step i, the transition t is chosen, and the pebble j goes from vertex x to vertex y.

Formally, I' consists of the following

- The integer m.
- A $q \times k$-grid $H = (V, A)$, whose vertex in line i, column j is denoted by $v_{i,j}$.
- A set S consisting of elements $M_{i,t,j,x,y}$ for $i \in [q], t \in T, j \in [k], x, y \in V$ such that $x \rightarrow_t y$. For $i = 1$, the element is present iff $x = I[j]$. For $i = q$, the element is present iff $y = F[j]$.
- A partition consisting of the sets $S_{v_{i,j}} = \{M_{i',t,j',x,y} \in S : i' = i, j' = j\}$.
- For each arc $a = (u, v) \in A$, a function $f_a : S_u \cup S_v \rightarrow [m]$ defined as follows. For an horizontal arc $a = (v_{i,j}, v_{i,j+1})$, we have $f_a(M_{i,t,j,x,y}) = t$ and $f_a(M_{i,t,j+1,x,y}) = t$. For a vertical arc $a = (v_{i,j}, v_{i+1,j})$, we have $f_a(M_{i,t,j,x,y}) = y$ and $f_a(M_{i+1,t,j,x,y}) = x$.

The reduction is clearly polynomial, and we verify that I is a positive instance of EPG iff I' is a positive instance of the GL- $=$ problem. □

References

[1] D. Maier, The complexity of some problems on subsequences and supersequences, Journal of the ACM 25 (2) (1978) 322–336.
[2] R.G. Downey, M.R. Fellows, Parameterized Complexity, Springer-Verlag, 1999.
[3] R. Niedermeier, Invitation to Fixed-Parameter Algorithms, Oxford University Press, 2006.
[4] J. Flum, M. Grohe, Parameterized Complexity Theory, Springer-Verlag, 2006.
[5] M.R. Fellows, M.T. Hallett, U. Stege, Analogs & duals of the MAST problem for sequences & trees, Journal of Algorithms 49 (1) (2003) 192–216.
[6] B.M.E. Moret, J. Tang, T. Warnow, Reconstructing phylogenies from gene-content and gene-order data, in: O. Gascuel (Ed.), Mathematics of Phylogeny and Evolution, Oxford University Press, 2004.
[7] J. Hodge, R.E. Klima, The Mathematics of Voting and Elections: A Hands-on Approach, in: Mathematical World, vol. 22, AMS, 2000.
[8] C. Dwork, R. Kumar, M. Naor, D. Sivakumar, Rank aggregation methods for the web, in: WWW10, 2001.
[9] C. Dwork, R. Kumar, M. Naor, D. Sivakumar, Rank aggregation revisited, 2001.
[10] R. Fagin, R. Kumar, D. Sivakumar, Efficient similarity search and classification via rank aggregation, in: Proc. ACM SIGMOD'03, 2003, pp. 301–312.
[11] H.L. Bodlaender, M.R. Fellows, M.T. Hallett, Beyond NP-completeness for problems of bounded width: hardness for the W hierarchy (extended abstract), in: Proc. STOC'94, ACM, 1994, pp. 449–458.
[12] H.L. Bodlaender, M.R. Fellows, M.T. Hallett, The hardness of perfect phylogeny, feasible register assignment and other problems on thin colored graphs, Theoretical Computer Science 244 (1) (2000) 167–188.
[13] H.L. Bodlaender, R.G. Downey, M.R. Fellows, M.T. Hallett, H.T. Wareham, Parameterized complexity analysis in computational biology, Computer Applications in the Biosciences 11 (1) (1995) 49–57.
[14] H.L. Bodlaender, R.G. Downey, M.R. Fellows, H.T. Wareham, The parameterized complexity of sequence alignment and consensus, Theoretical Computer Science 147 (1–2) (1994) 31–54.
[15] P.D. Seymour, Packing directed circuits fractionally, Combinatorica 15 (1995) 281–288.
[16] G. Even, J. Naor, B. Schieber, M. Sudan, Approximating minimum feedback sets and multicuts in directed graphs, Algorithmica 20 (1998) 151–174.
[17] J. Chen, Y. Liu, S. Lu, B. O'Sullivan, I. Razgon, A fixed-parameter algorithm for the directed feedback vertex set problem, in: Proc. STOC'08, 2008, pp. 177–186.
[18] L. Cai, J. Chen, R.G. Downey, M.R. Fellows, On the parameterized complexity of short computation and factorization, Archive for Mathematical Logic 36 (4–5) (1997) 321–337.
[19] K. Pietrzak, On the parameterized complexity of the fixed alphabet shortest common supersequence and longest common subsequence problems, Journal of Computer and System Sciences 67 (4) (2003) 757–771.
[20] H.T. Wareham, The parameterized complexity of intersection and composition operations on sets of finite-state automata, in: Proc. ICIAA'00, in: LNCS, vol. 2088, 2000, pp. 302–310.
[21] M.T. Hallett, An integrated complexity analysis of problems from computational biology, Ph.D. Thesis, Department of Computer Science, University of Victoria, Victoria, BC, Canada, 1996.